MWCT2xx2A Reference Manual

Supports MWCT2xx2A and MWCT2xx2

Document Number: MWCT2XX2ARM
Rev. 2, 12/2023

Contents

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

Chapter 1
About This Document

Chapter 2
Introduction

Chapter 3
Memory Map

Chapter 4
Clock Distribution

Chapter 5
ROM Bootloader

Chapter 6
Power Management

Chapter 7
Memory Resource Protection (MRP)

Chapter 8
Miscellaneous Control Module (MCM)

Chapter 9
System Integration Module (SIM)

Chapter 10

Interrupt Controller (INTC)

Chapter 11
Enhanced Direct Memory Access (eDMA)

Chapter 12
DMA Channel Multiplexer (DMAMUX)

Chapter 13
Power Management Controller (PMC)

Chapter 14
Event Generator (EVTG)

Chapter 15
Inter-Peripheral Crossbar Switch (XBAR)

Chapter 16
On-Chip Clock Synthesis (OCCS)

Chapter 17
Flash Memory Controller (FMC)

Chapter 18
Flash Memory Module (FTFA)

Chapter 19
Computer Operating Properly (COP) Watchdog

Chapter 20
External Watchdog Monitor (EWM)

Chapter 21
Cyclic Redundancy Check (CRC)

Chapter 22
12-bit Cyclic Analog-to-Digital Converter (ADC)

Chapter 23
Comparator (CMP)

Chapter 24
12-bit Digital-to-Analog Converter (DAC)

Chapter 25
Operational Amplifier (OPAMP)

Chapter 26
Enhanced Flexible Pulse Width Modulator (PWM)

Chapter 27
Quad Timer (TMR)

Chapter 28
Periodic Interrupt Timer (PIT)

Chapter 29
Queued Serial Communications Interface (QSCI)

Chapter 30
Queued Serial Peripheral Interface (QSPI)

Chapter 31
Low Power Inter-Integrated Circuit (LPI2C)

Chapter 32
General-Purpose Input/Output (GPIO)

Chapter 1
About This Document

1.1 Overview

1.1.1 Purpose

This document describes the features, architecture, and programming model of MWCT2xx2A devices.

1.1.2 Audience

This document is primarily for system architects and software application developers who are using or considering using these devices 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."

Chapter 2
Introduction

2.1 Target Applications

The series of MWCT2xx2A is an automotive qualified wireless power transmitter controller that integrates all required functions for Wireless Power Consortium (WPC) “Qi” compliant wireless power transmitter design. It is a safe and secure device which utilizes the DSASS security module to support on-chip Qi authentication.

2.2 System Block Diagram

NOTE

The following figure shows the maximum memory configurations supported.

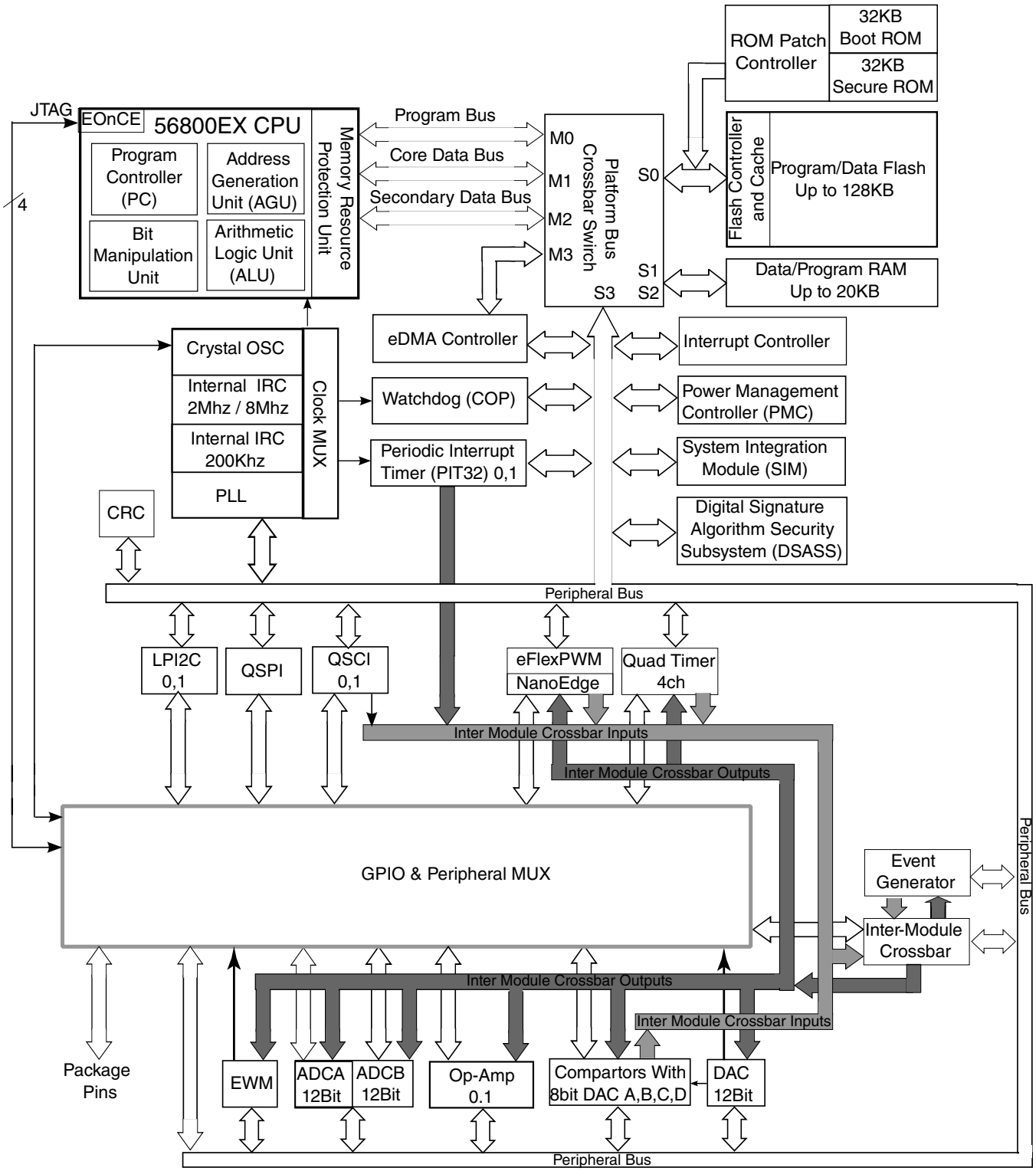


Figure 2-1. System block diagram

2.3 Product Family

See the device datasheet for detailed feature table and comparison, as well as the pinout information.

Chapter 3
Memory Map

3.1 Introduction

This device contains various memories and memory-mapped peripherals which are located in one contiguous memory space.

3.2 Program/Data Memory Maps

The 56800EX core processor is word addressed for the program memory space where a word is 16 bits. Separate memory maps are supported for program space accessed by the Program Data Bus (PDB) and for data space accessed by the primary and secondary data bus (XDB1, XDB2).

NOTE

BothTable 3-1 and Table 3-2 include:

- Primary program/data flash/Boot ROM
- Program/data RAM /data ROM

Each of these memories consists of a single physical flash memory that is mirrored across the program and data memory maps.

Table 3-1. Program Memory Map with Large program RAM address mode - (Used by core PDB)

Range	Size (Words)	Use
0x00_0000–0x00_FFFF	64KW	Primary program/data flash array
0x01_0000–0x05_FFFF	320KW	Reserved
0x06_0000–0x06_27FF	10KW	Program/data RAM (mapped from data memory space 0x00_0000–0x00_27FF)[1]
0x06_2800–0x07_7FFF	86KW	Reserved
0x07_8000–0x07_FFFF	32KW	BOOT and Security ROM[2]

Table continues on the next page...

Table 3-1. Program Memory Map with Large program RAM address mode - (Used by core PDB) (continued)

Range	Size (Words)	Use
0x08_0000–0x1F_FFFF	1536KW	Reserved

1. In order to utilize 19bit address mode, RAM is placed in address range below 0x08_0000
2. In order to utilize 19bit address mode, Boot ROM is placed in address range below 0x08_0000

Table 3-2. Data Memory Map- (Used by core XDB1, XDB2)

Range	Size (Words)	Use
0x00_0000–0x00_27FF	10KW	Program/data RAM
0x00_2800–0x00_7FFF	22KB	Reserved
0x00_8000–0x00_BFFF	16KW	Primary program/data flash (Mapped from program memory space 0x00_C000 - 0X00_FFFF)
0x00_C000–0x00_DFFF	8KW	Core and system peripherals
0x00_E000–0x00_FFFF	8KW	Slave peripherals
0x01_0000–0x01_FFFF	64KW	Reserved
0x02_0000–0x02_FFFF	64KW	Primary program/data Flash (Mapped from program memory space 0x00_0000 - 0X00_FFFF)
0x03_0000–0x07_FFFF	320KW	Reserved
0x07_8000–0x07_FFFF	32KW	BOOT and Security ROM / DATA ROM (Mapped from program memory space 0x07_8000 - 0X07_FFFF)
0x08_0000–0xFF_FEFF	15.5MW	Reserved
0xFF_FF00–0xFF_FFFF	128W	EOnCE registers

Flash is mapped into data memory space from 0x00_8000 to 0x00_BFFF and 0x02_0000-0x02_FFFF. Access of data flash is 25 MHz.

BOOT ROM is mapped into data memory space from 0x07_8000 to 0x07_FFFF. Access of data ROM is 50Mhz at normal CPU mode and 100Mhz at fast CPU mode.

3.3 Core and System Peripheral Memory Map

The core and system peripheral memory map is the portion of the data-space memory map assigned to core and system peripherals.

Table 3-3. Memory Map for On-Platform Peripherals

Peripheral	Base Address (in words)	Size (in words)
Core Configuration MCM	0x00_c000	512w

Table continues on the next page...

Table 3-3. Memory Map for On-Platform Peripherals (continued)

Peripheral	Base Address (in words)	Size (in words)
Reserved	0x00_c200	512w
	0x00_c400	512w
Reserved	0x00_c600	512w
eDMA2 - Control/Status/Error/priority register array	0x00_c800	512w
	0x00_ca00	512w
	0x00_cc00	512w
	0x00_ce00	512w
eDMA2 - TCD array	0x00_d000	512w
	0x00_d200	512w
	0x00_d400	512w
	0x00_d600	512w
Reserved	0x00_d800	512w
Reserved	0x00_da00	512w
Reserved	0x00_dc00	512w
Platform Flash Controller (PFLSHC)	0x00_de00	512w

3.4 Slave Peripheral Memory Map

The slave peripheral memory map is the portion of the data-space memory map assigned to slave peripherals.

Table 3-4. Memory Map for Slave Peripherals

Peripheral	Instance Name	Base Address (in words)	Size (in words)
12-bit DAC	DAC	0x00_e000	16
Reserved		0x00_e010	16
Comparator (with 8-bit reference DAC)	CMPA	0x00_e020	8
Comparator (with 8-bit reference DAC)	CMPB	0x00_e028	8
Comparator (with 8-bit reference DAC)	CMPC	0x00_e030	8
Comparator (with 8-bit reference DAC)	CMPD	0x00_e038	8
Reserved		0x00_e040	32
Operational Amplifier A	OPAMPA	0x00_e060	8
Operational Amplifier B	OPAMPB	0x00_e068	8
Reserved		0x00_e070	16
Queued Serial Communications Interface module	QSCI0	0x00_e080	16
Queued Serial Communications Interface module	QSCI1	0x00_e090	16
Reserved		0x00_e0a0	16

Table continues on the next page...

Table 3-4. Memory Map for Slave Peripherals (continued)

Peripheral	Instance Name	Base Address (in words)	Size (in words)
Queued Serial Peripheral Interface module	QSPI	0x00_e0b0	16
Reserved		0x00_e0c0	32
Programmable Interval Timer	PIT0	0x00_e100	16
Programmable Interval Timer	PIT1	0x00_e110	16
Reserved		0x00_e120	32
General Purpose Timer	TMRA0	0x00_e140	16
General Purpose Timer	TMRA1	0x00_e150	16
General Purpose Timer	TMRA2	0x00_e160	16
General Purpose Timer	TMRA3	0x00_e170	16
Reserved		0x00_e180	64
Reserved		0x00_e1c0	32
Reserved		0x00_e1e0	32
General Purpose I/O	GPIOA	0x00_e200	16
General Purpose I/O	GPIOB	0x00_e210	16
General Purpose I/O	GPIOC	0x00_e220	16
General Purpose I/O	GPIOD	0x00_e230	16
General Purpose I/O	GPIOE	0x00_e240	16
General Purpose I/O	GPIOF	0x00_e250	16
Reserved		0x00_e260	64
Power Management Controller	PMC	0x00_e2a0	16
On-chip Clock Control System	OCCS	0x00_e2b0	16
Reserved		0x00_e2c0	64
Interrupt Controller	INTC	0x00_e300	32
Windowed Computer Operating Properly	WCOP	0x00_e320	16
External Watchdog Monitor	EWM	0x00_e330	16
Inter-Peripheral Crossbar Switch	XBAR	0x00_e340	64
Event Generator	EVTG_A	0x00_e380	8
Event Generator	EVTG_B	0x00_e388	8
Event Generator	EVTG_C	0xxx_e390	8
Event Generator	EVTG_D	0x00_e398	8
Cyclical Redundancy Check	CRC	0x00_e3a0	16
eDMA input Multiplex0	DMA_MUX	0x00_e3b0	4
Reserved		0x00_e3b4	12
Flash Memory interface	FTFA	0x00_e3c0	48
Reserved		0x00_e3f0	16
System Integration Module	SIM	0x00_e400	128
LP Inter-Integrated Circuit module	LPI2C0	0x00_e480	64
LP Inter-Integrated Circuit module	LPI2C1	0x00_e4C0	64
12-bit Cyclic ADC	ADC_CYC	0x00_e500	128

Table continues on the next page...

Table 3-4. Memory Map for Slave Peripherals (continued)

Peripheral	Instance Name	Base Address (in words)	Size (in words)
Reserved		0x00_e580	128
Pulse Width Modulator with nano-edge placement	eFlexPWMA	0x00_e600	256
Reserved		0x00_e700	6400

Chapter 4
Clock Distribution

4.1 Overview

Clock generation is performed by two modules: On-Chip Clock Synthesis (OCCS) and System Integration Module (SIM). OCCS generates the clocks and SIM controls the distribution of the clocks to different circuitry. CPU and associated memories are able to operate in either normal system clock rate which is maximum 50MHz, or fast system clock rate which is two times normal system clock rate.

The OCCS encapsulates all on-chip oscillators, a crystal oscillator and a PLL. Its role is to provide to the SIM module a master clock (MSTR_2X_CLK) running at two times the system clock (SYS_CLK) rate. The OCCS's internal oscillators are also outputs to WCOP, EWM and PIT modules for use as secondary clock sources. A range of 160MHz ~ 240MHz clock from PLL is directly provided to eFlexPWM NanaEdge if it is implemented in device.

The OCCS module includes:

- 8 MHz / 2 MHz Internal Reference Clock (IRC) oscillator
- 200 kHz internal reference clock (IRC) oscillator
- 8 MHz to 16 MHz crystal oscillator
- Phase lock loop (PLL) with lock detection
- Loss of reference clock detection

The SIM uses the MSTR_2X_CLK clock from OCCS to generate all system and peripheral bus clocks for the device. The MSTR_2X_CLK clock is divided by two to generate SYS_CLK. MSTR_2X_CLK clock also generates BUS_CLK for clocking the peripherals, which either is divided by two from MSTR_2X_CLK when CPU operates at normal system clock rate, or is divided by four from MSTR_2X_CLK when CPU operates at fast system clock rate. A fast speed peripheral bus clock (BUS_2X_CLK) which is two times bus clock rate is also available. The following peripherals can select clock rate in the SIM module between BUS_CLK and BUS_2X_CLK:

- Quad Timer (TMR)

- Queued SCIs (QSCIs)
- eFlexPWM/FlexPWM
- LPI2C Functional clock

Table 4-1. Clock Summary

Clock	Description	Originate		Comments
		Normal mode	Fast speed mode	
MSTR_2X_CLK	Master Clock	OCCS (upto 100MHz)	OCCS (Upto 200MHz)	Maximum frequency governed by PLL and postscaler of OCCS
SYS_CLK	System Clock	MSTR_2X_CLK /2	MSTR_2X_CLK /2	Maximum frequency governed by the master clock
BUS_CLK	Bus Clock **NOTE:** also called IPBus Clock in peripheral modules	MSTR_2X_CLK /2	MSTR_2X_CLK /4	Up to 50MHz at both modes
BUS_2X_CLK	High Speed Bus Clock	MSTR_2X_CLK	MSTR_2X_CLK /2	Up to100MHz at both modes
CPU_CLK	CPU clock	SYS_CLK	SYS_CLK	For CPU and associated memories
PWM_2X_CLK	NanoEdge PWM Clock	PLL	PLL	160MHz to 240MHz
$F_{rosc8MHz}$	High Speed IRC Clock	8MHz IRC	8MHz IRC	8MHz (Typ)
$F_{rosc200K}$	Low Speed IRC Clock	200KHz IRC	200KHz IRC	200KHz (Typ)
F_{cosc}	Crystal Oscillator Clock	Crystal Oscillator	Crystal Oscillator	4MHz to 16MHz

4.2 Clock Distribution

Custom system and peripheral bus clocks are output by SIM and consumed by the various peripherals and system-level modules for which they are intended. Some modules can be selected in SIM to operate at 2X bus clock (BUS_2X_CLK). No matter operating at bus clock or 2X bus clock, the module register accesses are always at bus lock rate. The following table summarizes the clocks that can be used by each of the modules.

Table 4-2. Clock Distribution

Module	Primary clock		Additional Clocks	Comments
	Normal	High Speed		
System				
INTC	SYS_CLK			
eDMA	BUS_CLK			
SIM	BUS_CLK			

Table continues on the next page...

Table 4-2. Clock Distribution (continued)

WCOP	BUS_CLK		$F_{rosc8Mhz}$ $F_{rosc200K}$ F_{cosc}	Additional clock selection is in module
EWM	BUS_CLK		$F_{rosc8Mhz}$ $F_{rosc200K}$ F_{cosc}	Additional clock selection is in module
CRC	BUS_CLK			
EVTG	BUS_CLK			
XBAR	BUS_CLK			
Human-Machine Interface (HMI)				
GPIO	BUS_CLK			
Analog				
PMC	BUS_CLK			
ADC	BUS_CLK			
DAC	BUS_CLK			
HCMP	BUS_CLK			
OPAMP	BUS_CLK			
Timers				
eFlexPWM	BUS_CLK	BUS_2X_CLK	PWM_2X_CLK	NanoEdge can only be operated when primary clock is configured to BUS_2X_CLK in SIM.
TMR	BUS_CLK	BUS_2X_CLK		Clock selection is in SIM
PIT32	BUS_CLK		$F_{rosc8Mhz}$ $F_{rosc200K}$ F_{cosc}	Clock selection is in module
Communication Interfaces				
QSPI	BUS_CLK			
QSCI	BUS_CLK	BUS_2X_CLK		Clock selection is in SIM
LPI2C	BUS_CLK	BUS_2X_CLK		Clock selection is in SIM

4.3 Dual speed clock modes

Following are the clocking modes.

1. **Normal mode**: In this mode core:system:bus frequency ratio is maintained at 1:1:1 (each 50 MHz max).
2. **Fast mode**: In this mode core:system:bus frequency ratio is maintained at 2:2:1 (maximum core frequency is 100 MHz).

4.3.1 Sequence involving Run mode switching

The run mode switching must follow a sequence; the violation of the sequence may cause the flash access clock frequency exceeding the specifications defined, which is maximum 25 MHz.

- **Normal mode to fast mode switching sequence**
  a. Set SIM_MISC0[FAST_MODE] to set ratio of core and bus to 2:1.
  b. Issue a software reset by writing 1 to SIM_CTRL[SWRST].
  c. Core clock frequency can be increased beyond 50 MHz by updating PLL clock output.
- **Fast mode to normal mode switching sequence**
  a. Core clock frequency must be reduced to below 50 MHz by updating PLL clock output.
  b. Clear SIM_MISC0[FAST_MODE] to set ratio of core and bus to 1:1.
  c. Issue a software reset by writing 1 to SIM_CTRL[SWRST].

NOTE

- Power-on-reset causes the chip to start up from the normal mode. Then the run mode is corresponding with SIM_MISC0[FAST_MODE], after software reset.
- Other type of reset, such as pin reset or COP reset, causes the chip to start up from the current run mode. E.g. if the current run mode is fast mode, it will still be fast mode after such reset.

Chapter 5
ROM Bootloader

5.1 Chip-specific Information

This chapter describes only part of the information for your device's specific ROM Bootloader. For more detailed information and explanations of how the ROM Bootloader works, refer to the Bootloader Reference Manual.

NOTE

The ROM Bootloader is designed at core frequency in normal mode. When the chip jumps to ROM bootloader at core frequency in fast mode, the clock frequency values mentioned in this chapter need to be halved.

5.1.1 Bootloader Peripheral Pinmux

This device has various peripherals (UART, I2C, etc.) supported by the ROM Bootloader. To use an interface for bootloader communications, the peripheral must be enabled in the BCA, see the "Bootloader Configuration Area (BCA)" section. If the BCA is invalid (such as all 0xFF bytes), then all peripherals will be enabled by default. The following table shows the pins used by the ROM Bootloader.

Table 5-1. Bootloader Peripheral Pinmux (for 64-pin package)

Peripheral	Instance	Alt Mode	Pins
UART/SCI	0	0	GPIOC2, QSCI0_TXD
		2	GPIOC3, QSCI0_RXD
UART/SCI	1	2	GPIOC11, QSCI1_TXD
		2	GPIOC12, QSCI1_RXD
LPI2C	0	0	GPIOC14, LPI2C0_SDA
		0	GPIOC15, LPI2C0_SCL

5.1.2 Bootloader Memory Access

While executing, the Bootloader uses the following address ranges in ROM and RAM memory:

- 0x07_8000 – 0x07_BFFF (Boot ROM, 16 K Words)
- 0x00_0000 – 0x00_09C0 [1] (RAM in Data memory map space), but this part of RAM can still be utilized in the user application.

5.2 Introduction

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

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

For the device, the Bootloader can interface with LPI2C, and UART/SCI peripherals in slave mode and respond to the commands sent by a master (or host) communicating on one of those ports. The host/master can be a firmware-download application running on a PC or an embedded host communicating with the Bootloader. Regardless of the host/master (PC or embedded host), the Bootloader always uses a command protocol to communicate with that host/master. Commands are provided to write to memory (internal flash or RAM), erase flash, and get/set bootloader options and property values. The host application can query the set of available commands.

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

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

Features supported by the Bootloader in ROM:

- Supports LPI2C, and UART/SCI peripheral interfaces

1. In byte addressing. Uses can get ROM reserved region by the command `get-property 12`.

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

Table 5-2. Commands supported by the Bootloader in ROM

Command	Description	When flash security is enabled, then this command is
Execute	Run user application code that never returns control to the bootloader	Not supported
FlashEraseAll	Erase the entire flash array	Not supported
FlashEraseRegion	Erase a range of sectors in flash	Not supported
WriteMemory	Write data to memory	Not supported
ReadMemory	Read data from memory	Not supported
FlashProgramOnce	Writes data provided in a command packet to a specified range of bytes in the program once field	Not supported
FlashReadOnce	Returns the contents of the program once field by given index and byte count	Not supported
FlashReadResource	Returns the contents of the IFR field or Flash Version ID, by given offset, byte count and option	Not supported
FlashSecurityDisable	Attempt to unlock flash security using the backdoor key	Supported
GetProperty	Get the current value of a property	Supported
Reset	Reset the chip	Supported
FlashEraseAllUnsecure	Erase the entire flash array, including protected sectors	Supported

5.3 Functional Description

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

5.3.1 The Bootloader Configuration Area (BCA)

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

Table 5-3. Configuration Fields for the Bootloader

Offset	Size (bytes)	Configuration Field	Description
0x00 - 0x03	4	tag	Magic number to verify bootloader configuration is valid. Must be set to 'kcfg'.
0x04 - 0x07	4	crcStartAddress	Start address for application image CRC check. To generate the CRC, refer to the CRC chapter. If the bits are all set then bootloader by default will not perform any CRC check.
0x08 - 0x0B	4	crcByteCount	Byte count for application image CRC check. If the bits are all set then bootloader by default will not perform any CRC check.
0x0C - 0x0F	4	crcExpectedValue	Expected CRC value for application CRC check. If the bits are all set then bootloader by default will not perform any CRC check.
0x10	1	enabledPeripherals	Bitfield of peripherals to enable. bit 0 UART bit 1 LPI2C bootloader will enable the peripheral if corresponding bit is set to 1.
0x11	1	i2cSlaveAddress	If not 0xFF, used as the 7-bit I2C slave address. If 0xFF, defaults to 0x10 for I2C slave address
0x12 - 0x13	2	peripheralDetectionTimeout	Timeout in milliseconds for active peripheral detection. If 0xFFFF, defaults to 5 seconds.
0x14 - 0x15	2	-	Reserved
0x16- 0x17	2	-	Reserved
0x18 - 0x1B	4	-	Reserved
0x1C	1	Reserved	-
0x1D	1	Reserved	-
0x1E	1	bootFlags	One's complement of direct boot flag. 0xFE represents direct boot.

Table continues on the next page...

Table 5-3. Configuration Fields for the Bootloader (continued)

Offset	Size (bytes)	Configuration Field	Description
0x1F	1	Reserved	-
0x20 - 0x23	4	Reserved	-
0x24 - 0x27	4	Reserved	-
0x28	1	Reserved	-
0x29 - 0x2F	7	Reserved	-
0x30 - 0x33	4	Reserved	-
0x34 - 0x3F	12	Reserved	-

NOTE

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

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

Table 5-4. tag Configuration Field

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

5.3.2 Start-up Process

The following condition will force the hardware to start the Bootloader:

- Any type of reset.
- A user applications running on flash or RAM calls into the Bootloader entry point address in ROM, through the bootloader tree located starting from 0x78100.

The FOPT register determines the power mode of boot. The FOPT register is loaded from the flash configuration field at address 0x40D in flash memory. For more details, see the "FOPT Register" section for the Flash Memory Module configuration.

When the ROM is executed out of reset, vector fetches from the CPU are redirected to the ROM's vector table in ROM memory at offset 0x07_8000. This ensures that any exceptions will be handled by the ROM.

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

1. Initializes the bootloader's .data and .bss sections.
2. Reads bootloader configuration data from flash at address 0x3C0 (Byte address). The configuration data is only used if the tag field is set to the expected 'kcfg' value. If the tag is incorrect, then the configuration values are set to default, as if the data was all 0xFF bytes.
3. ROM runs at internal 50 MHz clock.
4. Enabled peripherals are initialized.
5. After coming out of reset, the ROM bootloader initializes all peripherals, looks to see if a valid application is in flash (see Verifying the application in flash using CRC-32 for more details), then auto-detects any peripheral activities within a peripheral detect timeout period.

   ```
   #define BL_DEFAULT_PERIPHERAL_DETECT_TIMEOUT 5000   // 5s
   ```

   Before the BL_DEFAULT_PERIPHERAL_DETECT_TIMEOUT expires and even if a valid application exists in flash, the ROM bootloader still runs the automatic peripheral detection. After the BL_DEFAULT_PERIPHERAL_DETECT_TIMEOUT time period expires:
   - If no active peripherals are detected but there is a valid application in flash, then the ROM bootloader jumps to the application. [2]
   - If no active peripherals are detected and there is no valid application in flash, then the ROM bootloader continues to auto-detect peripherals activities.

2. ROM assumes the default application is at flash 0x0000. This is initial ROM starts. For user application which is on address (flash or RAM) other than 0x0000 flash address, user can use EXECUTE command to specify jump address and start running user application. See EXECUTE command for more details.

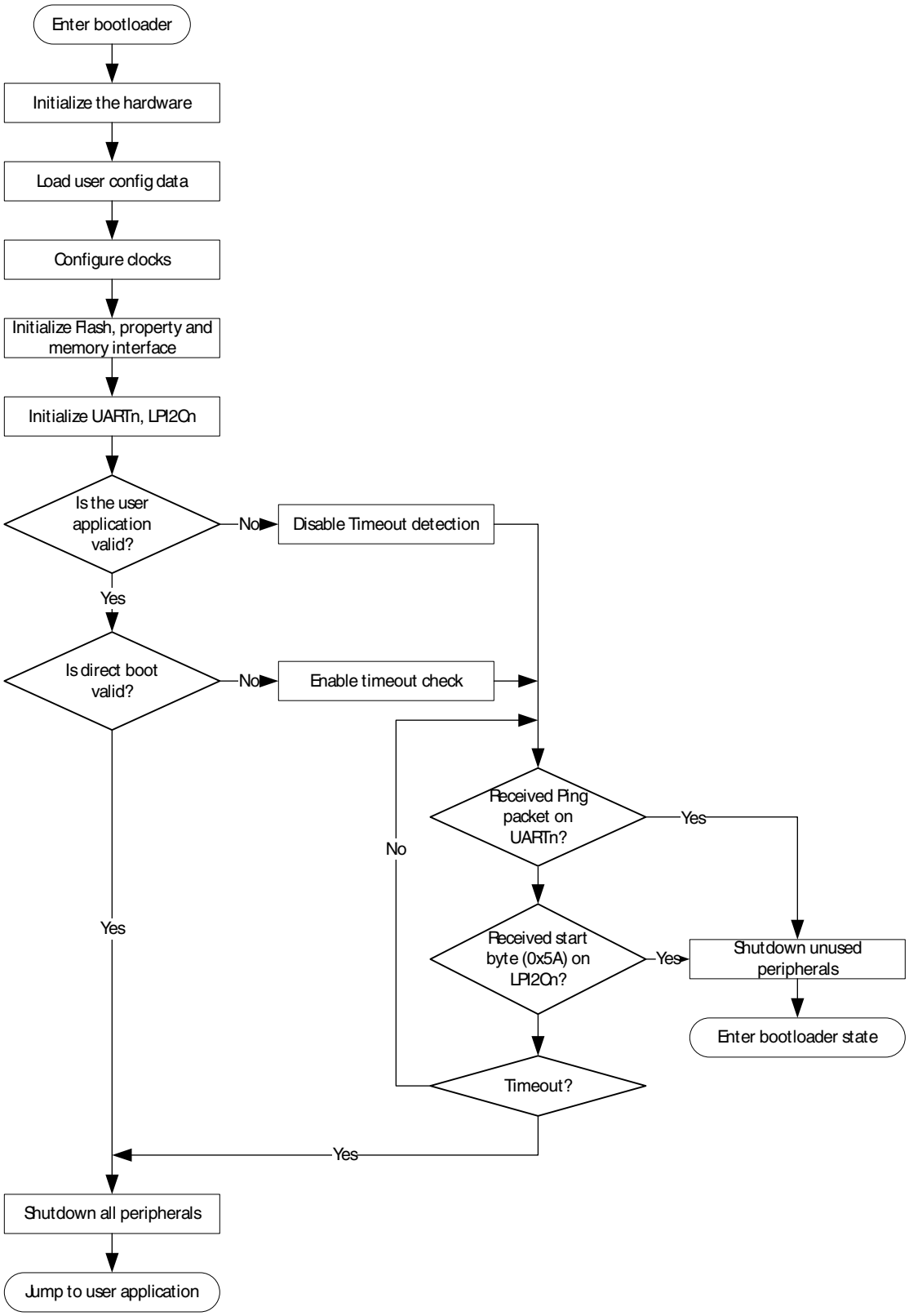


Figure 5-1. Bootloader Start-up Flowchart

5.3.3 Bootloader Entry Point / API Tree

To run the Bootloader, a user application simply calls the runBootloader function. To get the address of the entry point, the user application reads the word containing the pointer to the bootloader API tree at ROM 0x78100.

The bootloader API tree is a structure that contains pointers to other structures, which have the function and data addresses for the bootloader. The bootloader entry point is always the 1st word of the API tree.

```
typedef struct bootloader_api_tree
{
    volatile void (*runBootloader)(void *arg);            //!< Function to start the
bootloader executing.
    standard_version_t version; //!< flash driver API version number.//!< flash driver API
version number.
    volatile flash_driver_interface_t falsh_api;
} bootloader_api_tree_t;
```

The prototype of the entry point is:

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

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

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

NOTE

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

```
#define BOOTLOADER_TREE_LOCATION (0x78100UL)
#define API_TREE                 ((bootloader_api_tree_t*)BOOTLOADER_TREE_LOCATION)

API_TREE->runBootloader(0);
```

5.3.4 Bootloader Protocol

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

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

Commands may include an optional data phase:

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

NOTE

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

5.3.4.1 Command with no data phase

The protocol for a command with no data phase contains:

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

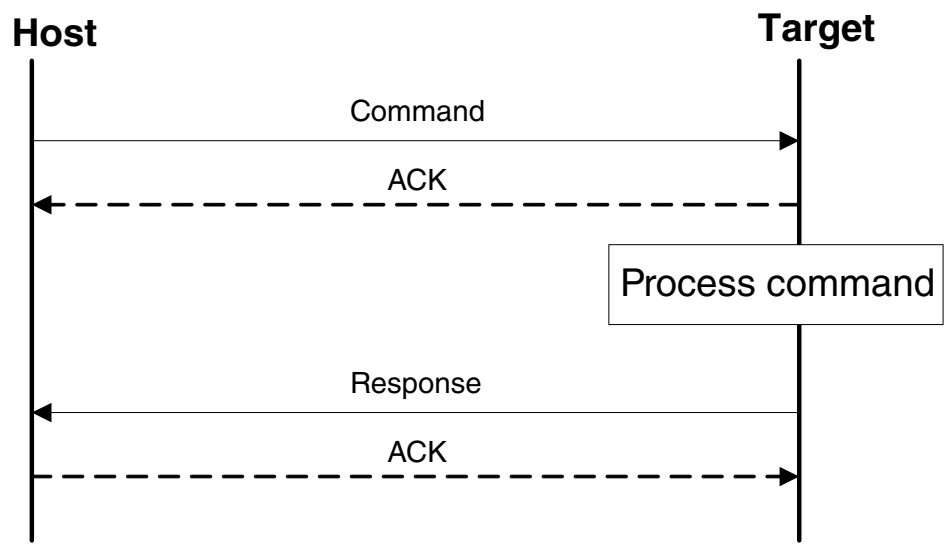


Figure 5-2. Command with No Data Phase

5.3.4.2 Command with incoming data phase

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

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

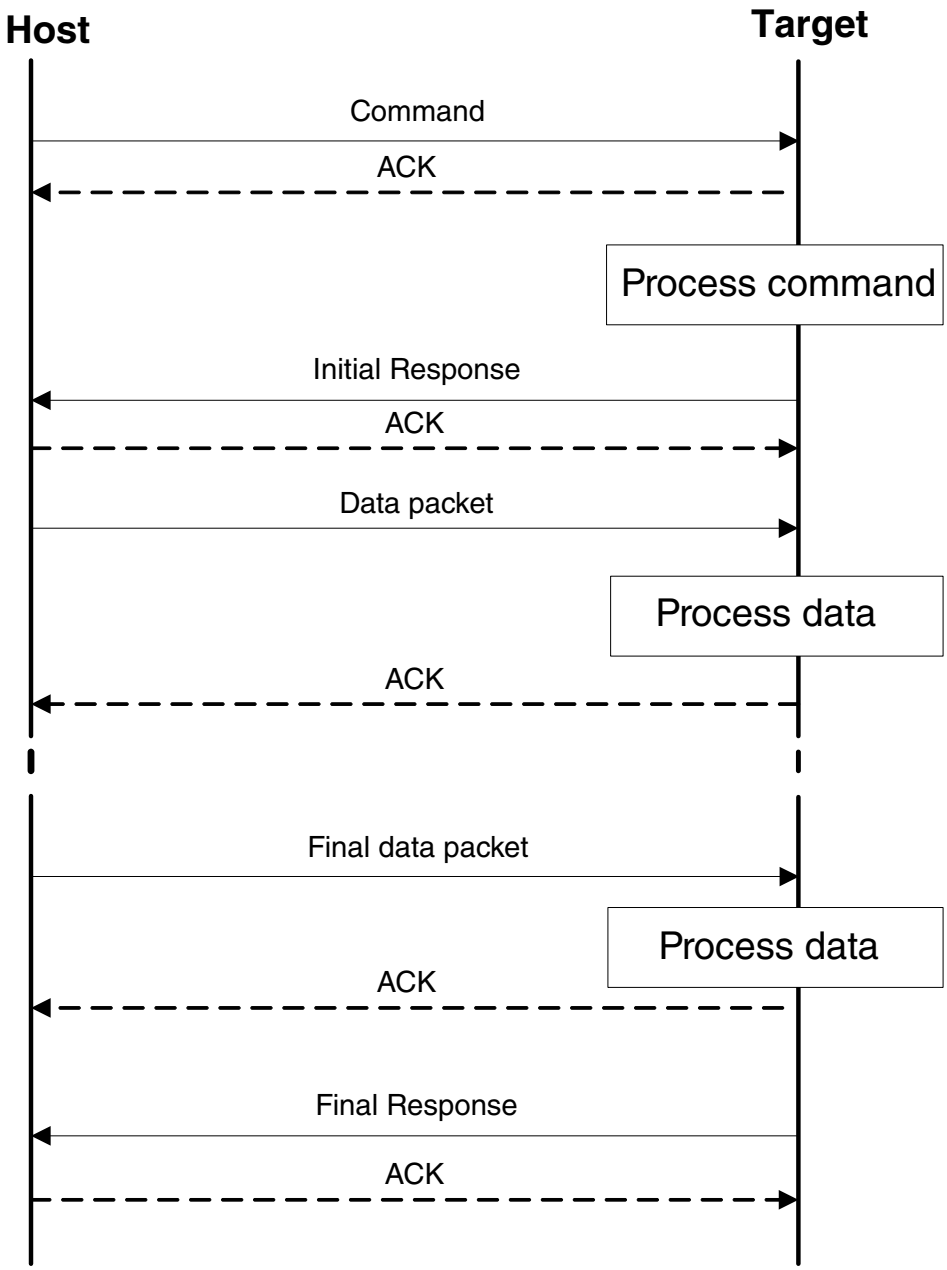


Figure 5-3. Command with incoming data phase

NOTE

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

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

5.3.4.3 Command with outgoing data phase

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

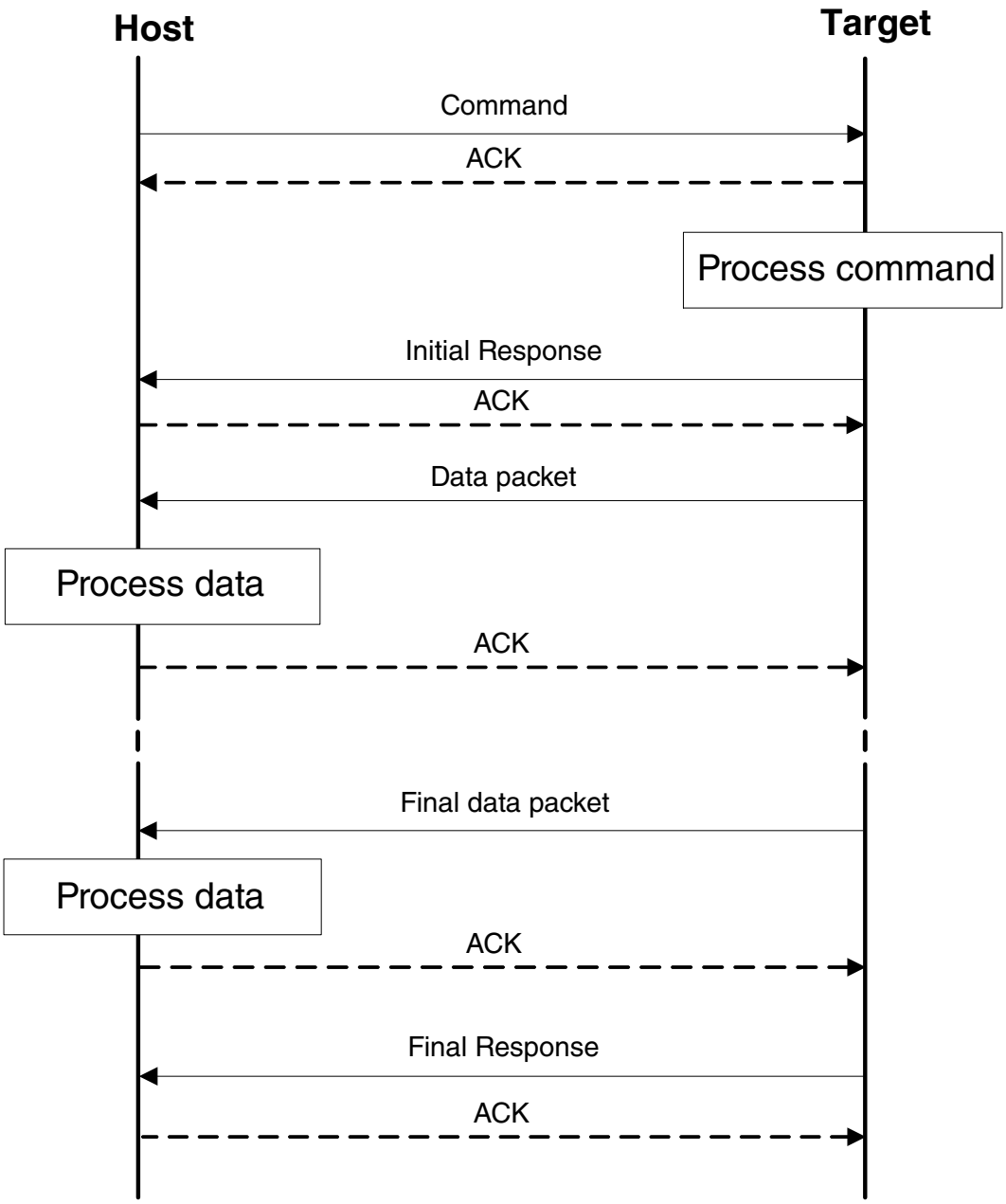


Figure 5-4. Command with outgoing data phase

NOTE

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

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

5.3.5 Bootloader Packet Types

The Bootloader device works in slave mode. All data communication is initiated by a host, which is either a PC or an embedded host . The Bootloader device is the target, which receives a command or data packet. All data communication between host and target is packetized.

NOTE

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

There are 6 types of packets used in the device:

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

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

5.3.5.1 Ping packet

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

Table 5-5. Ping Packet Format

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

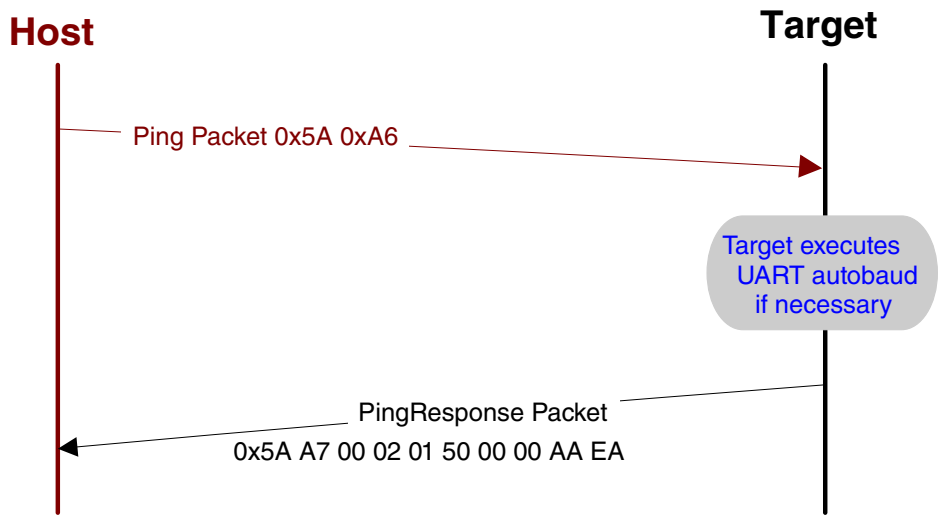


Figure 5-5. Ping Packet Protocol Sequence

5.3.5.2 Ping Response Packet

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

Table 5-6. Ping Response Packet Format

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

5.3.5.3 Framing Packet

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

The framing packet described in this section is used for serial peripherals including UART, I2C, .

Table 5-7. Framing Packet Format

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

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

Table 5-8. Special Framing Packet Format

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

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

Table 5-9. packetType Field

packetType	Name	Description
0xA1	kFramingPacketType_Ack	The previous packet was received successfully; the sending of more packets is allowed.
0xA2	kFramingPacketType_Nak	The previous packet was corrupted and must be re-sent.
0xA3	kFramingPacketType_AckAbort	Data phase is being aborted.
0xA4	kFramingPacketType_Command	The framing packet contains a command packet payload.
0xA5	kFramingPacketType_Data	The framing packet contains a data packet payload.
0xA6	kFramingPacketType_Ping	Sent to verify the other side is alive. Also used for UART autobaud.

Table continues on the next page...

Table 5-9. packetType Field (continued)

packetType	Name	Description
0xA7	kFramingPacketType_PingResponse	A response to Ping; contains the framing protocol version number and options.

CRC16 algorithm:

```
uint16_t crc16_update(const uint8_t * src, uint32_t lengthInBytes)
{
    uint32_t crc = 0;
    uint32_t j;
    for (j=0; j < lengthInBytes; ++j)
    {
        uint32_t i;
        uint32_t byte = src[j];
        crc ^= byte << 8;
        for (i = 0; i < 8; ++i)
        {
            uint32_t temp = crc << 1;
            if (crc & 0x8000)
            {
                temp ^= 0x1021;
            }
            crc = temp;
        }
    }
    return crc;
}
```

5.3.5.4 Command packet

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

Table 5-10. Command Packet Format

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

Table 5-11. Command Header Format

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

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

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

Table 5-12. Commands that are supported

Command	Name
0x01	FlashEraseAll
0x02	FlashEraseRegion
0x03	ReadMemory
0x04	WriteMemory
0x05	Reserved
0x06	FlashSecurityDisable
0x07	GetProperty
0x08	Reserved
0x09	Execute
0x0A	Reserved
0x0B	Reset
0x0C	Reserved
0x0D	FlashEraseAllUnsecure

Table continues on the next page...

Table 5-12. Commands that are supported (continued)

Command	Name
0x0E	FlashProgramOnce
0x0F	FlashReadOnce
0x10	FlashReadResource
0x11	Reserved
0x12	Reserved

Table 5-13. Responses that are supported

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

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

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

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

5.3.5.5 Data packet

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

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

5.3.5.6 Response packet

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

- GenericResponse
- GetPropertyResponse
- ReadMemoryResponse
- FlashReadOnceResponse
- FlashReadResourceResponse

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

Table 5-14. GenericResponse Parameters

Byte #	Parameter	Descripton
0 - 3	Status code	The Status codes are errors encountered during the execution of a command by the target (Bootloader). If a command succeeds, then a kStatus_Success code is returned. Table 5-54, Bootloader Status Error Codes, lists the status codes returned to the host by the Bootloader for ROM.
4 - 7	Command tag	The Command tag parameter identifies the response to the command sent by the host.

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

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

Table 5-15. GetPropertyResponse Parameters

Byte #	Value	Parameter
0 - 3		Status code
4 - 7		Property value

Table continues on the next page...

Table 5-15. GetPropertyResponse Parameters (continued)

Byte #	Value	Parameter
. . .		. . .
		Can be up to maximum 6 property values, limited to the size of the 32-bit command packet and property type.

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

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

Table 5-16. ReadMemoryResponse Parameters

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

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

Table 5-17. FlashReadOnceResponse Parameters

Byte #	Value	Parameter
0 – 3		Status Code
4 – 7		Byte count to read
…		…
		Can be up to 20 bytes of requested read data.

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

Table 5-18. FlashReadResourceResponse Parameters

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

5.3.6 Bootloader Command API

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

- For a list of commands supported by the Bootloader in ROM, see Table 5-2, Commands supported.
- For a list of status codes returned by the Bootloader in ROM, see Table 5-54, Bootloader Status Error Codes.

NOTE

All the examples in this section depict byte traffic on serial peripherals that use framing packets.

5.3.6.1 Execute command

The execute command results in the bootloader setting the program counter to the code at the provided jump address. Prior to the jump, the system is returned to the reset state.

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

Table 5-19. Parameters for Execute Command

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

The Execute command has no data phase.

Response: Before executing the Execute command, the target (Bootloader) will validate the parameters and run CRC check (if CRC parameters are not zero in the image), then return a GenericResponse packet with a status code either set to kStatus_Success or an appropriate error status code.

5.3.6.2 Reset command

The Reset command will result in bootloader resetting the chip.

The Reset command requires no parameters.

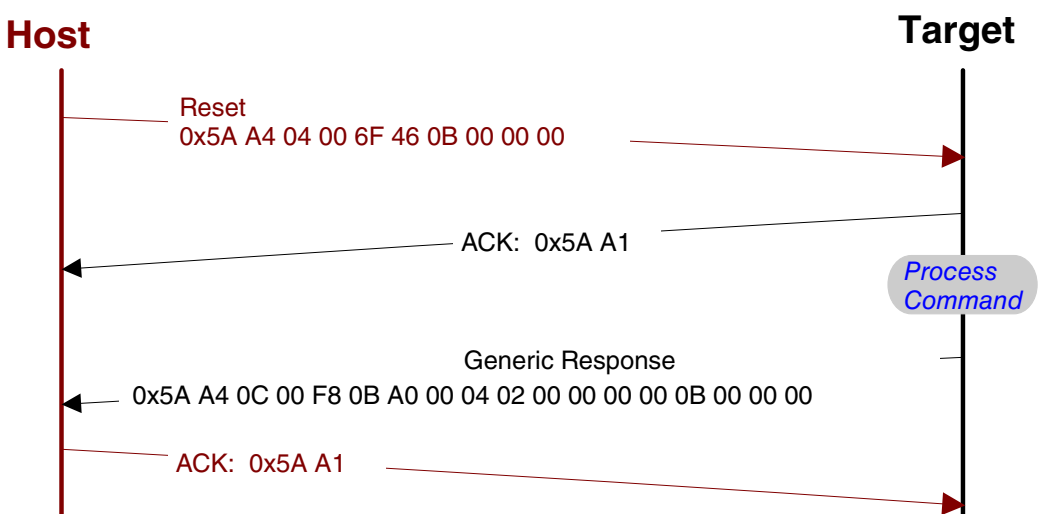


Figure 5-6. Protocol Sequence for Reset Command

Table 5-20. Reset Command Packet Format (Example)

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

The Reset command has no data phase.

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

5.3.6.3 GetProperty command

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

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

For a list of properties and their associated 32-bit property tags supported by the Bootloader in ROM, see Table 5-50.

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

Table 5-21. Parameters for GetProperty Command

Byte #	Command
0 - 3	Property tag

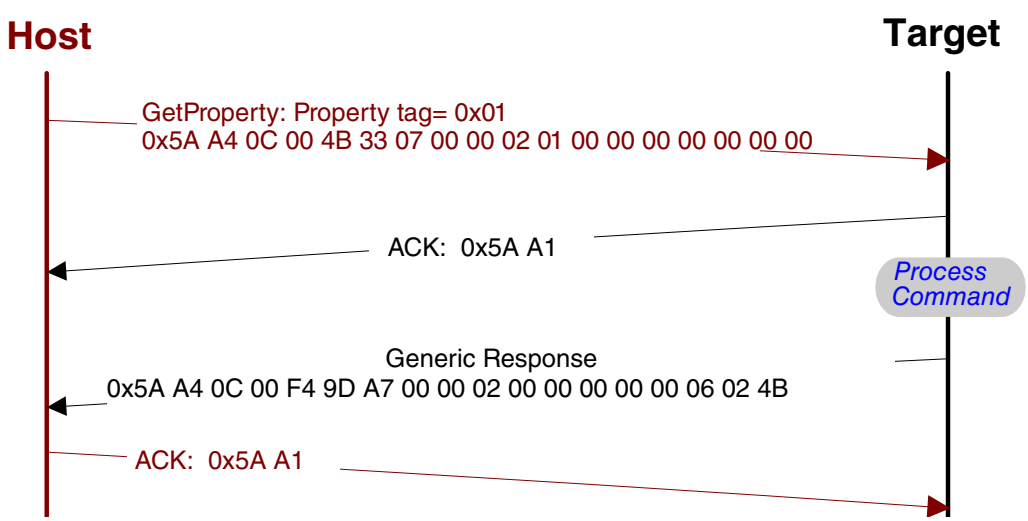


Figure 5-7. Protocol Sequence for GetProperty Command

Table 5-22. GetProperty Command Packet Format (Example)

GetProperty	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0x4B 0x33
Command packet	commandTag	0x07 – GetProperty
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	propertyTag	0x00000001 - CurrentVersion

The GetProperty command has no data phase.

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

Table 5-23. GetProperty Response Packet Format (Example)

GetPropertyResponse	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0C 0x00 (12 bytes)
	crc16	0xF4 0x9D
Command packet	responseTag	0xA7
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	status	0x00000000
	propertyValue	0x0006024B - CurrentVersion

5.3.6.4 FlashEraseAll command

The FlashEraseAll command performs an erase of the entire flash memory. If any flash regions are protected, then the FlashEraseAll command will fail and return an error status code. Executing the FlashEraseAll command will release flash security if it (flash security) was enabled, by setting the FTFA_FSEC register. However, the FSEC field of

the flash configuration field is erased, so unless it is reprogrammed, the flash security will be re-enabled after the next system reset. The Command tag for FlashEraseAll command is 0x01 set in the commandTag field of the command packet.

The FlashEraseAll command requires no parameters.

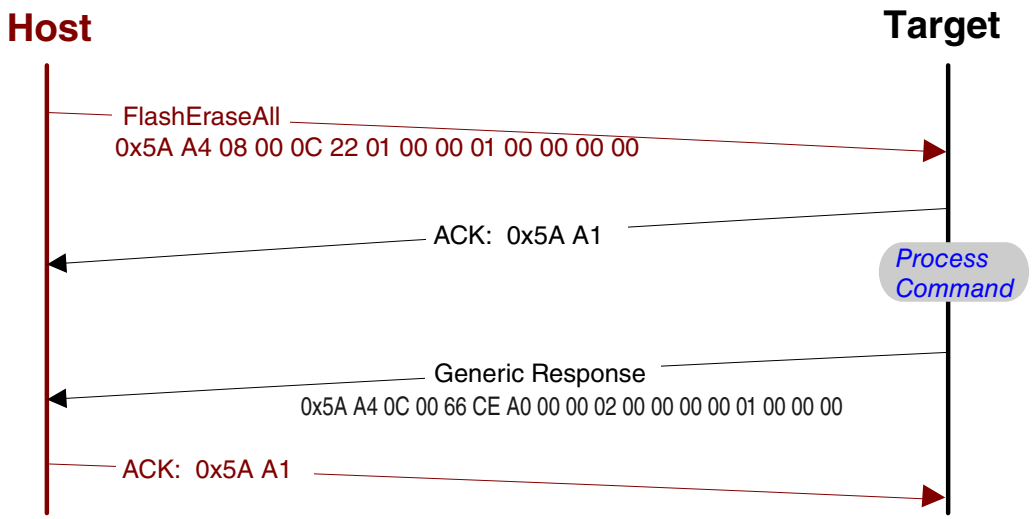


Figure 5-8. Protocol Sequence for FlashEraseAll Command

Table 5-24. FlashEraseAll Command Packet Format (Example)

FlashEraseAll	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x08 0x00
	crc16	• If MemoryID = 0 (internal flash), then crc16 = 0x0C 0x22 • If MemoryID = 1 (QSPI0 memory), then crc16 = 0xB8 0x54
Command packet	commandTag	0x01 - FlashEraseAll
	flags	0x00
	reserved	0x00
	parameterCount	0x01
	MemoryID	• If MemoryID = 0x00h, then internal flash. • If MemoryID = 0x01h, then QSPI0 memory. **NOTE:** 0x01h option (QSPI) **NOT** applicable in this device.

The FlashEraseAll command has no data phase.

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

Table 5-25. FlashEraseAll Response Status Codes

Status Code	Description
kStatus_FLASH_Success (0)	Executed successfully
kStatus_FLASH_InvalidArgument (4)	Invalid argument
kStatus_FLASH_EraseKeyError (107)	Erase key is invalid
kStatus_FLASH_AccessError (103)	Invalid instruction codes and out-of bound addresses
kStatus_FTFx_ProtectionViolation (104)	The program/erase operation is requested to execute on protected areas
kStatus_FLASH_CommandFailure (105)	Run-time error during the command execution

5.3.6.5 FlashEraseRegion command

The FlashEraseRegion command performs an erase of one or more sectors of the flash memory .

The start address and number of bytes are the 2 parameters required for the FlashEraseRegion command. The start and byte count parameters must be 4-byte aligned ([1:0] = 00), or the FlashEraseRegion command will fail and return kStatus_FlashAlignmentError (0x101). If the region specified does not fit in the flash memory space, the FlashEraseRegion command will fail and return kStatus_FlashAddressError (0x102). If any part of the region specified is protected, the FlashEraseRegion command will fail and return kStatus_MemoryRangeInvalid (0x10200).

Table 5-26. Parameters for FlashEraseRegion Command

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

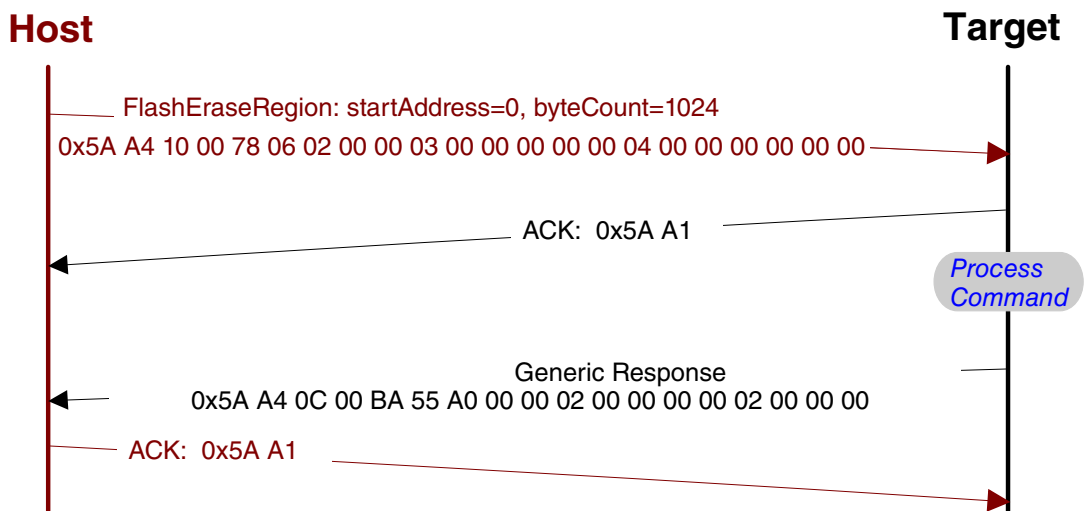


Figure 5-9. Protocol Sequence for FlashEraseRegion Command

Table 5-27. FlashEraseRegion Command Packet Format (Example)

FlashEraseRegion	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x10 0x00
	crc16	0x78 0x06
Command packet	commandTag	0x02, kCommandTag_FlashEraseRegion
	flags	0x00
	reserved	0x00
	parameterCount	0x03
	startAddress	0x00 0x00 0x00 0x00 (0x0000_0000)
	byte count	0x00 0x04 0x00 0x00 (0x400)
	memory_id	0x00 0x00 0x00 0x00 (internal flash)

The FlashEraseRegion command has no data phase.

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

Table 5-28. FlashEraseRegion Response Status Codes

Status Code
kStatus_Success (0x0)
kStatus_MemoryRangeInvalid (0x10200)
kStatus_FlashAlignmentError (0x101)
kStatus_FlashAddressError (0x102)
kStatus_FlashAccessError (0x103)

Table continues on the next page...

Table 5-28. FlashEraseRegion Response Status Codes (continued)

Status Code
kStatus_FlashProtectionViolation (0x104)
kStatus_FlashCommandFailure (0x105)

5.3.6.6 FlashEraseAllUnsecure command

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

The FlashEraseAllUnsecure command requires no parameters.

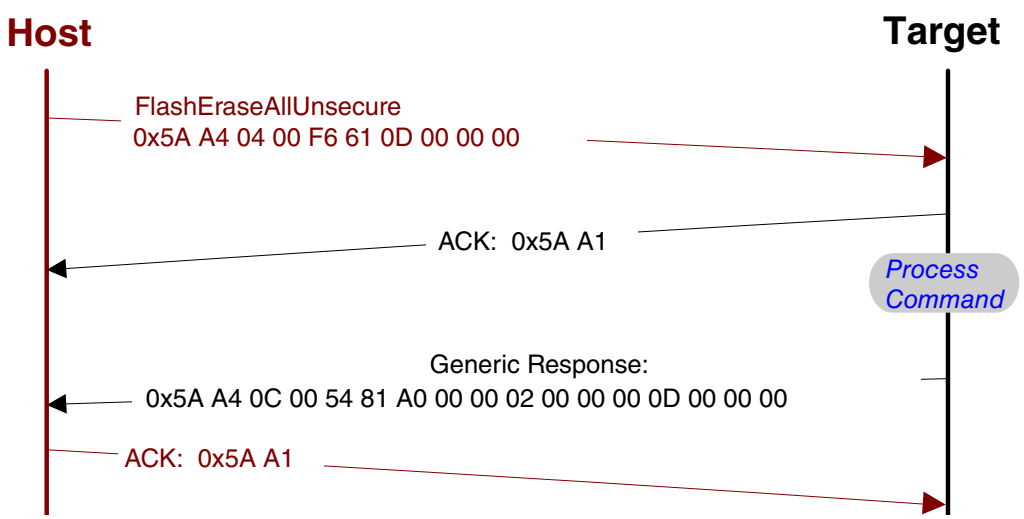


Figure 5-10. Protocol Sequence for FlashEraseAllUnsecure Command

Table 5-29. FlashEraseAllUnsecure Command Packet Format (Example)

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

Table continues on the next page...

Table 5-29. FlashEraseAllUnsecure Command Packet Format (Example) (continued)

FlashEraseAllUnsecure	Parameter	Value
	reserved	0x00
	parameterCount	0x00

The FlashEraseAllUnsecure command has no data phase.

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

Table 5-30. FlashEraseAllUnsecure Response Status Codes

Status Code	Description
kStatus_FLASH_Success (0)	Executed successfully
kStatus_FLASH_InvalidArgument (4)	Invalid argument
kStatus_FLASH_EraseKeyError (107)	Erase key is invalid
kStatus_FLASH_AccessError (103)	Invalid instruction codes and out-of bound addresses
kStatus_FTFx_ProtectionViolation (104)	The program/erase operation is requested to execute on protected areas
kStatus_FLASH_CommandFailure (105)	Run-time error during the command execution

5.3.6.7 FlashProgramOnce command

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

- The program once field only supports programming once, so any attempted to reprogram a program once field will get an error response.
- Writing to the program once field (index field is 0-7, each is 8 Bytes long).

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

Table 5-31. Parameters for FlashProgramOnce Command

Byte #	Command
0 - 3	Index of program once field
4 - 7	Byte count (8 Bytes long)
8 - 11	Data
12 - 16	Data

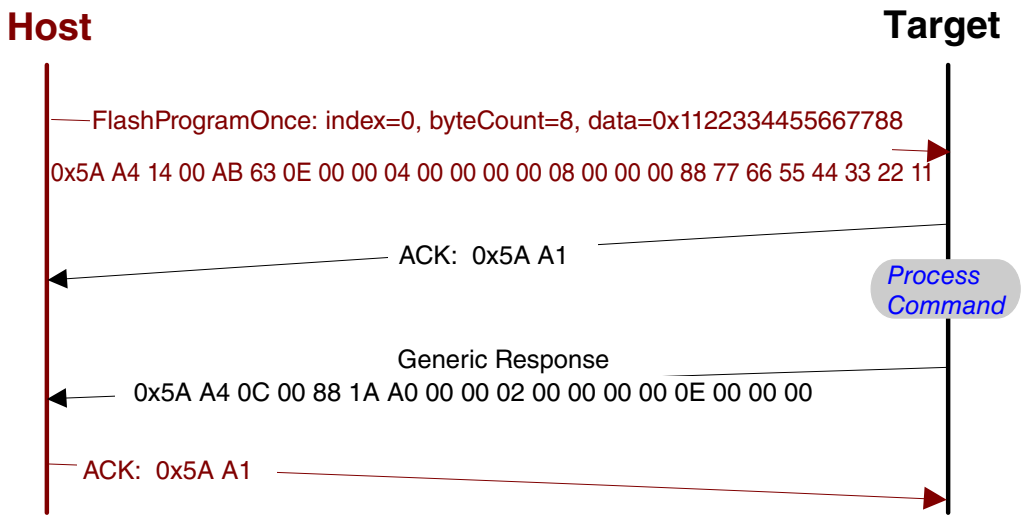


Figure 5-11. Protocol Sequence for FlashProgramOnce Command

Table 5-32. FlashProgramOnce Command Packet Format (Example)

FlashProgramOnce	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x14 0xx00
	crc16	0xAB 0x63
Command packet	commandTag	0x0E – FlashProgramOnce
	flags	0
	reserved	0
	parameterCount	4
	index	0x0000_0000
	byteCount	0x0000_0008
	data	0x1122_3344 5566_7788

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

Table 5-33. FlashProgramOnce Response Status Codes

Status Code	Description
kStatus_FLASH_Success (0)	Executed successfully
kStatus_FLASH_InvalidArgument (4)	Invalid argument
kStatus_FLASH_AccessError (103)	Invalid instruction codes and out-of bound addresses
kStatus_FTFx_ProtectionViolation (104)	The program/erase operation is requested to execute on protected areas
kStatus_FLASH_CommandFailure (105)	Run-time error during the command execution

5.3.6.8 FlashReadOnce command

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

Table 5-34. Parameters for FlashReadOnce Command

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

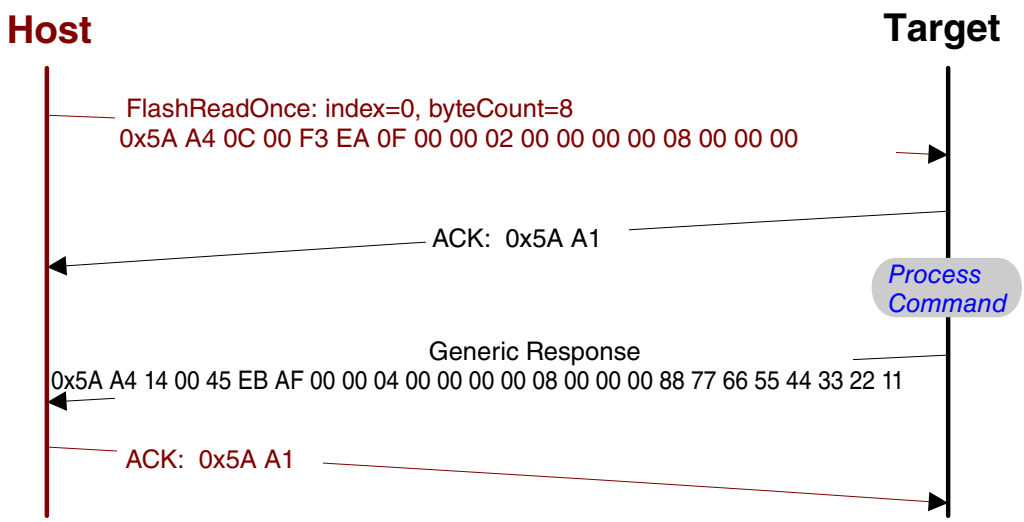


Figure 5-12. Protocol Sequence for FlashReadOnce Command

Table 5-35. FlashReadOnce Command Packet Format (Example)

FlashReadOnce	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4
	length	0x0C 0x00
	crc	0xF3 0xEA
Command packet	commandTag	0x0F – FlashReadOnce
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	index	0x0000_0000
	byteCount	0x0000_0008

Table 5-36. FlashReadOnce Response Format (Example)

FlashReadOnce Response	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4
	length	0x14 0x00
	crc	0x45 0xEB
Command packet	commandTag	0xAF
	flags	0x00
	reserved	0x00
	parameterCount	0x04
	status	0x0000_0000
	byteCount	0x0000_0008
	data	0x1122334455667788

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

Table 5-37. FlashReadOnce Response Status Codes

Status Code	Description
kStatus_FLASH_Success (0)	Executed successfully
kStatus_FLASH_InvalidArgument (4)	Invalid argument
kStatus_FLASH_AccessError (103)	Invalid instruction codes and out-of bound addresses
kStatus_FTFx_ProtectionViolation (104)	The program/erase operation is requested to execute on protected areas
kStatus_FLASH_CommandFailure (105)	Run-time error during the command execution

5.3.6.9 FlashReadResource command

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

Table 5-38. Parameters for FlashReadResource Command

Byte #	Parameter	Command
0 - 3	start address	Start address of specific non-volatile memory to be read

Table continues on the next page...

Table 5-38. Parameters for FlashReadResource Command (continued)

Byte #	Parameter	Command
4 - 7	byteCount	Byte count to be read
8 - 11	option	0: IFR 1: Flash Version ID

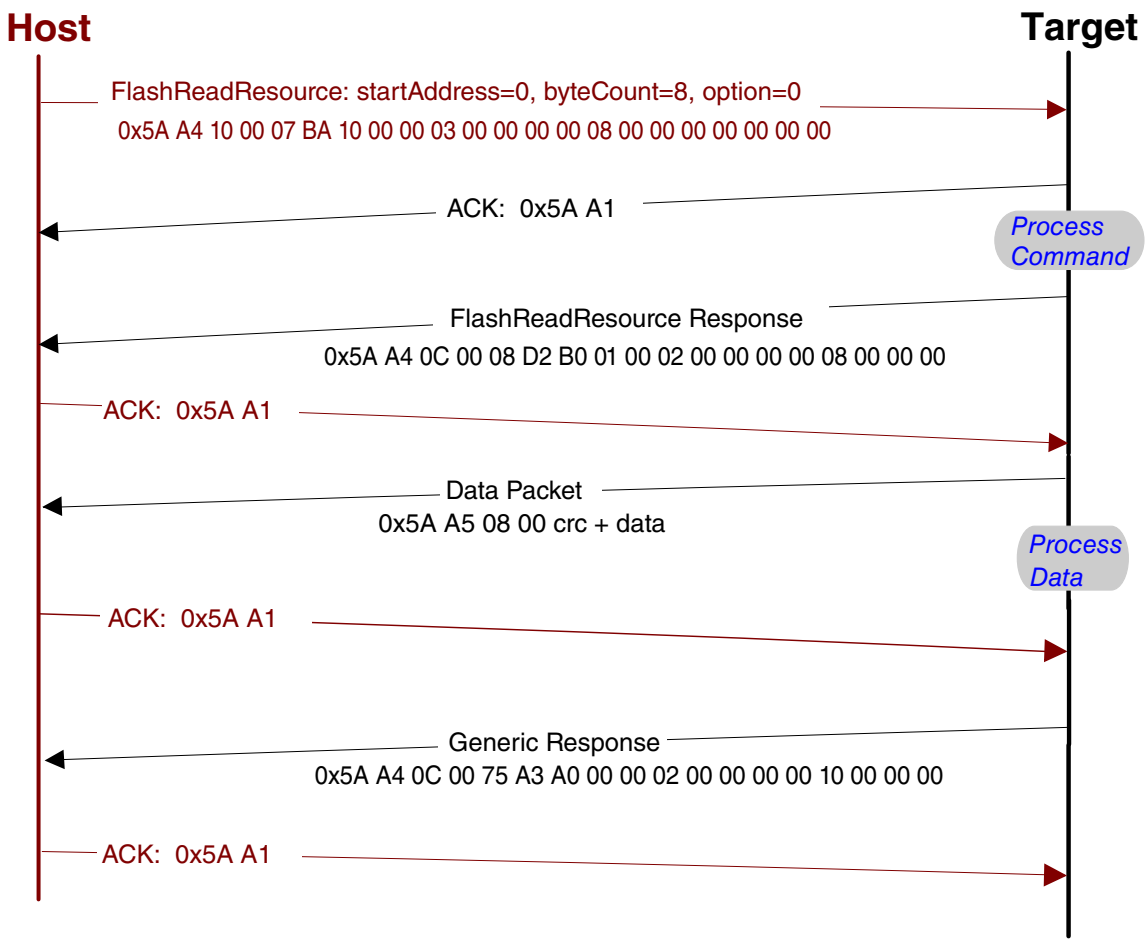


Figure 5-13. Protocol Sequence for FlashReadResource Command

Table 5-39. FlashReadResource Command Packet Format (Example)

FlashReadResource	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4
	length	0x10 0x00
	crc	0x07 0xBA
Command packet	commandTag	0x10 – FlashReadResource
	flags	0x00
	reserved	0x00

Table continues on the next page...

Table 5-39. FlashReadResource Command Packet Format (Example) (continued)

FlashReadResource	Parameter	Value
	parameterCount	0x03
	startAddress	0x0000_0000
	byteCount	0x0000_0008
	option	0x0000_0000

Table 5-40. FlashReadResource Response Format (Example)

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

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

Table 5-41. FlashReadResource Response Status Codes

Status Code	Description
kStatus_FLASH_Success (0)	Executed successfully
kStatus_FLASH_InvalidArgument (4)	Invalid argument
kStatus_FLASH_AlignmentError (101)	Parameter is not aligned with the specified baseline
kStatus_FLASH_AccessError (103)	Invalid instruction codes and out-of bound addresses
kStatus_FTFx_ProtectionViolation (104)	The program/erase operation is requested to execute on protected areas
kStatus_FLASH_CommandFailure (105)	Run-time error during the command execution

5.3.6.10 FlashSecurityDisable command

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

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

Table 5-42. Parameters for FlashSecurityDisable Command

Byte #	Command
0 - 3	Backdoor key low word
4 - 7	Backdoor key high word

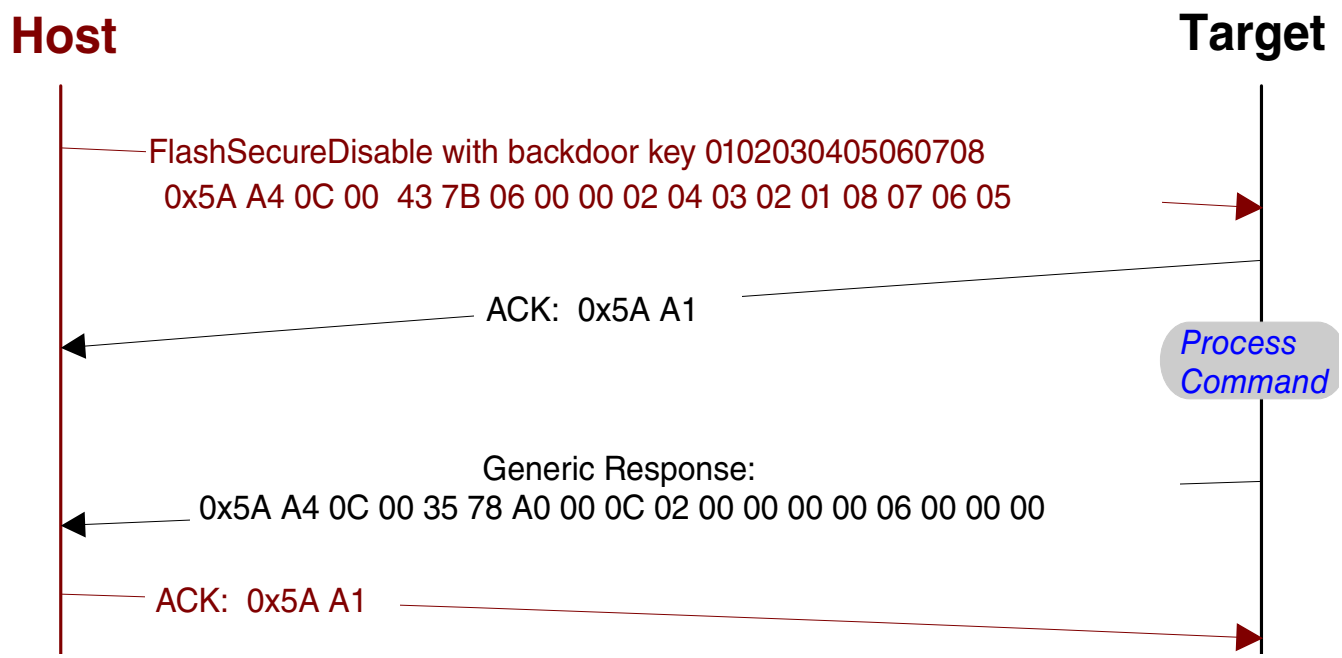


Figure 5-14. Protocol Sequence for FlashSecurityDisable Command

Table 5-43. FlashSecurityDisable Command Packet Format (Example)

FlashSecurityDisable	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0x43 0x7B
Command packet	commandTag	0x06 - FlashSecurityDisable
	flags	0x00

Table continues on the next page...

Table 5-43. FlashSecurityDisable Command Packet Format (Example) (continued)

FlashSecurityDisable	Parameter	Value
	reserved	0x00
	parameterCount	0x02
	Backdoorkey_low	0x04 0x03 0x02 0x01
	Backdoorkey_high	0x08 0x07 0x06 0x05

The FlashSecurityDisable command has no data phase.

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

5.3.6.11 WriteMemory command

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

Special care must be taken when writing to flash.

- First, any flash sector written to must have been previously erased with a FlashEraseAll, FlashEraseRegion, or FlashEraseAllUnsecure command.
- Writing to flash requires the start address to be 4-byte aligned ([1:0] = 00).
- The byte count will be rounded up to a multiple of 4, and the trailing bytes will be filled with the flash erase pattern (0xFF).
- If the VerifyWrites property is set to true, then writes to flash will also perform a flash verify program operation.

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

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

Table 5-44. Parameters for WriteMemory Command

Byte #	Command
0 - 3	Start address
4 - 7	Byte count
8 - 11	memory ID **NOTE:** The only option is 0x0 for this device.

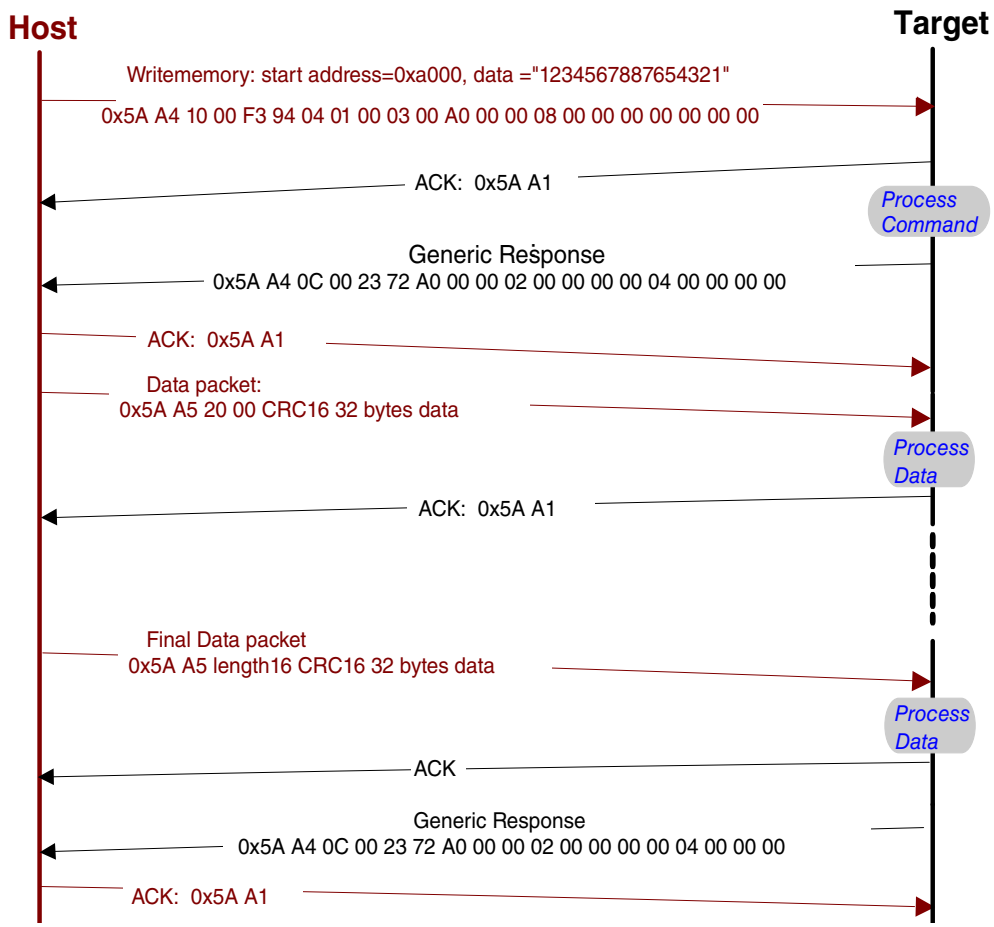


Figure 5-15. Protocol Sequence for WriteMemory Command

Table 5-45. WriteMemory Command Packet Format (Example)

WriteMemory	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0x06 0x5A
Command packet	commandTag	0x04 - writeMemory
	flags	0x01, *kCommandFlag_HasDataPhase*
	reserved	0x00
	parameterCount	0x03
	startAddress	0x0000A000
	byteCount	0x00000008
	memory ID	0x0

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

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

Table 5-46. WriteMemory Response Status Codes

Status Code	Description
kStatus_FLASH_Success (0)	Executed successfully
kStatus_OutOfRange (3)	Address is out of memory range
kStatusMemoryRangeInvalid (10200)	Memory range is invalid
kStatus_FLASH_InvalidArgument (4)	Invalid argument
kStatus_FLASH_AddressError (102)	Address is out of range
kStatus_FLASH_AccessError (103)	Invalid instruction codes and out-of bound addresses
kStatus_FTFx_ProtectionViolation (104)	The program/erase operation is requested to execute on protected areas
kStatus_FLASH_CommandFailure (105)	Run-time error during the command execution

5.3.6.12 ReadMemory command

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

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

Table 5-47. Parameters for ReadMemory command

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

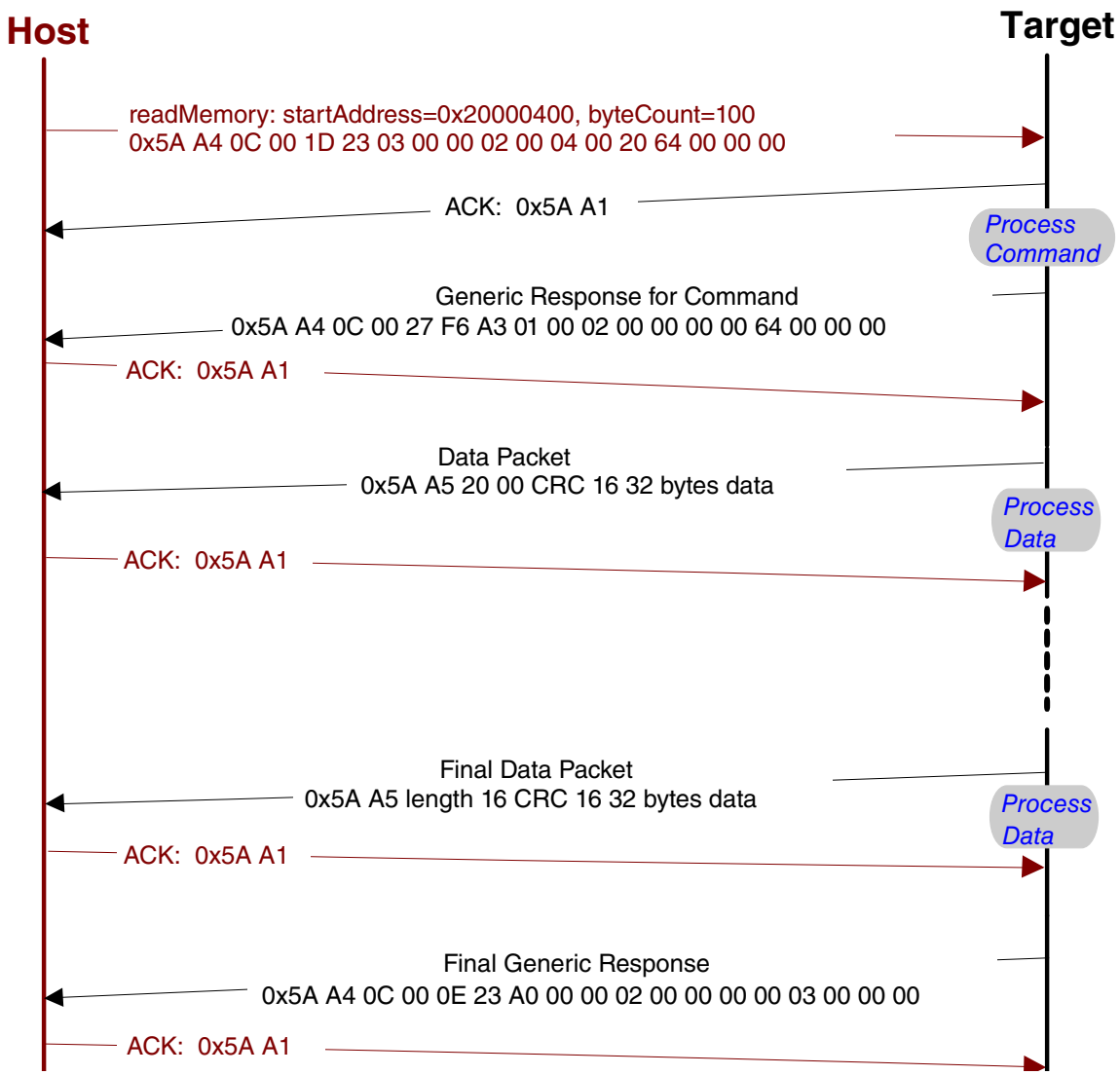


Figure 5-16. Command sequence for ReadMemory

Table 5-48. ReadMemory Command Packet Format (Example)

ReadMemory	Parameter	Value
Framing packet	Start byte	0x5A 0xA4
	packetType	kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0x1D 0x23
Command packet	commandTag	0x03 - readMemory
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	startAddress	0x20000400
	byteCount	0x00000064

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

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

Table 5-49. ReadMemory Response Status Codes

Status Code	Description
kStatus_FLASH_Success (0)	Executed successfully
kStatus_OutOfRange (3)	Address is out of memory range
kStatusMemoryRangeInvalid (10200)	Memory range is invalid

5.3.7 Bootloader Exit state

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

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

5.4 Peripherals Supported

This section describes the peripherals supported by the ROM Bootloader. To use an interface for bootloader communications, the peripheral must be enabled in the BCA, as shown in Table 5-3. If the BCA is invalid (such as all 0xFF bytes), then all peripherals will be enabled by default.

5.4.1 LPI2C Peripheral

The Bootloader in ROM supports loading data into flash via the LPI2C peripheral, where the LPI2C peripheral serves as the LPI2C slave. A 7-bit slave address is used during the transfer.

Customizing an LPI2C slave address is also supported. This feature is enabled if the Bootloader Configuration Area (BCA) (shown in Table 5-3) is enabled (tag field is filled with ‘kcfg’) and the i2cSlaveAddress field is filled with a value other than 0xFF. 0x10 is used as the default LPI2C slave address.

The maximum supported LPI2C baud rate depends on corresponding clock configuration field in the BCA. Typical supported baud rate is 400 kbps with factory settings (LPI2C baud rate not over 1.2 Mbps, for this device).

Because the LPI2C peripheral serves as an LPI2C slave device, each transfer should be started by the host, and each outgoing packet should be fetched by the host.

- An incoming packet is sent by the host with a selected LPI2C slave address and the direction bit is set as write.
- An outgoing packet is read by the host with a selected LPI2C slave address and the direction bit is set as read.
- 0x00 will be sent as the response to host if the target is busy with processing or preparing data.

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

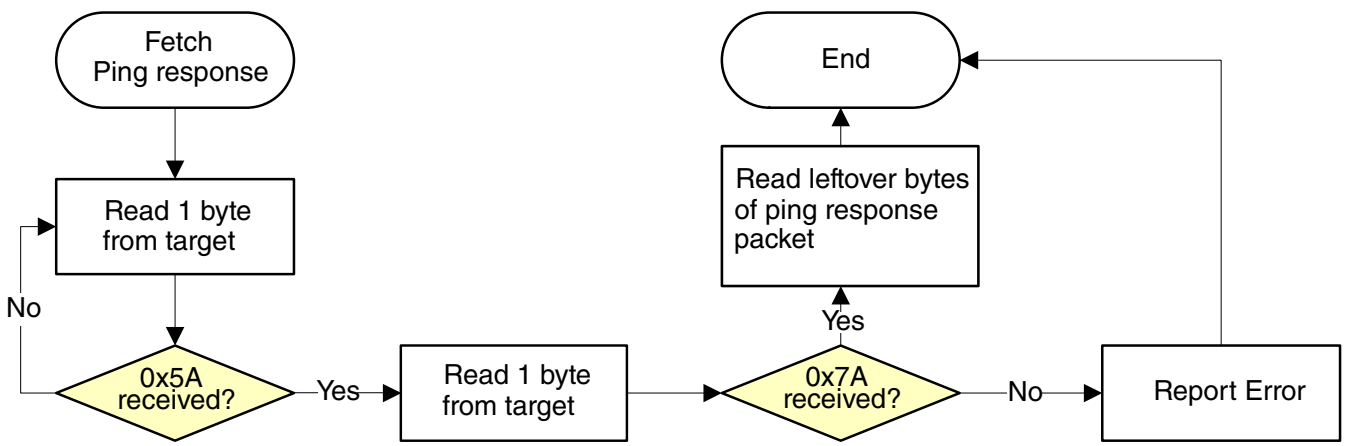


Figure 5-17. Host reads ping response from target via LPI2C

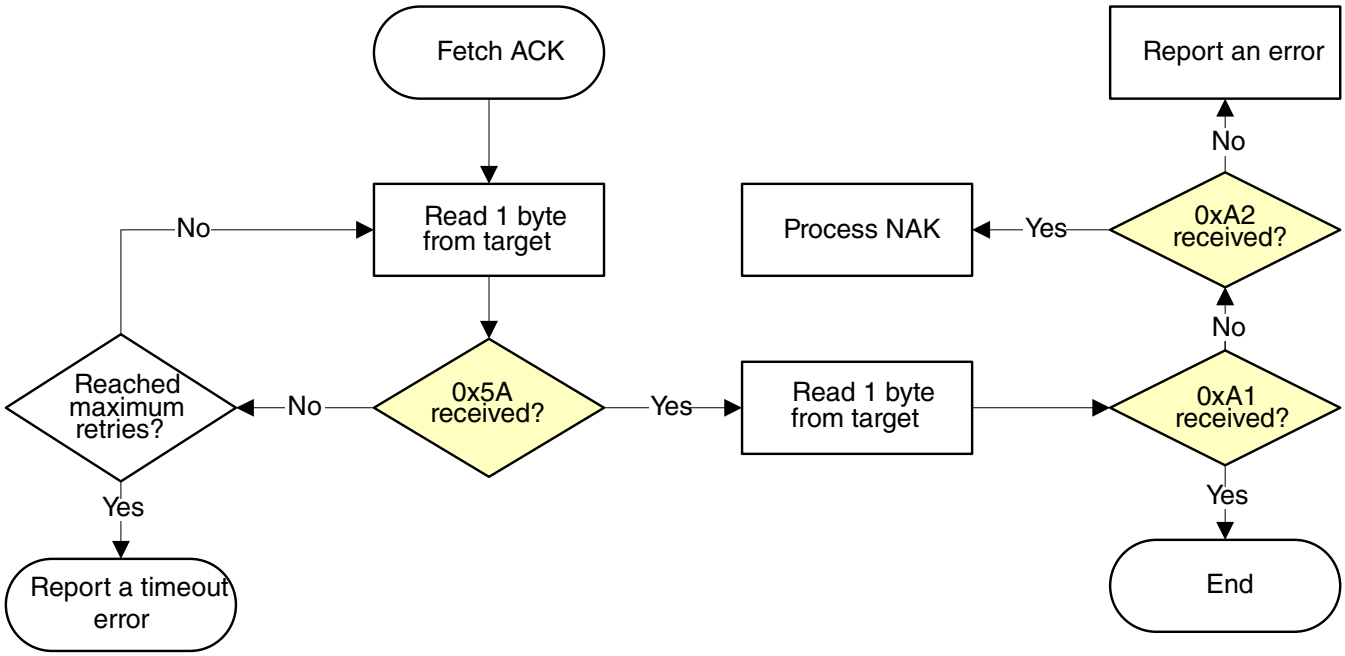


Figure 5-18. Host reads ACK packet from target via LPI2C

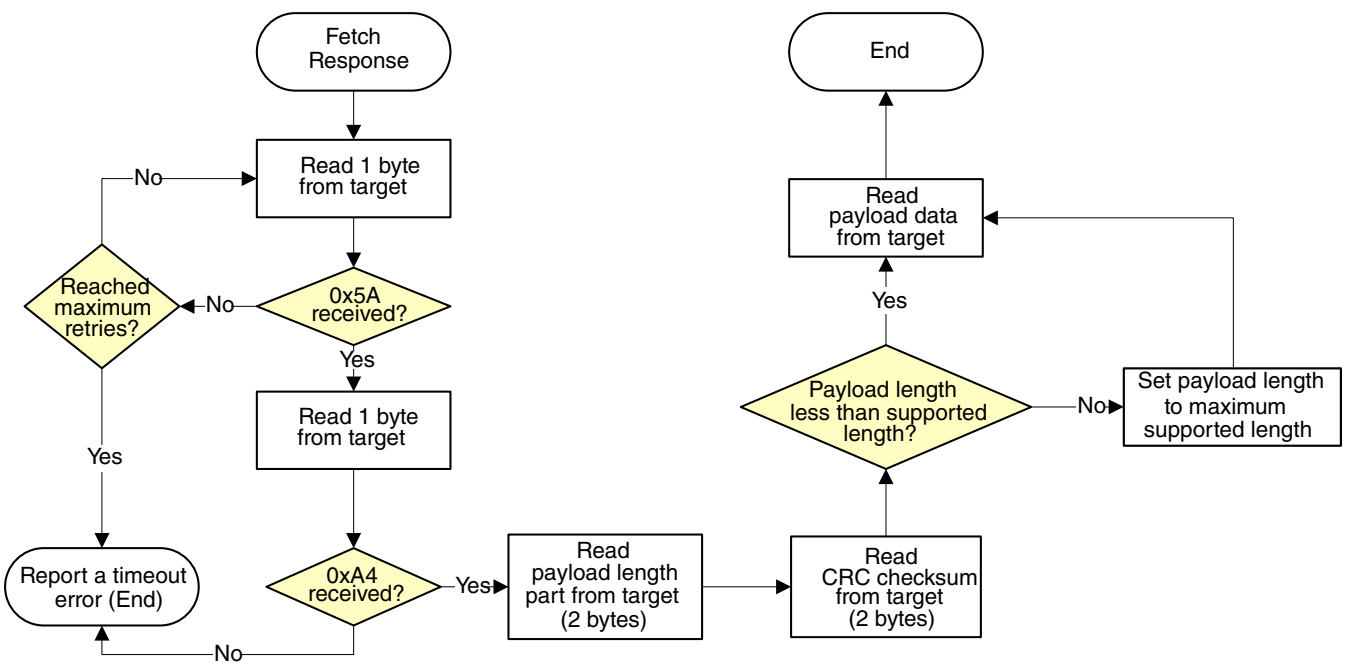


Figure 5-19. Host reads response from target via LPI2C

5.4.2 LPUART Peripheral

The Bootloader integrates an autobaud detection algorithm for the LPUART peripheral, thereby providing flexible baud rate choices.

Autobaud feature: If LPUART*n* is used to connect to the bootloader, then the LPUART*n*_RX pin must be kept high and not left floating during the detection phase in order to comply with the autobaud detection algorithm. After the bootloader detects the ping packet (0x5A 0xA6) on LPUART*n*_RX, the bootloader firmware executes the autobaud sequence. If the baudrate is successfully detected, then the bootloader will send a ping packet response [(0x5A 0xA7), protocol version (4 bytes), protocol version options (2 bytes) and crc16 (2 bytes)] at the detected baudrate. The Bootloader then enters a loop, waiting for bootloader commands via the LPUART peripheral.

NOTE

- The autobaud feature requires a ping packet with a higher accuracy (+/-3%), or the ping packet will be ignored as noise.
- The data bytes of the ping packet must be sent continuously (with no more than 80 ms between bytes) in a fixed LPUART transmission mode (8-bit data, no parity bit and 1 stop bit). If the bytes of the ping packet are sent one-by-one with more than 80 ms delay between them, then the autobaud detection algorithm may calculate an incorrect baud rate. In this case, the autobaud detection state machine should be reset.

Supported baud rates: The baud rate is closely related to the MCU core and system clock frequencies. Typical baud rates supported are 9600, 19200, 38400, 57600, and 115200.

Packet transfer: After autobaud detection succeeds, bootloader communications can take place over the LPUART peripheral. The following flow charts show:

- How the host detects an ACK from the target
- How the host detects a ping response from the target
- How the host detects a command response from the target

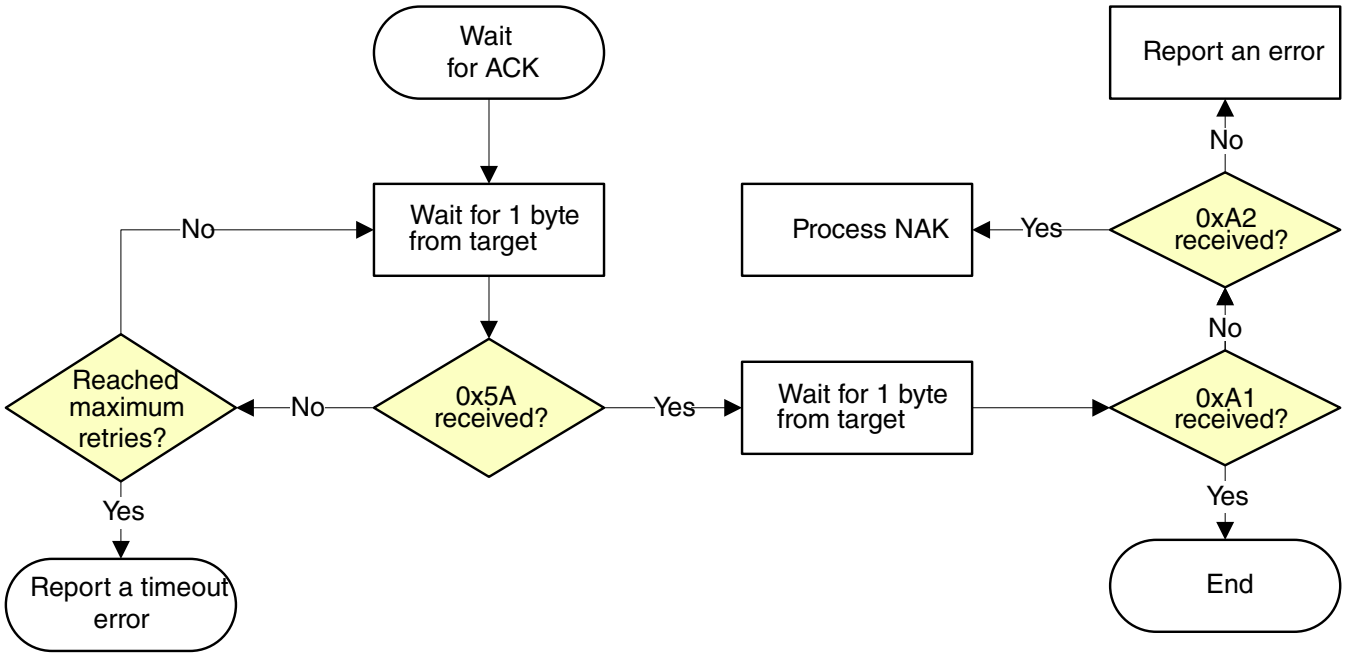


Figure 5-20. Host reads an ACK from target via LPUART

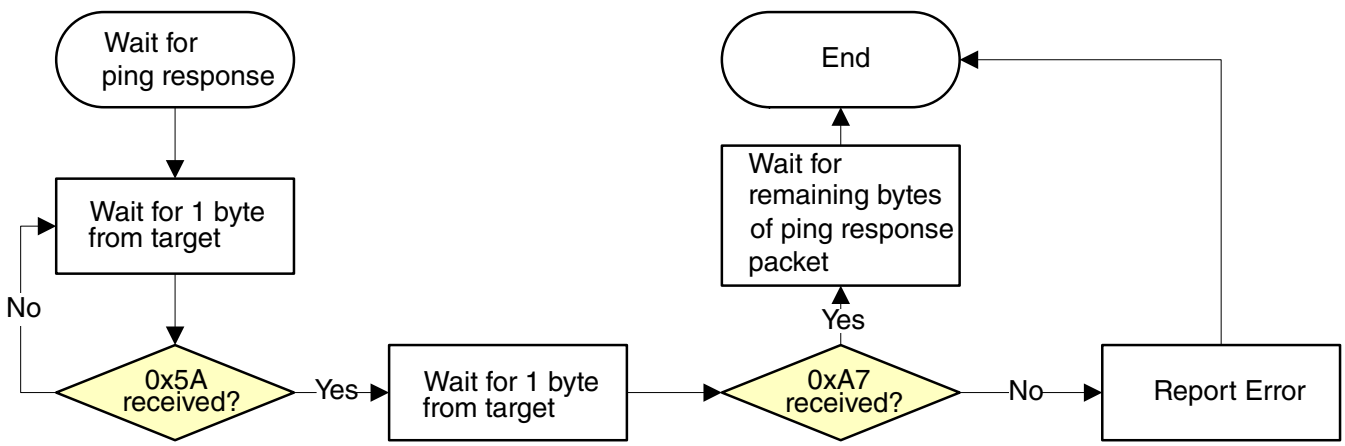


Figure 5-21. Host reads a ping response from target via LPUART

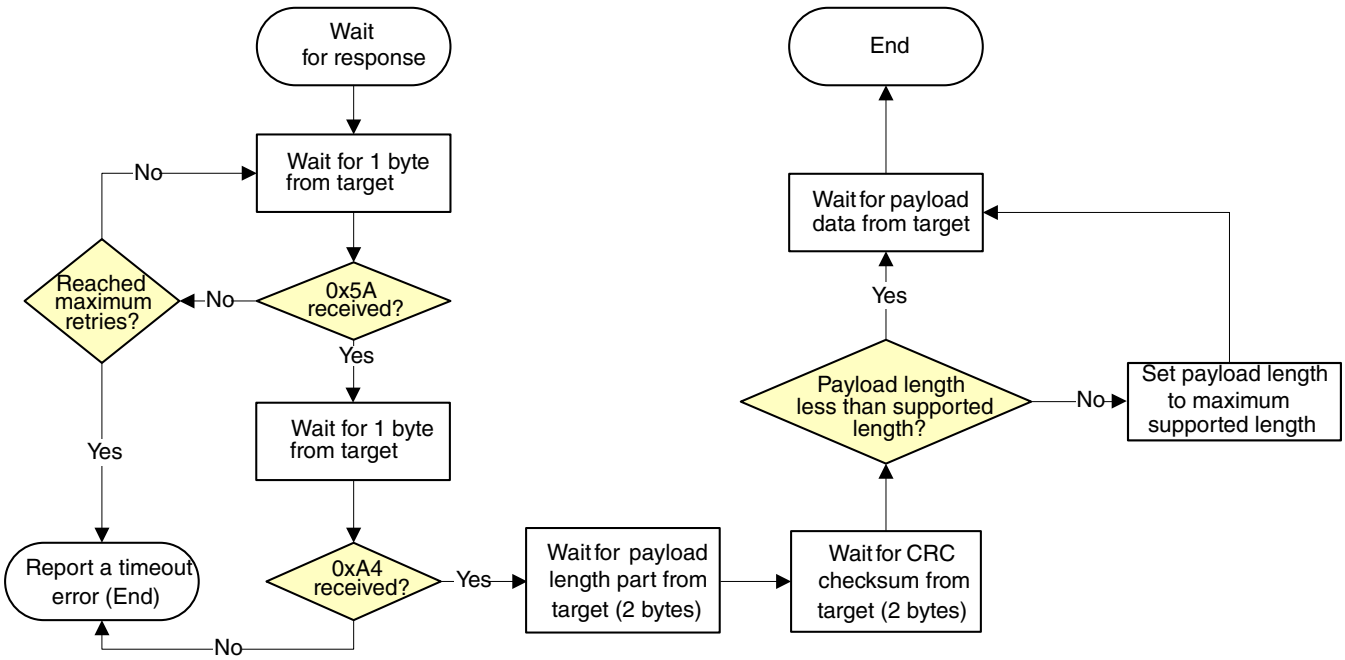


Figure 5-22. Host reads a command response from target via LPUART

5.5 GetProperty Command Properties

This section lists the properties of the GetProperty commands.

Table 5-50. Properties used by GetProperty Commands, sorted by Value

Property	Writable	Tag Value	Size	Description
CurrentVersion	No	01h	4	Current bootloader version.
AvailablePeripherals	No	02h	4	The set of peripherals supported on this chip.
FlashStartAddress	No	03h	4	Start address of program flash.
FlashSizeInBytes	No	04h	4	Size in bytes of program flash.
FlashSectorSize	No	05h	4	The size in bytes of one sector of program flash. This is the minimum erase size.
FlashBlockCount	No	06h	4	Number of blocks in the flash array.
AvailableCommands	No	07h	4	The set of commands supported by the bootloader.
CRCCheckStatus	No	08h	4	The status of the application CRC check.
VerifyWrites	Yes	0Ah	4	Controls whether the bootloader will verify writes to flash. VerifyWrites feature is enabled by default. 0 - No verification is done. 1 - Enable verification.
MaxPacketSize	No	0Bh	4	Maximum supported packet size for the currently active peripheral interface.

Table continues on the next page...

Table 5-50. Properties used by GetProperty Commands, sorted by Value (continued)

Property	Writable	Tag Value	Size	Description
ReservedRegions	No	0Ch	16	List of memory regions reserved by the bootloader. Returned as value pairs (<start-address-of-region>, <end-address-of-region>). • If HasDataPhase flag is not set, then the Response packet parameter count indicates the number of pairs. • If HasDataPhase flag is set, then the second parameter is the number of bytes in the data phase.
RAMStartAddress	No	0Eh	4	Start address of RAM segment. The first parameter to GetProperty command identifies the segment. See the device specific memory map for number of RAM segments the device contains.
RAMSizeInBytes	No	0Fh	4	Size in bytes of RAM segment. The first parameter to GetProperty command identifies the segment. See the device specific memory map for number of RAM segments the device contains.
SystemDeviceId	No	10h	4	Value of the Kinetis System Device Identification register.
FlashSecurityState	No	11h	4	Indicates whether Flash security is enabled 0 - Flash security is disabled 1 - Flash security is enabled
FacSupport	No	13h	4	FAC (Flash Access Control) support flag 0 - FAC not supported 1 - FAC supported
FlashAcessSegmentSize	No	14h	4	The size in bytes of 1 segment of flash
FlashAcessSegmentCount	No	15h	4	FAC segment count (The count of flash access segments within the flash model.)
FlashReadMargin	Yes	16h	4	The margin level setting for flash erase and program verify commands. 0 = Normal 1 = User (default) 2 = Factory
TargetVersion	No	18h	4	SoC target build version number

5.5.1 Property Definitions

Get/Set property definitions are provided in this section.

5.5.1.1 CurrentVersion Property

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

Table 5-51. Fields of CurrentVersion property:

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

5.5.1.2 AvailablePeripherals Property

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

Table 5-52. Peripheral bits:

Bit	[31:7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
Peripheral	Reserved	Reserved	Reserved	Reserved	Reserved	SPI Slave	LPI2C Slave	LPUART

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

5.5.1.3 AvailableCommands Property

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

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

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

Table 5-53. Command bits:

Bit	[31:18]	[17]	[16]	[15]	[14]	[13]	[12]	[11]	[10]	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]

Table continues on the next page...

Table 5-53. Command bits: (continued)

Command	Reserved	Reserved	Reserved	FlashReadResource	FlashReadOnce	FlashProgramOnce	FlashEraseAllUnsecure	Reserved	Reset	Reserved	Execute	Reserved	GetProperty	FlashSecurityDisable	Reserved	WriteMemory	ReadMemory	FlashEraseRegion	FlashEraseAll

5.6 Verifying the application in flash using CRC-32

Using CRC-32 and a given address range, the ROM bootloader supports performing an application integrity check.

Before application integrity testing: Application should have the same BCA fields (offset 0x3C0 from the beginning of user application), but only CRC-related fields are used by ROM for doing application integrity check.

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

Application integrity testing:

- If all of the above fields are unset (all 0xFF bytes for crcStartAddress, crcByteCount, and crcExpectedValue), then the ROM bootloader returns kStatus_AppCrcCheckInactive (10402), and ROM will not do CRC check on application.
- If any one of the above fields are set (crcStartAddress, crcByteCount, and crcExpectedValue), then the ROM bootloader checks if the given address range of the application is valid, and if the application resides in internal flash, by running a CRC check on the image in flash:
  - If the the given address range of the application is not valid (false), then the bootloader returns kStatus_AppCrcCheckOutOfRange.
  - If the given address range of the application is valid (true), then the CRC check is performed.
    - If the CRC check fails, then the bootloader returns kStatus_AppCrcCheckFailed;
    - If the CRC check succeeds, then it returns kStatus_AppCrcCheckPassed.

After application integrity testing:

- The bootloader will jump to the application if kStatus_AppCheckPassed is return, or no CRC check (*kStatus_AppCrcCheckInactive*); if kStatus_AppCrcCheckPassed is not returned, then the bootloader will stay active, and wait for further commands.

5.7 Bootloader Status Error Codes

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

Table 5-54. Bootloader Status Error Codes, sorted by Value

Error Code	Value	Description
kStatus_Success	0	Operation succeeded without error.
kStatus_Fail	1	Operation failed with a generic error.
kStatus_ReadOnly	2	Requested value cannot be changed because it is read-only.
kStatus_OutOfRange	3	Requested value is out of range.
kStatus_InvalidArgument	4	The requested command's argument is undefined.
kStatus_Timeout	5	A timeout occurred.
kStatus_FlashSizeError	100	Not used.
kStatus_FlashAlignmentError	101	Address or length does not meet required alignment.
kStatus_FlashAddressError	102	Address or length is outside addressable memory.
kStatus_FlashAccessError	103	The FTFA_FSTAT[ACCERR] bit is set.
kStatus_FlashProtectionViolation	104	The FTFA_FSTAT[FPVIOL] bit is set.
kStatus_FlashCommandFailure	105	The FTFA_FSTAT[MGSTAT0] bit is set.
kStatus_FlashUnknownProperty	106	Unknown Flash property.
kStatus_FlashEraseKeyError	107	The key provided does not match the programmed flash key.
kStatus_FlashRegionExecuteOnly	108	The area of flash is protected as execute only.
kStatus_I2C_SlaveTxUnderrun	200	I2C Slave TX Underrun error.
kStatus_I2C_SlaveRxOverrun	201	I2C Slave RX Overrun error.
kStatus_I2C_AribtrationLost	202	I2C Arbitration Lost error.
kStatus_UnknownCommand	10000	The requested command value is undefined.
kStatus_SecurityViolation	10001	Command is disallowed because flash security is enabled.
kStatus_AbortDataPhase	10002	Abort the data phase early.
kStatus_Ping	10003	Internal: received ping during command phase.
kStatusMemoryRangeInvalid	10200	Memory range conflicts with a protected region.
kStatus_UnknownProperty	10300	The requested property value is undefined.
kStatus_ReadOnlyProperty	10301	The requested property value cannot be written.
kStatus_InvalidPropertyValue	10302	The specified property value is invalid.
kStatus_AppCrcCheckPassed	10400	CRC check is valid and passed.

Table continues on the next page...

Table 5-54. Bootloader Status Error Codes, sorted by Value (continued)

Error Code	Value	Description
kStatus_AppCrcCheckFailed	10401	CRC check is valid but failed.
kStatus_AppCrcCheckInactive	10402	CRC check is inactive.
kStatus_AppCrcCheckInvalid	10403	CRC check is invalid, because the BCA is invalid or the CRC parameters are unset (all 0xFF bytes).
kStatus_AppCrcCheckOutOfRange	10404	CRC check is valid but addresses are out of range.

5.8 ROM flash driver API

5.8.1 Introduction

The main purpose of these APIs is to simplify the use of flash driver APIs, which are exported from ROM bootloader. With APIs, the user does not need to care about the details of flash commands. A set of parameters are required to ensure all APIs work properly.

5.8.2 Struct of FlashDriverInterface

The bootloader flash driver API tree contains pointers of flash driver functions. The following code snippet shows example of the struct of flash driver interface.

```
/*******************************************************************************
                          Structure with API function pointers members

*******************************************************************************/
//! @brief Interface for the flash driver.
typedef struct FlashDriverInterface
{
    standard_version_t version; //!< flash driver API version number.
    status_t (*flash_init)(flash_config_t *config);
    status_t (*flash_erase_all)(flash_config_t *config, uint32_t key);
    status_t (*flash_erase_all_unsecure)(flash_config_t *config, uint32_t key);
    status_t (*flash_erase)(flash_config_t *config, uint32_t start, uint32_t lengthInBytes,
uint32_t key);
    status_t (*flash_program)(flash_config_t *config, uint32_t start, uint8_t *src, uint32_t
lengthInBytes);
    status_t (*flash_get_security_state)(flash_config_t *config, flash_security_state_t
*state);
    status_t (*flash_security_bypass)(flash_config_t *config, const uint8_t *backdoorKey);
    status_t (*flash_verify_erase_all)(flash_config_t *config, flash_margin_value_t margin);
    status_t (*flash_verify_erase)(flash_config_t *config,
                                   uint32_t start,
                                   uint32_t lengthInBytes,
                                   flash_margin_value_t margin);
    status_t (*flash_verify_program)(flash_config_t *config,
                                     uint32_t start,
```

```
                                    uint32_t lengthInBytes,
                                    const uint8_t *expectedData,
                                    flash_margin_value_t margin,
                                    uint32_t *failedAddress,
                                    uint32_t *failedData);
    status_t (*flash_get_property)(flash_config_t *config, flash_property_tag_t
whichProperty, uint32_t *value);
    status_t (*flash_program_once)(ftfx_config_t *config, uint32_t index, uint8_t *src,
uint32_t lengthInBytes);
    status_t (*flash_read_once)(ftfx_config_t *config, uint32_t index, uint8_t *dst,
uint32_t lengthInBytes);
    status_t (*flash_read_resource)(flash_config_t *config,
                                    uint32_t start,
                                    uint8_t *dst,
                                    uint32_t lengthInBytes,
                                    flash_read_resource_opt_t option);
} flash_driver_interface_t;
```

5.8.3 Details of structs

The following table shows the list of structs related to flash_config_t.

Struct Name	Description
flash_config_t	Struct passed to flash driver API
ftfx_config_t	Struct of flash driver state information
flash_mem_desc_t	Struct for flash memory descriptor
flash_ops_config_t	Active FTFx information for the current operation
flash_spec_mem_t	Struct for ftfx special memory access information

- Struct flash_config_t

```
#define FTFx_FLASH_COUNT 1
typedef struct _flash_config
{
    ftfx_config_t ftfxConfig[FTFx_FLASH_COUNT];
} flash_config_t;
```

NOTE

For this device, there is only one FLASH, so FTFx_FLASH_COUNT equals to 1.

- Struct ftfx_config_t

  It defines the flash driver state information.

```
typedef struct _ftfx_config
{
    flash_mem_desc_t flashDesc;
    flash_ops_config_t opsConfig;
    uint32_t flexramBlockBase;              /*!< The base address of the FlexRAM/
acceleration RAM */
    uint32_t flexramTotalSize;              /*!< The size of the FlexRAM/acceleration
RAM */
    uint16_t eepromTotalSize;               /*!< The size of EEPROM area which was
partitioned from FlexRAM */
    uint16_t reserved;
```

```
    uint32_t *runCmdFuncAddr;               /*!< An buffer point to the flash execute-
in-RAM function. */
    flash_ifr_desc_t ifrDesc;
} ftfx_config_t;
```

- Struct flash_mem_desc_t

  It defines the struct of flash memory descriptor.

```
typedef struct _flash_mem_descriptor
{
    uint8_t type;                     /*!< Type of flash block.*/
    uint8_t index;                    /*!< Index of flash block.*/
    uint8_t reserved[2];
    struct {
        uint32_t isIndBlock:1;
        uint32_t hasIndPfsizeReg:1;
        uint32_t hasProtControl:1;
        uint32_t hasIndProtReg:1;
        uint32_t hasXaccControl:1;
        uint32_t hasIndXaccReg:1;
        uint32_t :18;
        uint32_t ProtRegBits:8;
    } feature;

    uint32_t blockBase;                /*!< A base address of the flash block */
    uint32_t totalSize;                /*!< The size of the flash block. */
    uint32_t sectorSize;               /*!< The size in bytes of a sector of flash. */
    uint32_t blockCount;               /*!< A number of flash blocks. */
    flash_spec_mem_t accessSegmentMem;
    flash_spec_mem_t protectRegionMem;
} flash_mem_desc_t;
```

- Struct flash_spec_mem_t

  It defines the special flash memory access information.

```
typedef struct _flash_special_mem
{
    uint32_t base;  /*!< Base address of flash special memory.*/
    uint32_t size;  /*!< size of flash special memory.*/
    uint32_t count; /*!< flash special memory count.*/
} flash_spec_mem_t;
```

- Struct flash_ops_config_t

  It defines the active flash information for the current operation.

```
typedef struct _flash_ops_config
{
    uint32_t convertedAddress;          /*!< A converted address for the current flash
type.*/
    struct {
        uint8_t sectorCmd;
        uint8_t sectionCmd;
        uint8_t resourceCmd;
        uint8_t checkCmd;
        uint8_t swapCtrlCmd;
        uint8_t blockWriteUnitSize;
        uint8_t reserved[2];
    } addrAligment;
} flash_ops_config_t;
```

5.8.4 Flash Driver APIs

Flash commands	Flash Driver API functions
	FLASH_Init
0x01, Read 1s Section	FLASH_VerifyErase
0x02 Program Check	FLASH_VerifyProgram
0x03 Read Resource	FLASH_ReadResource
0x09 Erase Flash Sector	FLASH_Erase
0x40 Read 1s All Blocks	FLASH_VerifyEraseAll
0x41 Read Once	FLASH_ReadOnce
0x43 Program Once	FLASH_ProgramOnce
0x44 Erase All Blocks	FLASH_EraseAll
0x45 Verify Backdoor Access Key	FLASH_SecurityBypass
0x49 Erase All Blocks Unsecure	FLASH_EraseAllUnsecure
	FLASH_GetSecurityState
	FLASH_GetProperty

- FLASH_Init
  It checks and initializes the flash module for the other flash API functions.

NOTE

FLASH_Init must be always called before calling other API functions.

```
status_t FLASH_Init(flash_config_t *config)
```

Parameter	Description
config	Config Pointer to storage for the driver runtime state.

Possible status response

Value	Constant	Description
4	kStatus_InvalidArgument	Config is NULL.
100	kStatus_FLASH_SizeError	Returned flash is incorrect.
0	kStatus_Success	This function has performed successfully.

Refer to the Bootloader Reference Manual, for more details on each API command.

5.8.5 Integrate flash driver API to user project

There are several steps to integrate the Flash Driver APIs to user project.

1. Add flash driver related files (rom_api.h, rom_flash_api.h and rom_flash_api.c, etc.) to corresponding project.
2. Declaim variable for an instance of flash_config_t.
   To call flash driver API, an instance of this structure need to be allocated by the user and passed into each of the driver APIs. For example,

```
flash_config_t g_flashState;

while
#define FTFx_FLASH_COUNT 1
typedef struct _flash_config
{
    ftfx_config_t ftfxConfig[FTFx_FLASH_COUNT];
} flash_config_t;
```

then call `flash_init(g_flashState)`.

Chapter 6
Power Management

6.1 Introduction

Power management provides an on-chip supply regulation to regulate an external VDD (3.3V in typical) supply down to 2.7V and 1.2V levels, for use with internal circuitry and Power-On-Reset (POR) generator and Low Voltage Interrupts (LVIs). On-chip regulators are stable and have sufficient capacity and dynamic response, allowing device to operate correctly at all time and under all combinations of specified loads, frequencies, voltages, temperatures and fabrication process variations. POR generates a reset signal when power is applied to the device. It ensures that the device starts operating in a known state. The low-voltage detection monitors VDD, and is able to generate interrupts when VDD drops below the preset voltage thresholds. CPU can response to LVIs, and place the device in a safe state before supply voltage drops below the operating voltage.

Power consumption during device operation is managed by powering off non-essential analog modules, and gating off the clock to digital circuitry associated with the modes of RUN, WAIT, and STOP.

6.2 Function Description

The device is power-supplied by both digital supply VDD and analog supply VDDA. The VDDA is used directly by analog modules. Power regulation is managed by the Power Management Controller (PMC) module whose input is from VDD. There are two different types of regulators in PMC:

- Small regulator used to maintain a stable low-noise 2.7V supply as well as 1.2V supply, for PMC, oscillator, PLL and other critical analog circuitry.
- A separate large regulator generates a 1.2V whose output is stabilized by external capacitor on Vcap pin, to supply power for other internal digital circuitry (such as CPU, memory, timers, communication modules, etc.).

The large regulator supports both full power mode and standby mode. The maximum operating frequency in this standby mode is limited to 2 MHz. The small 1.2V regulator supports full power and standby mode. The small 2.7V regulator supports full power, standby, and power down modes. In the small 2.7V regulator's power down mode, the 2.7 V supply is completely shut off. As a result, all on-chip clock sources are disabled. To operate the chip in this mode, the user must provide an external clock through the CLKIN0/1 pin. The power mode controls for the regulators are in the SIM module.

The PMC incorporates power-on reset logic and two levels of low voltage detection. A hysteresis circuit ensures that power-on reset does not release until the supply rises above the Low-Voltage Warning (LVI_2p7) threshold, which is around 2.7V, where the device can operate safely over all conditions. On falling voltage conditions, the device continues to operate below LVI_2p7 and a lower Low-Voltage Alarm (LVI_2p2) threshold, which is around 2.2V, until a POR level is reached where the device is ultimately forced to reset. The interrupts associated with both thresholds are able to generate when each threshold is reached. This approach offers the opportunity to disable non-essential operations and configure the device in a safe minimum power state until the power is restored. Under falling voltage, POR reset does not assert until the low POR threshold is violated at 2.0 V, so that the maximum time is allowed for orderly shutdown.

The primary methods for power management during device operation are the module-specific clock enables, clock frequency control, clock gating associated with the use of low power modes, and clock gating added by power synthesis. The individual digital module can be placed in sleep state by turning off the module clock in SIM. Analog module can be placed in low power mode by enabling the power down mode in the respective module. Clock frequency control is performed by the OCCS module which provides flexible control of operating frequency. Clock tree implementation further limits power by denying clocks to registers which are not changing state.

6.3 User Power Management Methods

The primary means available to manage power consumption are clock enables, limiting clock frequency, using low power modes, and using DMA. Further indirect power savings can be achieved by reducing unnecessary signal transitions through careful crafting of application software.

The most basic form of power control are module enables. Selected peripherals provide their own clock gating controls back to the SIM. Using these controls, clock generation is gated off in the SIM when not required by the peripheral. Module enables are also used to power off embedded analog functions when not in use. It is therefore advantageous to leave system and peripheral functions disabled if not currently in use.

Clock frequency control is performed in the OCCS. The OCCS provides the means to perform glitch-free transitions between the available clock sources including external crystal oscillator, 8 MHz / 2 MHz Internal Reference Clock (IRC), 200 kHz internal oscillator and an external clock. Only clock sources currently in use should be powered on. The PLL should only be powered on when necessary to produce higher operating frequencies. A post scaler with up to 256x division is provided to reduce the active clock source frequency (PLL output or oscillator) even further for power savings. This gives the part flexible frequency control from full speed (100 MHz core rate and 50 MHz system bus rate) to below 1 kHz for both core and system bus.

Low power modes are entered via the core processor upon execution of STOP or WAIT instructions and conveyed to the SIM, which performs related clock gating for RUN, WAIT, and STOP modes. The processor and processor dependent clocks operate only in RUN mode. Peripheral clocks do not operate in any power mode until enabled individually by the user in the SIM. Once enabled, they normally run only in RUN and WAIT modes and must be specifically overridden with settings in the respective module, to remain clocked in STOP mode.

The DMA controller contributes to power management by permitting peripheral I/O to be serviced by the DMA while the core remains unclocked in WAIT mode. By eliminating the requirement for the core to service the related interrupts, the DMA decreases the frequency and duration of intervals where the core must be returned to RUN mode to service interrupts.

Additional power savings can be achieved by crafting application software to avoid unnecessary register state transitions and to configure module-specific frequency controls to use the lowest-supportable operating rates that satisfy application requirements.

6.4 Power Modes

The CPU has three primary modes of operation: RUN, WAIT, and STOP modes. Additional low power operating modes can reduce run-time power when maximum bus frequency is not required to support application needs.

The flash memory module has three power modes: Full Power mode, Low Power mode, and Low Power Stop mode. When the device enters low power mode, flash memory will enter the corresponding low power mode. Flash read is allowed in low power mode but operation frequency must be below 1 MHz, while all flash commands are not executable.

DMA can be enabled in any of the power modes. About more details of eDMA in STOP mode, see "Modes of Operation" section in the SIM chapter.

Various of the low power modes (set by adjusting clock rate in OCCS and SIM_PWRMODE register) function only if the FOPT[0] bit is set.

Table 6-1. Power Modes of Operation (fast mode: with core frequency 100 MHz and bus clock 50 MHz)

Chip Power Mode	CPU Clock	Peripheral Clock	Small Regulator 2.7 V	Small Regulator 1.2 V	Large Regulator 1.2 V	Flash Power Mode	Max. System Clock Freq.	Wakeup Source
RUN	ON	ON	Full Power	Full Power	Full Power	Full Power	100 MHz	NA
WAIT	OFF	ON	Full Power	Full Power	Full Power	Full Power	100 MHz	Peripheral interrupt and/or reset
STOP[1]	OFF	OFF[2]	Full Power	Full Power	Full Power	Full Power	100 MHz [2]	Interrupt from clock enabled peripherals (SD bit support)/ reset/ async interrupts (I2C, SCI, CMP)
LPRUN	ON	ON	Standby	Standby	Standby	Low Power [3]	2 MHz[3]	NA
LPWAIT	OFF	ON	Standby	Standby	Standby	Low Power[3]	2 MHz[3]	Peripheral interrupt and/or reset
LPSTOP[1]	OFF	OFF[2]	Standby	Standby	Standby	Low Power Stop[3]	2 MHz[2, 3]	Interrupt from clock enabled peripherals (SD bit support)/ reset/ async interrupts (I2C, SCI, CMP)
VLPRUN [4]	ON	ON	Power Down	Standby	Standby	Low Power	200 kHz	NA
VLPWAIT[4]	OFF	ON	Power Down	Standby	Standby	Low Power	200 kHz	Peripheral interrupt and/or reset
VLPSTOP[1, 4]	OFF	OFF[2]	Power Down	Standby	Standby	Low Power Stop	200 kHz[2]	Interrupt from clock enabled peripherals (SD bit support)/ reset/ async interrupts (I2C, SCI, CMP)

1. If DMA is enabled in any STOP mode, flash memory will not enter its Low Power Stop mode.

2. In all STOP modes, the clock to some portions of SIM logic is never gated. In addition, the user can enable the clock of select peripherals by setting the corresponding bit in one of the SIM's SDn registers.
3. For any chip LP mode, the clock source ROSC should be in standby mode. In that case, the maximum system frequency is 2 MHz. The PLL is shut down in LP modes. In all chip LP modes and flash memory Low Power modes, the maximum frequency for flash memory operation is 500 kHz due to the fixed frequency ratio of 1:4 between the CPU clock and the flash clock.
4. When the chip is in any VLP mode, all internal clock sources are unavailable. To operate the chip in those modes, the user must provide a clock through the CLKIN0/1 port.

Table 6-2. Power Modes of Operation (normal mode: with both core frequency and bus clock as 50 MHz)

Chip Power Mode	CPU Clock	Peripheral Clock	Small Regulator 2.7 V	Small Regulator 1.2 V	Large Regulator 1.2 V	Flash Power Mode	Max. System Clock Freq.	Wakeup Source
RUN	ON	ON	Full Power	Full Power	Full Power	Full Power	50 MHz	NA
WAIT	OFF	ON	Full Power	Full Power	Full Power	Full Power	50 MHz	Peripheral interrupt and/or reset
STOP[1]	OFF	OFF[2]	Full Power	Full Power	Full Power	Full Power	50 MHz [2]	Interrupt from clock enabled peripherals (SD bit support)/ reset/ async interrupts (I2C, SCI, CMP)
LPRUN	ON	ON	Standby	Standby	Standby	Low Power [3]	1 MHz[3]	NA
LPWAIT	OFF	ON	Standby	Standby	Standby	Low Power[3]	1 MHz[3]	Peripheral interrupt and/or reset
LPSTOP[1]	OFF	OFF[2]	Standby	Standby	Standby	Low Power Stop[3]	1 MHz[2, 3]	Interrupt from clock enabled peripherals (SD bit support)/ reset/ async interrupts (I2C, SCI, CMP)
VLPRUN [4]	ON	ON	Power Down	Standby	Standby	Low Power	100 kHz	NA
VLPWAIT[4]	OFF	ON	Power Down	Standby	Standby	Low Power	100 kHz	Peripheral interrupt and/or reset
VLPSTOP[1, 4]	OFF	OFF[2]	Power Down	Standby	Standby	Low Power Stop	100 kHz[2]	Interrupt from clock enabled peripherals (SD bit support)/ reset/ async

Table 6-2. Power Modes of Operation (normal mode: with both core frequency and bus clock as 50 MHz)

Chip Power Mode	CPU Clock	Peripheral Clock	Small Regulator 2.7 V	Small Regulator 1.2 V	Large Regulator 1.2 V	Flash Power Mode	Max. System Clock Freq.	Wakeup Source
								interrupts (I2C, SCI, CMP)

1. If DMA is enabled in any STOP mode, flash memory will not enter its Low Power Stop mode.
2. In all STOP modes, the clock to some portions of SIM logic is never gated. In addition, the user can enable the clock of select peripherals by setting the corresponding bit in one of the SIM's SDn registers.
3. For any chip LP mode the clock source ROSC should be in standby mode. In that case, the maximum system frequency is 1 MHz. The PLL is shut down in LP modes. In all chip LP modes and flash memory Low Power modes, the maximum frequency for flash memory operation is 500 kHz due to the fixed frequency ratio of 1:2 between the CPU clock and the flash clock.
4. When the chip is in any VLP mode, all internal clock sources are unavailable. To operate the chip in those modes, the user must provide a clock through the CLKIN0/1 port.

6.5 Power Mode Transitions

The following figure shows the chip's power modes and the available transitions among them.

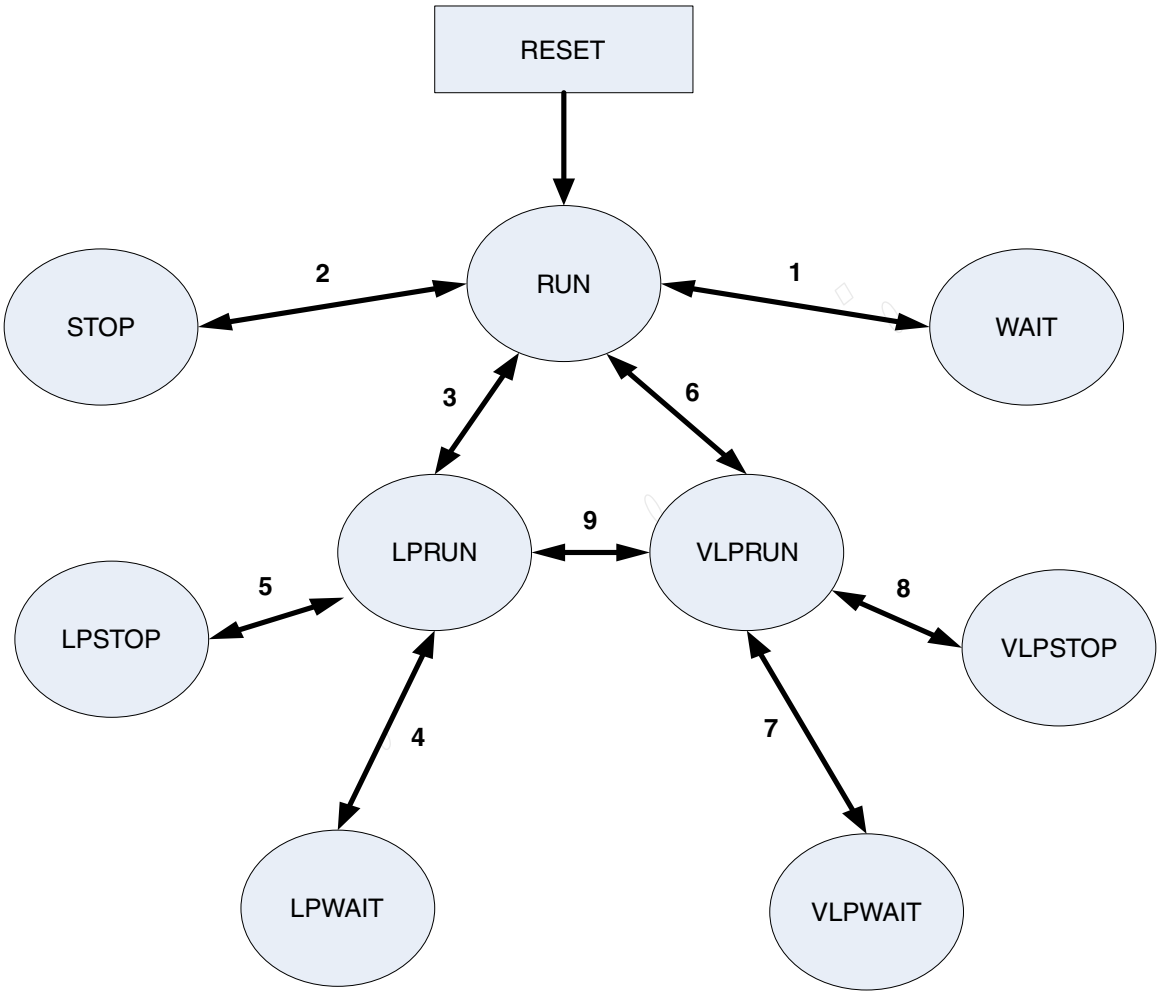


Figure 6-1. Power mode state transitions

The following table defines triggers for the various state transitions shown in the preceding figure.

NOTE

To prevent current leakage in any VLP mode, ensure the following settings apply before entering the VLP mode:

1. The OCCS_CTRL[PLLPD] bit is 1.
2. The PWMA_FRCTRL[FRAC_PU] bit is 0.
3. The OCCS_OSCTL1[ROPD], OSCTL2[COPD], and OSCTL2[ROPD200K] bits are each 1.

Table 6-3. Power Mode Transitions (fast mode: with core frequency 100 MHz and bus clock 50 MHz)

Transition number	From	To	Device configuration and/or trigger condition
1	RUN	WAIT	1. Ensure the SIM_CTRL[0] bit is clear (SIM_CTRL[WAIT_DISABLE] = 00b or 10b).

Table continues on the next page...

Table 6-3. Power Mode Transitions (fast mode: with core frequency 100 MHz and bus clock 50 MHz) (continued)

Transition number	From	To	Device configuration and/or trigger condition
			2. Execute the WAIT instruction.
	WAIT	RUN	Synchronous interrupt or reset
2	RUN	STOP	1. Ensure SIM_CTRL[2] is clear. 2. Execute the STOP instruction.
	STOP	RUN	1. Synchronous interrupt from a peripheral whose clock is enabled in stop mode by setting the corresponding SIM_SDn bit. 2. Asynchronous interrupt from an I2C module, a CMP module, or an LVD event. 3. POR or PIN reset.
3	RUN	LPRUN	1. If the system is running with the PLL, then switch the clock source to XOSC or internal oscillators by setting OCCS_CTRL[ZSRC] to 0b. 2. Before changing ZSRC, ensure the MSTR_OSC clock is stable. 3. Put the PLL in power down mode by setting OCCS_CTRL[PLLPD]. 4. If the internal oscillator is used during LP mode, put the crystal oscillator in power down mode by setting OSCTL2[COPD] to 1, and put IRC in standby mode (2MHz) by settting OCCS_OSCTL1[ROSB] to 1 . 5. Similarly, if the crystal oscillator is used, put the internal oscillators in power down mode by setting OSCTL1[ROPD] to 1 and OSCTL2[ROPD200K] to 1. 6. Configure the OCCS_DIVBY[COD] field to ensure the system clock frequency (SYS_CLK) does not exceed 2 MHz. 7. Set SIM_PWRMODE[LPMODE].
	LPRUN	RUN	Clear the SIM_PWRMODE[LPMODE] bit.
4	LPRUN	LPWAIT	1. Ensure SIM_CTRL[0] is clear. 2. Execute the WAIT instruction.
	LPWAIT	LPRUN	Synchronous interrupt or reset
5	LPRUN	LPSTOP	1. Ensure SIM_CTRL[2] is clear. 2. Execute the STOP instruction.
	LPSTOP	LPRUN	1. Synchronous interrupt from a peripheral whose clock is enabled in stop mode by setting the corresponding SIM_SDn bit. 2. Asynchronous interrupt from an I2C module, a CMP module, or an LVD event. 3. POR or PIN reset.
6	RUN	VLPRUN	1. Configure OCCS registers (EXT_SEL, PRECS, ZSRC, and COD) to ensure the CLKIN path is selected as the system clock (SYS_CLK) with a maximum frequency of 200 kHz. 2. Set SIM_PWRMODE[VLPMODE] to 1.
	VLPRUN	RUN	1. Clear the SIM_PWRMODE[VLPMODE] and SIM_PWRMODE[LPMODE] bits (if they are already set). 2. Wait for the PMC_STS[SR27] bit to be set.
7	VLPRUN	VLPWAIT	1. Ensure SIM_CTRL[0] is clear. 2. Execute the WAIT instruction.
	VLPWAIT	VLPRUN	Synchronous interrupt or reset
8	VLPRUN	VLPSTOP	1. Ensure SIM_CTRL[2] is clear. 2. Execute the STOP instruction.
	VLPSTOP	VLPRUN	1. Synchronous interrupt from a peripheral whose clock is enabled in stop mode by setting the corresponding SIM_SDn bit.

Table continues on the next page...

Table 6-3. Power Mode Transitions (fast mode: with core frequency 100 MHz and bus clock 50 MHz) (continued)

Transition number	From	To	Device configuration and/or trigger condition
			2. Asynchronous interrupt from an I2C module, a CMP module, or an LVD event. 3. POR or PIN reset.
9	LPRUN	VLPRUN	1. Configure OCCS registers (EXT_SEL, PRECS, ZSRC, and COD) to ensure the CLKIN path is selected as the system clock (SYS_CLK) with a maximum frequency of 200 kHz. 2. Set SIM_PWRMODE[VLPMODE] to 1.
	VLPRUN	LPRUN	1. Ensure the LPMODE bit is set. 2. Clear SIM_PWRMODE[VLPMODE] to 0. 3. Wait for PMC_STS[SR27] bit to be set.

Table 6-4. Power Mode Transitions (normal mode: with both core frequency and bus clock as 50 MHz)

Transition number	From	To	Device configuration and/or trigger condition
1	RUN	WAIT	1. Ensure the SIM_CTRL[0] bit is clear (SIM_CTRL[WAIT_DISABLE] = 00b or 10b). 2. Execute the WAIT instruction.
	WAIT	RUN	Synchronous interrupt or reset
2	RUN	STOP	1. Ensure SIM_CTRL[2] is clear. 2. Execute the STOP instruction.
	STOP	RUN	1. Synchronous interrupt from a peripheral whose clock is enabled in stop mode by setting the corresponding SIM_SDn bit. 2. Asynchronous interrupt from an I2C module, a CMP module, or an LVD event. 3. POR or PIN reset.
3	RUN	LPRUN	1. If the system is running with the PLL, then switch the clock source to XOSC or internal oscillators by setting OCCS_CTRL[ZSRC] to 0b. 2. Before changing ZSRC, ensure the MSTR_OSC clock is stable. 3. Put the PLL in power down mode by setting OCCS_CTRL[PLLPD]. 4. If the internal oscillator is used during LP mode, put the crystal oscillator in power down mode by setting OSCTL2[COPD] to 1, and put IRC in standby mode (2MHz) by settting OCCS_OSCTL1[ROSB] to 1. 5. Similarly, if the crystal oscillator is used, put the ROSC in power down mode by setting OSCTL1[ROPD] to 1 and OSCTL2[ROPD200K] to 1. 6. Configure the OCCS_DIVBY[COD] field to ensure the system clock frequency (SYS_CLK) does not exceed 1 MHz. 7. Set SIM_PWRMODE[LPMODE].
	LPRUN	RUN	Clear the SIM_PWRMODE[LPMODE] bit.
4	LPRUN	LPWAIT	1. Ensure SIM_CTRL[0] is clear. 2. Execute the WAIT instruction.
	LPWAIT	LPRUN	Synchronous interrupt or reset
5	LPRUN	LPSTOP	1. Ensure SIM_CTRL[2] is clear. 2. Execute the STOP instruction.

Table continues on the next page...

Table 6-4. Power Mode Transitions (normal mode: with both core frequency and bus clock as 50 MHz) (continued)

Transition number	From	To	Device configuration and/or trigger condition
	LPSTOP	LPRUN	1. Synchronous interrupt from a peripheral whose clock is enabled in stop mode by setting the corresponding SIM_SDn bit. 2. Asynchronous interrupt from an I2C module, a CMP module, or an LVD event. 3. POR or PIN reset.
6	RUN	VLPRUN	1. Configure OCCS registers (EXT_SEL, PRECS, ZSRC, and COD) to ensure the CLKIN path is selected as the system clock (SYS_CLK) with a maximum frequency of 100 kHz. 2. Set SIM_PWRMODE[VLPMODE] to 1.
	VLPRUN	RUN	1. Clear the SIM_PWRMODE[VLPMODE] and SIM_PWRMODE[LPMODE] bits (if they are already set). 2. Wait for the PMC_STS[SR27] bit to be set.
7	VLPRUN	VLPWAIT	1. Ensure SIM_CTRL[0] is clear. 2. Execute the WAIT instruction.
	VLPWAIT	VLPRUN	Synchronous interrupt or reset
8	VLPRUN	VLPSTOP	1. Ensure SIM_CTRL[2] is clear. 2. Execute the STOP instruction.
	VLPSTOP	VLPRUN	1. Synchronous interrupt from a peripheral whose clock is enabled in stop mode by setting the corresponding SIM_SDn bit. 2. Asynchronous interrupt from an I2C module, a CMP module, or an LVD event. 3. POR or PIN reset.
9	LPRUN	VLPRUN	1. Configure OCCS registers (EXT_SEL, PRECS, ZSRC, and COD) to ensure the CLKIN path is selected as the system clock (SYS_CLK) with a maximum frequency of 100 kHz. 2. Set SIM_PWRMODE[VLPMODE] to 1.
	VLPRUN	LPRUN	1. Ensure the LPMODE bit is set. 2. Clear SIM_PWRMODE[VLPMODE] to 0. 3. Wait for PMC_STS[SR27] bit to be set.

Chapter 7
Memory Resource Protection (MRP)

Memory resource protection allows two levels of software to co-exist in the DSC core, while maintaining hardware-enforced isolation. The levels are designated supervisor software and user software. The hardware feature along with appropriate software architecture allow the system to protect supervisor programs and resources being accessed from user programs.

This resource protection allows unverified or third-party software to safely run in user mode, allowing it to only access unprotected memory locations and resources.

The software enablement of resource protection is provided in the MCM's Resource Protection Control Register (RPCR).

The resource protection features provided are:

- Partition software into two modes: supervisor software and user software
- Partition system address spaces and resources, both code and data, into two modes
- Provide secure methods of transfer between modes
- Provide separate stacks and stack pointers for supervisor and user modes

7.1 Overview

The software is able to be partitioned into two modes: supervisor and user. The system resources are partitioned for both code and data into supervisor and user regions. Supervisor code can only be executed from supervisor regions and user code can only be executed from user regions. The data regions defined as supervisor can be accessed only by supervisor code; the user data regions can be accessed by both supervisor and user code.

Table 7-1. Resource Protection Accesses

System Resource	Supervisor Code	User Code
Supervisor data	Allowed	Prohibited

Table continues on the next page...

Table 7-1. Resource Protection Accesses (continued)

System Resource	Supervisor Code	User Code
User data	Allowed	Allowed
Peripheral space	Allowed	Prohibited
Debug resources	Allowed	Prohibited

NOTE

User mode cannot directly access peripheral space, or EOnCE registers. It can access only the user mode portion of flash memory, Boot ROM and RAM.

Supervisor code may access anywhere in supervisor or user space. Supervisor code may branch into user code by executing one of the three DSC "return from interrupt" instructions (RTI, RTID, FRTID). User code may access anywhere in user space. If user code attempts to branch into supervisor space or to reference data in supervisor space, the memory reference is aborted, producing a fault and generating an interrupt. User code may safely return control to supervisor program and data memory spaces with system calls using the SWI (software interrupt) instruction. Additionally, any interrupt exception forces entry to supervisor mode.

7.2 Features

The resource protection features are:

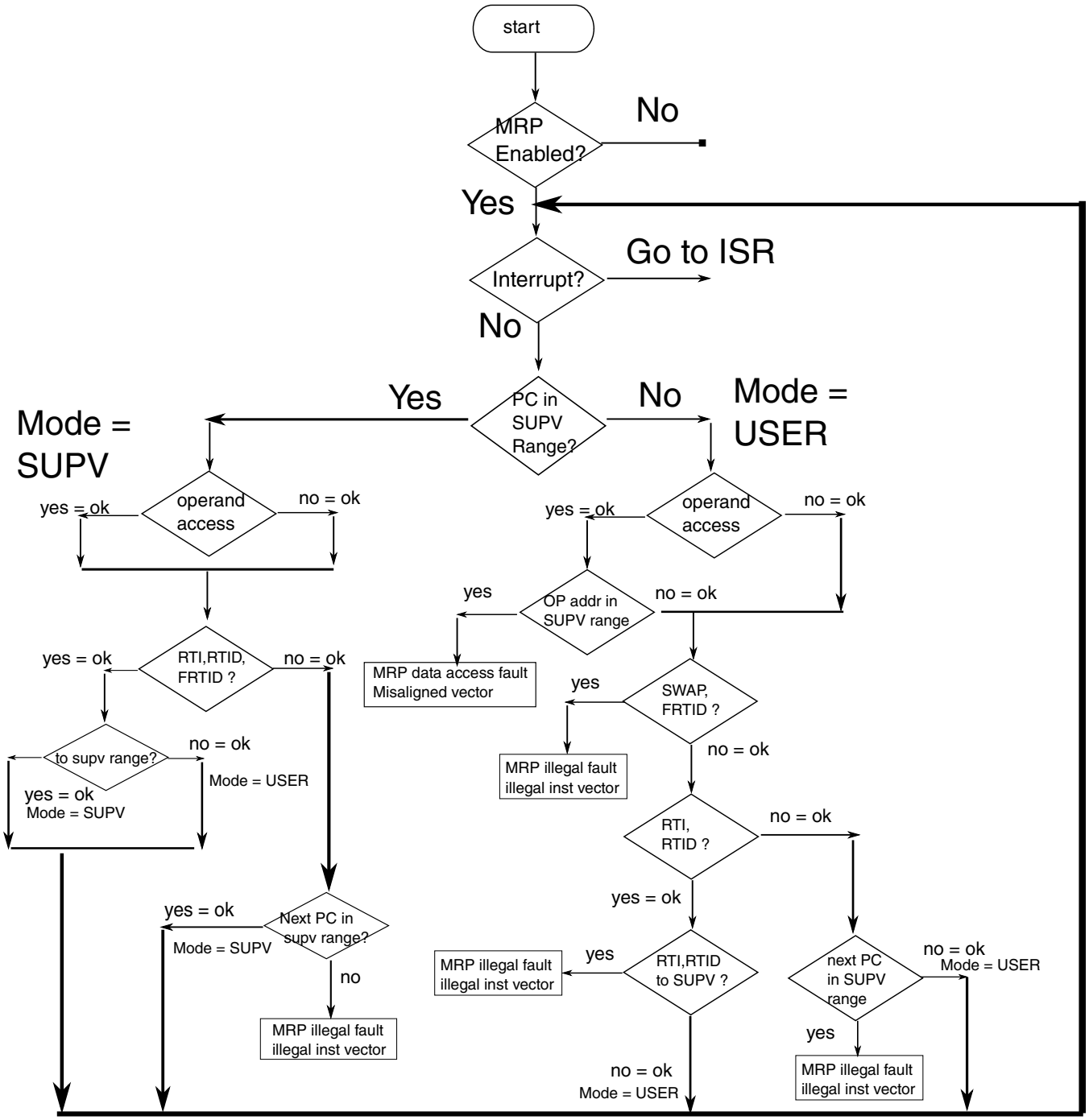
- Two stack pointer registers: active and "other"
- Flash memory regions defined with 8 KB granularity
- RAM memory regions defined with 512 byte granularity
- Boot ROM memory regions defined with 512 byte granularity
- Hardware-managed stack pointer register visible to software
- Supervisor stack pointer guaranteed on software faults and interrupts
- Stack pointer swap managed in hardware for supervisor-to-user transition via the RTI, RTID, FRTID instructions
- User-to-supervisor transition and stack pointer register swap managed on SWI instruction and all interrupts

7.3 Operation

Each of the protected code and data memory regions—flash memory, Boot ROM and RAM—are divided into two address spaces through supervisor-only memory-mapped programmable registers: the UFLASHBAR (User Flash Base Address Register),

UBROMBAR (User Boot ROM Base Address Register) and UPRAMBAR (User PRAM Base Address Register) . These registers define the base address for each region. Locations above this base address consist of user space, and locations below the base address are in the supervisor area.

The following diagram shows the allowed transitions between supervisor and user modes. Faulting transitions and illegal data references show the fault indicators for each illegal operation.



Terms:
ifault: Resource Protection illegal fault uses illegal instruction vector
dfault: Resource Protection data access fault uses misaligned vector
SUPV = supervisor
USER = user
operand access: current instruction that performs any operand read or write

Figure 7-1. MRP flows

The transition from supervisor code to user code is managed via the RTI, RTID, and FRTID instructions. The only transition from user code to supervisor code is via an interrupt or a fault triggered by a user software attempt to access a restricted region. Any other attempt results in a fault and subsequent return to supervisor mode. The preferred method for a user program to return to supervisor mode is to perform a system call via the SWI instruction. Additionally, user mode code is restricted from modifying SR (Status Register) bits [9:8] (interrupt mask level bits). Any attempt by user code to modify these bits is ignored.

Region transition is managed via the stack pointer. The DSC core controls the stack pointers (SP) as follows:

1. The hardware manages two SPs: a supervisor SP and a user SP. For correct and secure operation, the supervisor stack is located in supervisor RAM space and the user stack is located in user RAM space. Both spaces are defined by the contents of the UPRAMBAR.
2. The hardware manages one SP as the "active" stack pointer and the alternate as the "other" stack pointer. The active stack pointer resides in the core's SP register. The other stack pointer resides in a supervisor-only memory-mapped register.
3. On all faults and interrupts (supervisor mode entry points), the hardware guarantees the supervisor stack pointer is the active stack pointer. In other words, on all faults and interrupts:
   - If the supervisor stack pointer is in the SP register, interrupt processing continues.
   - If the user stack pointer is the active pointer and in the SP register, the hardware first swaps the SP register with the "other" stack pointer register, activating the supervisor stack pointer and saving the user SP before proceeding with interrupt processing.
4. Similar operations involving the two stack pointers are performed on possible supervisor mode exit points. On a return from interrupt instructions (RTI, RTID, FRTID), if the instruction is in supervisor code space and the target instruction address of the return is in user code space, the hardware:
   a. Swaps the SP holding the supervisor stack pointer with the "other" stack pointer register, activating the user stack pointer and saving the supervisor SP in the “other” SP register.
   b. Passes control to the user mode code.

   If the return target is located in the supervisor address space, then no stack pointer exchange is required and control immediately passes to the target instruction.
5. Stack pointer swaps occur such that all faults and interrupts are processed on the supervisor stack.
6. Stack pointer swaps do not incur any delay (they occur with zero cycle overhead).

The transfer from supervisor to user mode is done by the execution of a RTI, RTID, or FRTID instruction in supervisor space with a target in user space. All other transfers from supervisor code space to user code space result in a fault.

The only allowable state transition from user code to supervisor code is via a fault or an interrupt. The preferred, graceful method for a user program to perform a system call to pass control to supervisor code is via the SWI instruction.

7.4 Programming Model Overview

The resource protection registers are available only when memory resource protection is enabled. These registers reside in the MCM, and the addresses are offsets of the MCM's base address. The MCM is a supervisor-only space, so all registers must be accessed by supervisor code.

7.5 Memory Resource Protection Restrictions

In resource protection mode, the following restrictions apply:

- An attempt to execute the following supervisor-only instructions when in user mode produces an illegal fault:
  - FTRID
  - SWAP
- An attempt to execute the following instructions when in user mode results in an illegal fault if the target is in supervisor space:
  - RTI
  - RTID
- In resource protection mode, the following registers are supervisor only:
  - All shadow registers
- The SR (status register) bits [9:8] (interrupt mask level bits) cannot be modified by user code.

7.6 Base Address Setup

The UFLASHBAR, UBROMBAR and UPRAMBAR registers express the size of the portion of the flash memory, or Boot ROM, or program RAM that is used for supervisor space when resource protection is enabled. The hardware manages this information correctly for accesses to flash memory, or Boot ROM, or RAM for both program and data memory accesses.

UFLASHBAR Example

If resource protection is enabled and the primary program/data flash memory size is 64 KB (32 KW), setting the UFLASHBAR to 8000h defines the supervisor access region to be 32 KB. As a result, for both the program memory map and the data memory map, the supervisor access region occupies the bottom half of the primary program/data flash memory space and the user access region occupies the upper 32 KB of the space.

- Program (PDB bus) accesses to flash memory use the program memory map.
- Data (XAB1 or XAB2 bus) accesses to flash memory use the data memory map.

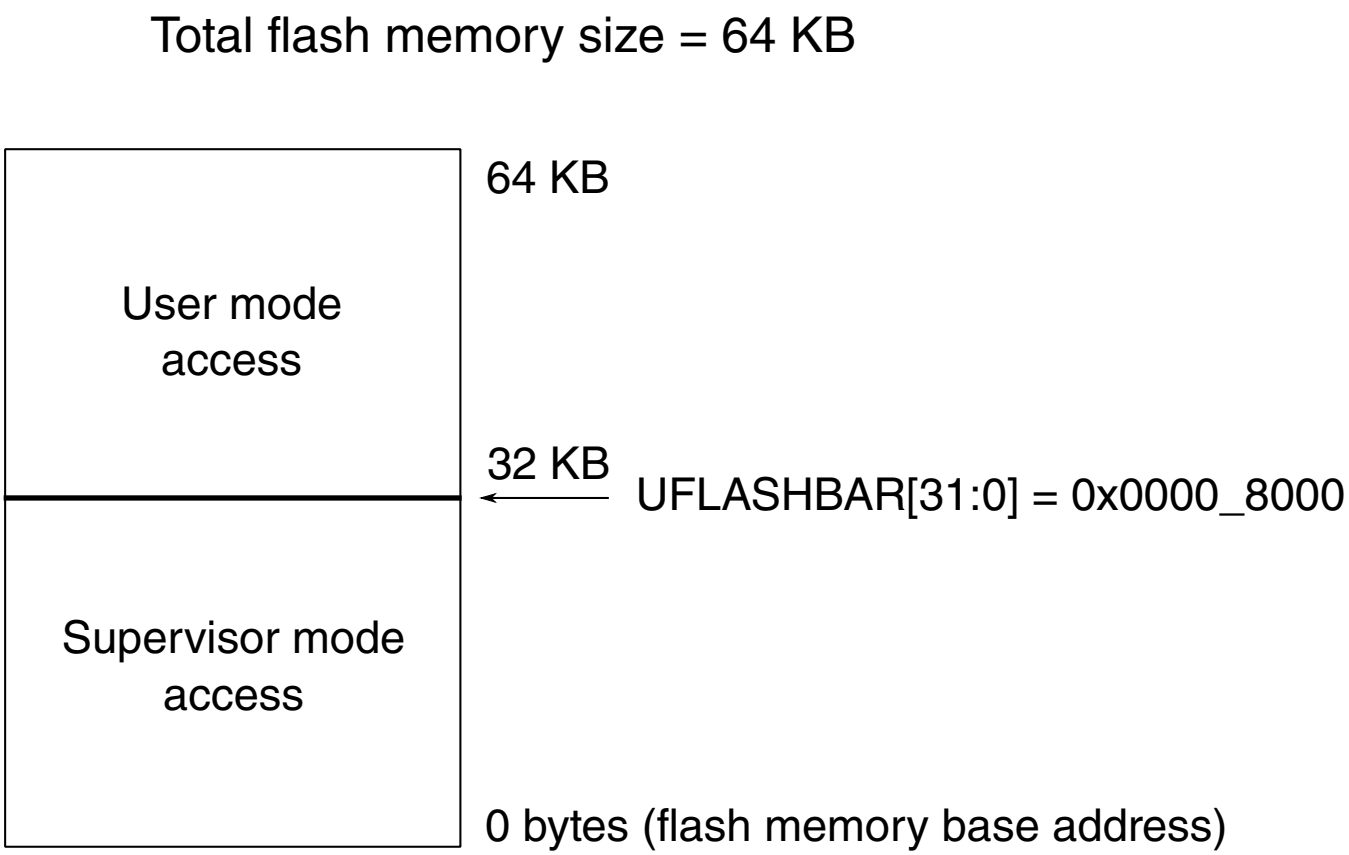


Figure 7-2. Flash Memory Base Address Setup Example

UPRAMBAR Example

If resource protection is enabled and the RAM size is 32 KB (16 KW), setting the UPRAMBAR to 4000h defines the supervisor access region to be 16 KB. As a result, for both the program memory map and the data memory map, the supervisor access region occupies the bottom half of the RAM space and the user access region occupies the upper 16 KB of the space.

- Program (PDB bus) accesses to RAM use the program memory map.
- Data (XAB1 or XAB2 bus) accesses to RAM use the data memory map.

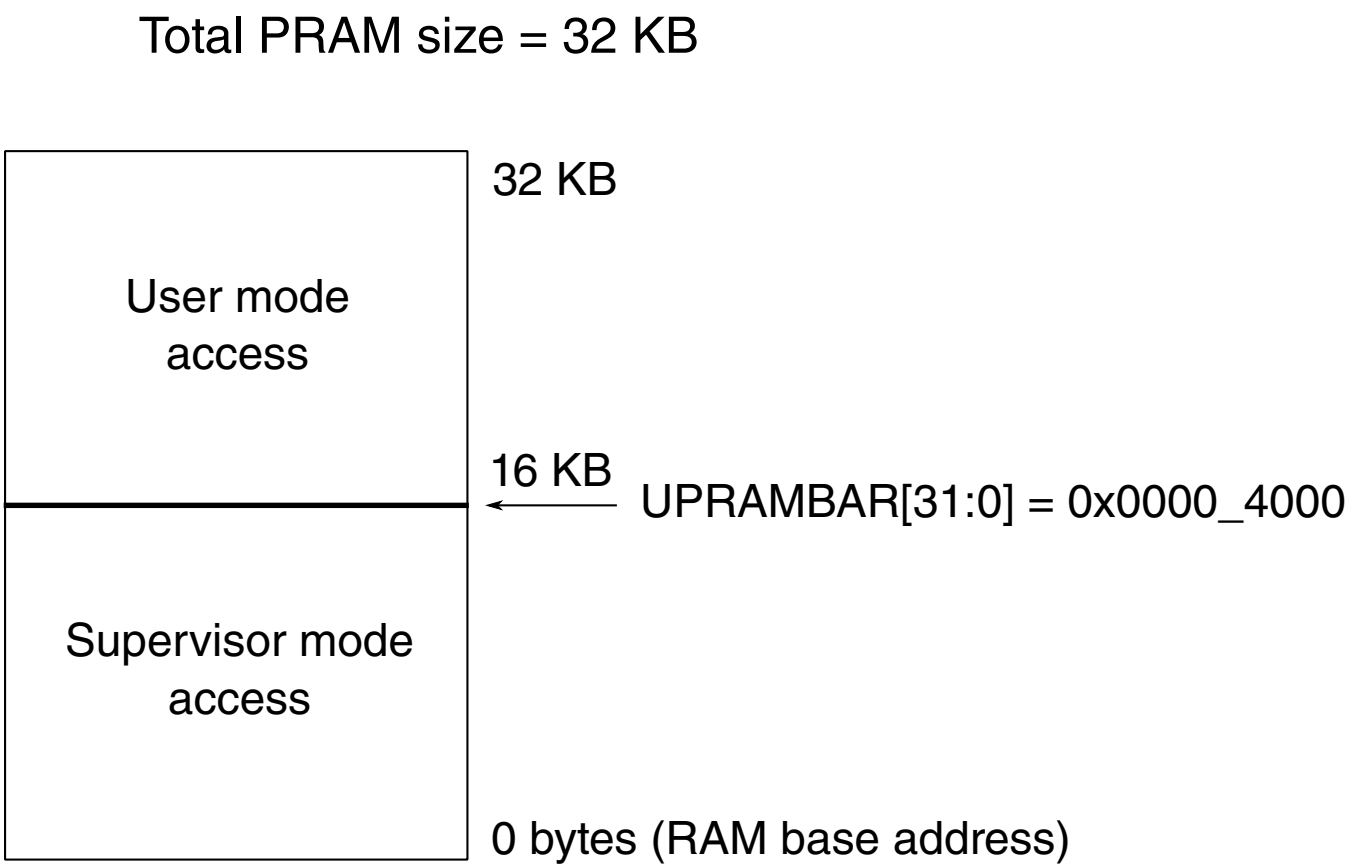


Figure 7-3. RAM Base Address Setup Example

NOTE

Similar case also applies to UBROMBAR setup.

7.7 Programming Example

To set up MRP and enter user mode, follow these required steps:

1. Initialize the UFLASHBAR, setting the base address of the user region in flash memory.
2. Initialize the UPRAMBAR, setting the base address of the user region in RAM.
3. Initialize the UBROMBAR, setting the base address of the user region in Boot ROM.
4. Set up the user stack pointer via the MCM_SRPOSP register.
5. Enable resource protection using the MCM_RPCR[RPE] bit.
6. Push the desired user Status Register (SR) value and user Program Counter (PC) on the supervisor stack.
7. Execute a return from interrupt instruction (RTI, RTID, or FRTID), which reads the user Status Register (SR) value and user Program Counter (PC) from the supervisor stack to switch from supervisor to user mode.

It is recommended that user control is relinquished by the execution of an SWI instruction, returning control to the supervisor.

Chapter 8
Miscellaneous Control Module (MCM)

8.1 Introduction

The Miscellaneous Control Module (MCM) provides a myriad of miscellaneous control functions.

8.1.1 Features

The MCM provides the following:

- Program-visible information about the configuration and revision of the core and select system peripherals
- Registers for capturing information about core and core-peripheral bus errors, if enabled
- Control and configuration of memory resource protection (MRP)

8.2 Functional Description

This section describes the functional description of MCM module.

8.2.1 Core Data Fault Recovery Registers

To aid in recovery from certain types of access errors, the MCM module supports a number of registers that capture access address, attribute, and data information on bus cycles terminated with an error response. These registers can then be read during the resulting exception service routine and the appropriate recovery performed.

The details on the core fault recovery registers are provided in the above sections. It is important to note these registers are used to capture fault recovery information on any processor-initiated system bus cycle terminated with an error.

8.3 Memory Map/Register Descriptions

8.3.1 MCM register descriptions

Restriction

The MCM is a supervisor-only space. All MCM registers must be accessed by supervisor code.

In addition, each MCM register must be written in an access size equal to the register's width. For example, a 32-bit register must be written using a 32-bit access.

NOTE

The base address and offsets for these registers are presented in terms of bytes.

8.3.1.1 MCM memory map

MCM base address: 1_8000h

Offset	Register	Width (In bits)	Access	Reset value
8h	Crossbar switch (AXBS) slave configuration (PLASC)	16	RO	000Fh
Ah	Crossbar switch (AXBS) master configuration (PLAMC)	16	RO	000Fh
Ch	Core platform control register (CPCR)	32	RW	0000_0000h
10h	Core fault address register (CFADR)	32	RO	Table 8-
14h	Core fault attributes register (CFATR)	8	RO	Table 8-
15h	Core fault location register (CFLOC)	8	RO	00h
16h	Core fault interrupt enable register (CFIER)	8	RW	00h
17h	MCM interrupt status register (CFISR)	8	W1C	00h
18h	Core fault data register (CFDTR)	32	RO	Table 8-
20h	Resource Protection Control Register (RPCR)	32	RW	0000_0000h
24h	User Flash Base Address Register (UFLASHBAR)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
28h	User Program RAM Base Address Register (UPRAMBAR)	32	RW	0000_0000h
2Ch	User Boot ROM Base Address Register (UBROMBAR)	32	RW	0000_0000h
30h	Resource Protection Other Stack Pointer (SRPOSP)	32	RW	0000_0000h
34h	Memory Protection Illegal PC (SRPIPC)	32	W1C	0000_0000h
38h	Resource Protection Misaligned PC (SRPMPC)	32	W1C	0000_0000h

8.3.1.2 Crossbar switch (AXBS) slave configuration (PLASC)

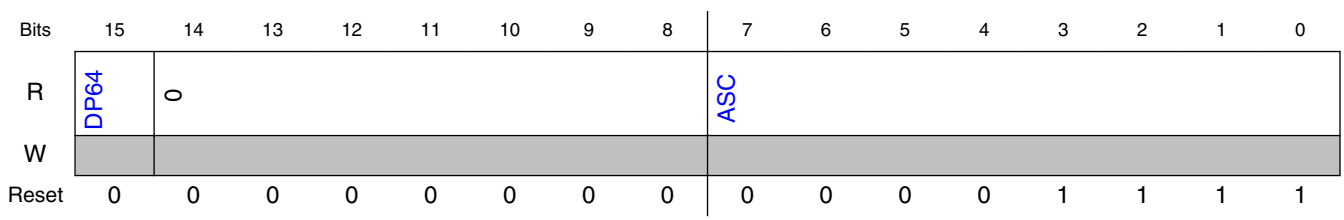
8.3.1.2.1 Offset

Register	Offset
PLASC	8h

8.3.1.2.2 Function

The PLASC is a 16-bit read-only register identifying the presence/absence of bus slave connections to the device's Crossbar Switch (AXBS), plus a 1-bit flag defining the internal data-path width (DP64). The state of this register is defined by a module input signal; it can only be read from the programming model. Any attempted write is ignored.

8.3.1.2.3 Diagram



8.3.1.2.4 Fields

Field	Function
15 DP64	Indicates if the platform data-path is 32 or 64 bits wide 0b - Data-path width is 32 bits 1b - Data-path width is 64 bits

Table continues on the next page...

Field	Function
14-8 —	Reserved
7-0 ASC	Each bit in the ASC field indicates if there is a corresponding connection to the AXBS slave input port. For this device, this field always read 0x0F. 0000_0000b - A bus slave connection to AXBS input port *n* is absent 0000_0001b - A bus slave connection to AXBS input port *n* is present

8.3.1.3 Crossbar switch (AXBS) master configuration (PLAMC)

8.3.1.3.1 Offset

Register	Offset
PLAMC	Ah

8.3.1.3.2 Function

The PLAMC is a 16-bit read-only register identifying the presence/absence of bus master connections to the device's Crossbar Switch (AXBS). The state of this register is defined by a module input signal; it can only be read from the programming model. Any attempted write is ignored.

8.3.1.3.3 Diagram

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

8.3.1.3.4 Fields

Field	Function
15-8 —	Reserved
7-0 AMC	Each bit in the AMC field indicates if there is a corresponding connection to the AXBS master input port. For this device, this field always reads 0x0F. 0000_0000b - A bus master connection to AXBS input port *n* is absent 0000_0001b - A bus master connection to AXBS input port *n* is present

8.3.1.4 Core platform control register (CPCR)

8.3.1.4.1 Offset

Register	Offset
CPCR	Ch

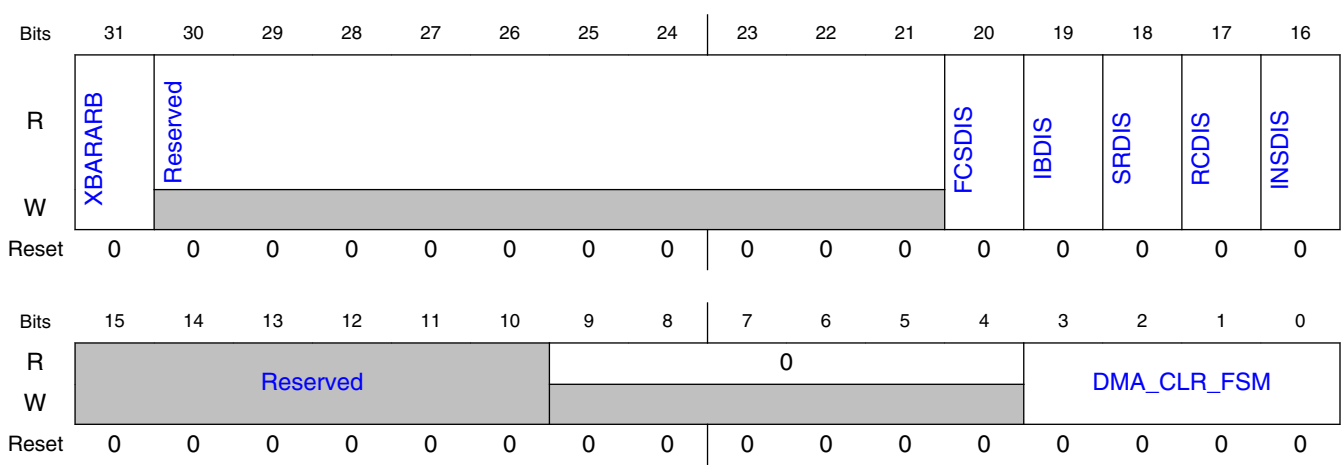
8.3.1.4.2 Function

The 32-bit CPCR provides a program-visible register for user-defined control functions. Typically it controls the configuration of various chip-level modules. The lower word of this register is output from the MCM to other modules where the user-defined control functions are implemented. The upper word of this register is used to control core functions.

Restriction

This register must be written in a 32-bit access to change the core configuration.

8.3.1.4.3 Diagram



8.3.1.4.4 Fields

Field	Function
31 XBARARB	Select DMA Controller priority in AXBS Crossbar Switch arbitration scheme Set the priority of the DMA Controller in the AXBS Crossbar Switch arbitration scheme. This device has 2 bus masters connected to the AXBS Crossbar Switch: the DMA Controller and the DSC core. For more information about AXBS Crossbar Switch arbitration, see Arbitration . 0b - Fixed-priority arbitration is selected: DSC core has a higher priority than the DMA Controller's priority 1b - Round-robin priority arbitration is selected: DMA Controller and DSC core have equal priority
30-21 —	This read-only bitfield is reserved and is reset to zero. Do not write to this bitfield (write only zeros) or indeterminate results will occur.
20 FCSDIS	Disable Flash Memory Controller stall Disables the Flash Memory Controller's ability to allow a flash memory access to initiate when a flash memory command is executing. 0b - Stall logic is enabled. While a flash memory command is executing, a flash memory access can occur without causing a bus error. The flash memory command completes execution, and then the flash memory access occurs. 1b - Stall logic is disabled. While a flash memory command is executing, an attempted flash memory access causes a bus error.
19 IBDIS	Disable core instruction buffer 0b - Core long-word instruction buffer enabled 1b - Core long-word instruction buffer disabled
18 SRDIS	Disable core new shadow region When this bit is 1, only the AGU shadow registers supported by the DSP56800E core are enabled. When this bit is 0, the additional AGU shadow registers on the DSP56800EX core are also enabled. 0b - Core new shadow region enabled 1b - Core new shadow region disabled
17 RCDIS	Disable core reverse carry When this bit is 0, the core supports bit-reverse addressing mode. When this bit is 1, the core does not support this mode. 0b - Core reverse carry enabled 1b - Core reverse carry disabled
16 INSDIS	Disable instructions supported only by DSP56800EX core The instructions supported only by the DSP56800EX core are the BFSC and 32-bit multiply and MAC instructions. 0b - BFSC and 32-bit multiply and MAC instructions enabled 1b - BFSC and 32-bit multiply and MAC instructions disabled
15-10 —	Reserved
9-4 —	Reserved
3-0 DMA_CLR_FSM	Clear FSM for DMA Control bits to clear FSM for channels 0-3 of the DMA slow to fast gasket. **NOTE:** Do not write to this bitfield (write only zeros) or indeterminate results will occur.

8.3.1.5 Core fault address register (CFADR)

8.3.1.5.1 Offset

Register	Offset
CFADR	10h

8.3.1.5.2 Function

The CFADR is a read-only register indicating the address of the last core access terminated with an error response.

NOTE

This register is not initialized at reset, so its reset value is unknown.

8.3.1.5.3 Diagram

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

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

8.3.1.5.4 Fields

Field	Function
31-0 ADDR	Indicates the faulting address of the last core access terminated with an error response.

8.3.1.6 Core fault attributes register (CFATR)

8.3.1.6.1 Offset

Register	Offset
CFATR	14h

8.3.1.6.2 Function

The read-only CFATR register captures the processor's attributes of the last faulted core access to the system bus.

NOTE

This register is not initialized at reset, so its reset value is unknown.

8.3.1.6.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	DIR	SIZE			0	BUFFER	1	TYPE
W								
Reset	u	u	u	u	u	u	u	u

8.3.1.6.4 Fields

Field	Function
7 DIR	Direction of last faulted core access 0b - Core read access 1b - Core write access
6-4 SIZE	Size of last faulted core access 000b - 8-bit 001b - 16-bit 010b - 32-bit
3 —	Reserved
2 BUFFER	Indicates if last faulted core access was bufferable 0b - Non-bufferable 1b - Bufferable
1	Reserved

Table continues on the next page...

Field	Function
—	
0 TYPE	Type of last faulted core access 0b - Instruction 1b - Data

8.3.1.7 Core fault location register (CFLOC)

8.3.1.7.1 Offset

Register	Offset
CFLOC	15h

8.3.1.7.2 Function

The read-only CFLOC register indicates the location of the last captured fault.

8.3.1.7.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	LOC		0					
W								
Reset	0	0	0	0	0	0	0	0

8.3.1.7.4 Fields

Field	Function
7-6 LOC	Location of last captured fault 00b - Error occurred on M0 (instruction bus) 01b - Error occurred on M1 (operand A bus) 10b - Error occurred on M2 (operand B bus) 11b - Reserved
5-0 —	Reserved

8.3.1.8 Core fault interrupt enable register (CFIER)

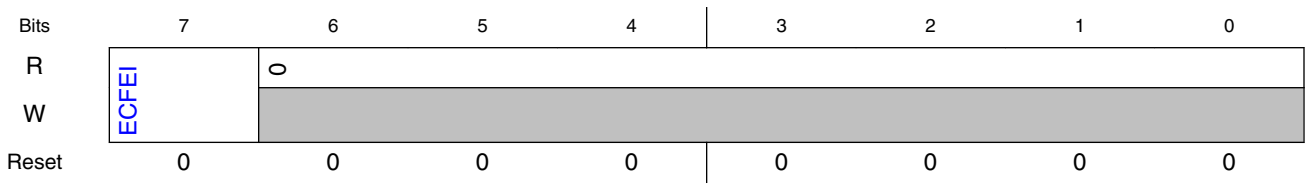
8.3.1.8.1 Offset

Register	Offset
CFIER	16h

8.3.1.8.2 Function

The CFIER register enables the system bus-error interrupt.

8.3.1.8.3 Diagram



8.3.1.8.4 Fields

Field	Function
7 ECFEI	Enable core fault error interrupt 0b - Do not generate an error interrupt on a faulted system bus cycle 1b - Generate an error interrupt to the interrupt controller on a faulted system bus cycle
6-0 —	Reserved

8.3.1.9 MCM interrupt status register (CFISR)

8.3.1.9.1 Offset

Register	Offset
CFISR	17h

8.3.1.9.2 Function

This register indicates if a core fault interrupt has occurred.

8.3.1.9.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	CFEI	CFEDL	0					
W	W1C	W1C						
Reset	0	0	0	0	0	0	0	0

8.3.1.9.4 Fields

Field	Function
7 CFEI	Core fault error interrupt flag Indicates if a bus fault has occurred. Writing a 1 clears this bit and negates the interrupt request. Writing a 0 has no effect. **NOTE:** This bit reports core faults regardless of the setting of CFIER[ECFEI]. Therefore, if the error interrupt is disabled and a core fault occurs, this bit is set. Then, if the error interrupt is subsequently enabled, an interrupt is immediately requested. To prevent an undesired interrupt, clear the captured error by writing one to CFEI before enabling the interrupt. 0b - No bus error 1b - A bus error has occurred. The faulting address, attributes (and possibly write data) are captured in the CFADR, CFATR, and CFDTR registers. The error interrupt is enabled only if CFIER[ECFEI] is set.
6 CFEDL	Core fault error data lost flag Indicates that the address, attribute and data registers have been reloaded in response to a bus error before the previous error data was retrieved. 0b - No bus error data lost 1b - A bus error has occurred before the previous error condition was cleared.
5-0 —	Reserved

8.3.1.10 Core fault data register (CFDTR)

8.3.1.10.1 Offset

Register	Offset
CFDTR	18h

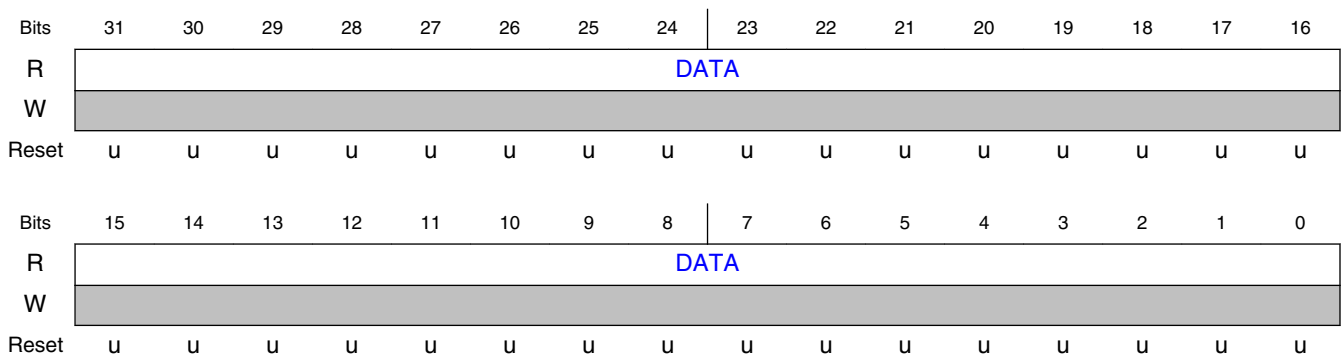
8.3.1.10.2 Function

The CFDTR is a read-only register for capturing the data associated with the last faulted processor write data access from the device's internal bus. The CFDTR is valid only for faulted internal bus-write accesses; CFLOC[LOC] is cleared.

NOTE

This register is not initialized at reset, so its reset value is unknown.

8.3.1.10.3 Diagram



8.3.1.10.4 Fields

Field	Function
31-0 DATA	Contains write data associated with the faulting access of the last internal bus write access. The data value is taken directly from the write data bus. Read data is not captured.

8.3.1.11 Resource Protection Control Register (RPCR)

8.3.1.11.1 Offset

Register	Offset
RPCR	20h

8.3.1.11.2 Function

This register enables/disables memory resource protection (MRP) and locks/unlocks the values of the RP-related registers.

The DSC core provides resource protection functionality. Refer to the detailed MRP description for more information about how to use the RPCR and other RP registers.

NOTE

The following write accesses to the resource protection registers are ignored and result in a bus error:

- Any non-32-bit write
- Any attempted write when resource protection hardware features are not enabled
- Any attempted write when RPCR[RL] is set (reads are allowed when RPCR[RL] is set)

The bus error interrupt must be enabled; otherwise, the errors are ignored by the DSC core.

8.3.1.11.3 Diagram

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

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

8.3.1.11.4 Fields

Field	Function
31-2 —	Reserved
1 RL	Register Lock This bit controls whether the values of the UFLASHBAR, UPRAMBAR, SRPOSP, SRPIPC, and SRPMPC registers can be modified. 0b - RP register values may be changed 1b - RP registers are locked and may not be changed until after a system reset
0	Resource Protection Enable

Field	Function
RPE	0b - Resource protection disabled 1b - Resource protection enabled

8.3.1.12 User Flash Base Address Register (UFLASHBAR)

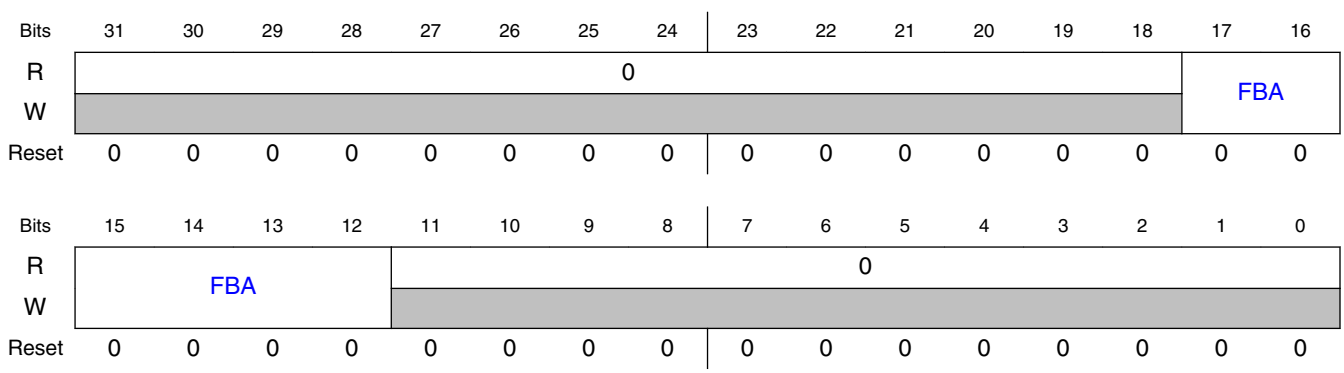
8.3.1.12.1 Offset

Register	Offset
UFLASHBAR	24h

8.3.1.12.2 Function

This register defines the size of the portion of flash memory that is used for supervisor space when resource protection is enabled. The register can be used only when the RPCR[RPE] bit is 1, and its value can be changed only when the RPCR[RL] bit is 0.

8.3.1.12.3 Diagram



8.3.1.12.4 Fields

Field	Function
31-18 —	Reserved
17-12 FBA	Flash Base Address for User Region Supports 4 KB granularity
11-0 —	Reserved

8.3.1.13 User Program RAM Base Address Register (UPRAMBAR)

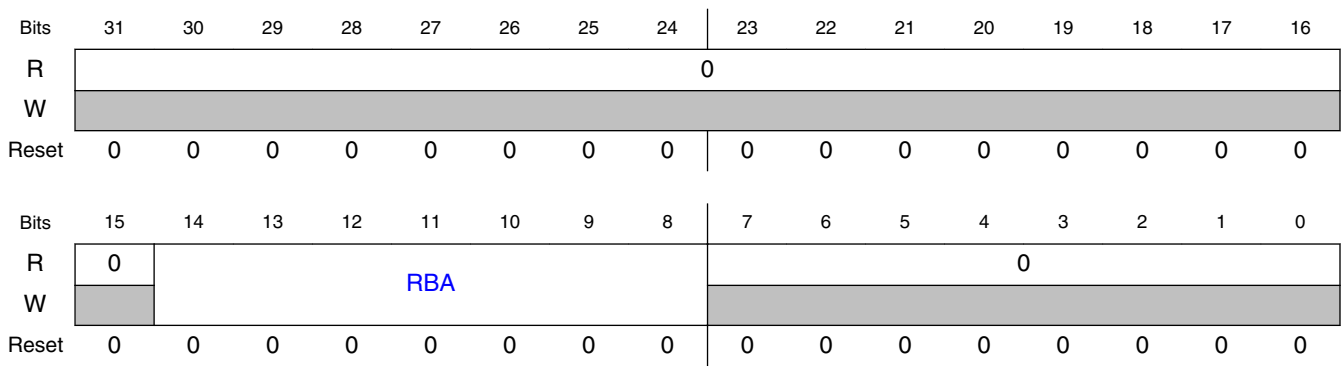
8.3.1.13.1 Offset

Register	Offset
UPRAMBAR	28h

8.3.1.13.2 Function

This register defines the size of the portion of program RAM that is used for supervisor space when resource protection is enabled. The register can be used only when the RPCR[RPE] bit is 1, and its value can be changed only when the RPCR[RL] bit is 0.

8.3.1.13.3 Diagram



8.3.1.13.4 Fields

Field	Function
31-15 —	Reserved
14-8 RBA	Program RAM Base Address for User Region Supports 256 byte granularity
7-0 —	Reserved

8.3.1.14 User Boot ROM Base Address Register (UBROMBAR)

8.3.1.14.1 Offset

Register	Offset
UBROMBAR	2Ch

8.3.1.14.2 Function

This register defines the size of the portion of boot ROM that is used for supervisor space when resource protection is enabled. The register can be used only when the RPCR[RPE] bit is 1, and its value can be changed only when the RPCR[RL] bit is 0.

8.3.1.14.3 Diagram

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

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

8.3.1.14.4 Fields

Field	Function
31-0 RBA	RBA Boot ROM Base Address for User Region

8.3.1.15 Resource Protection Other Stack Pointer (SRPOSP)

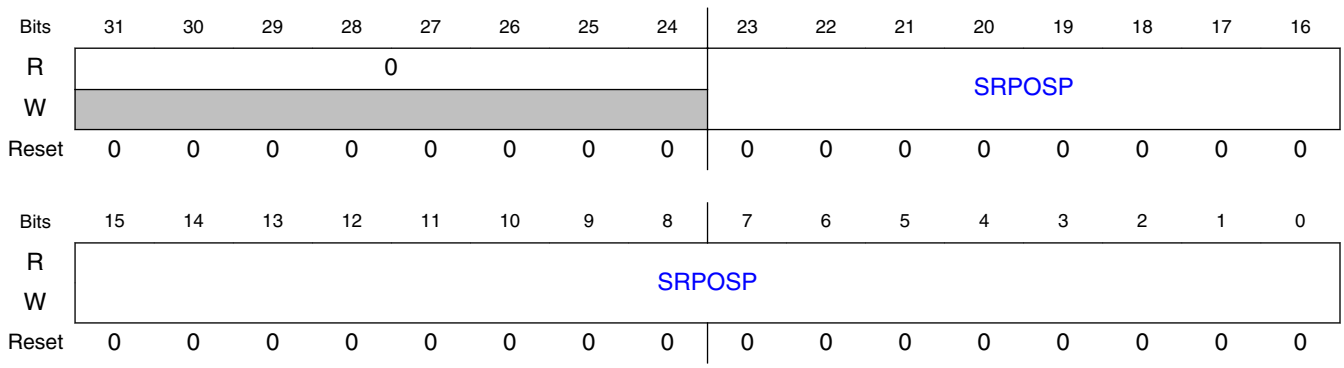
8.3.1.15.1 Offset

Register	Offset
SRPOSP	30h

8.3.1.15.2 Function

This register can be used only when the RPCR[RPE] bit is 1, and its value can be changed only when the RPCR[RL] bit is 0.

8.3.1.15.3 Diagram



8.3.1.15.4 Fields

Field	Function
31-24 —	Reserved
23-0 SRPOSP	Resource protection "other" SP

8.3.1.16 Memory Protection Illegal PC (SRPIPC)

8.3.1.16.1 Offset

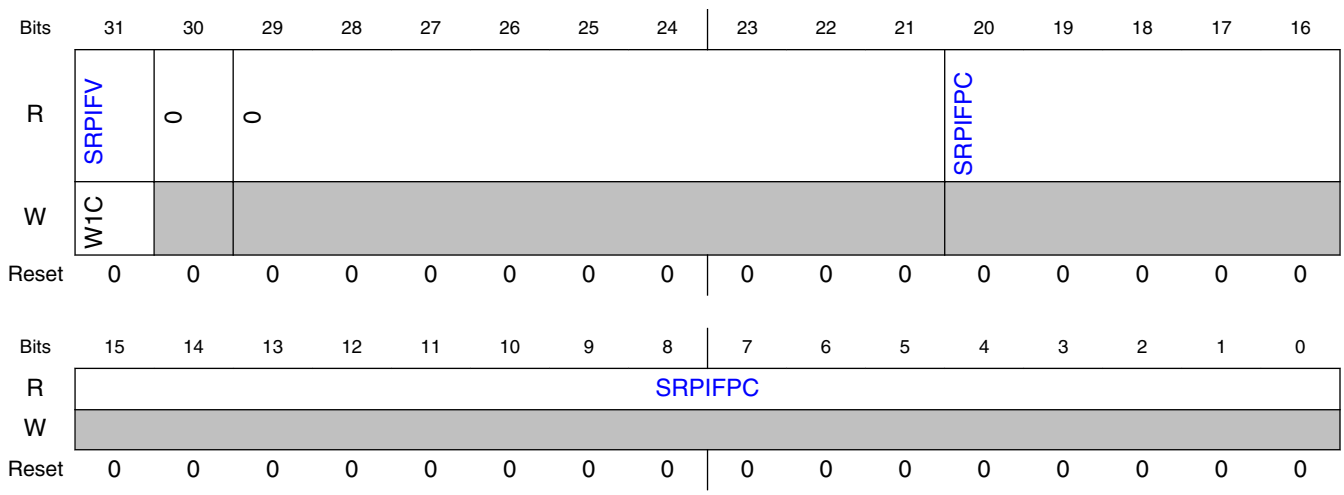
Register	Offset
SRPIPC	34h

8.3.1.16.2 Function

This register can be used only when the RPCR[RPE] bit is 1, and its value can be changed only when the RPCR[RL] bit is 0.

This register's fault indicators apply only to faults resulting from supervisor and user access errors when resource protection is in effect. Other faults are not indicated in this register.

8.3.1.16.3 Diagram



8.3.1.16.4 Fields

Field	Function
31 SRPIFV	Resource Protection Illegal Fault Valid When set, this bit indicates an RP illegal PC fault has occurred and the contents of SRPIFPC and SRPIFOR fields are valid. A write of 1 to this bit clears both the SRPIFOR and SRPIFV bits.
30 —	Reserved
29-21 —	Reserved
20-0 SRPIFPC	Resource Protection Illegal Faulting PC This is the 21-bit illegal faulting PC only for a resource protection fault.

8.3.1.17 Resource Protection Misaligned PC (SRPMPC)

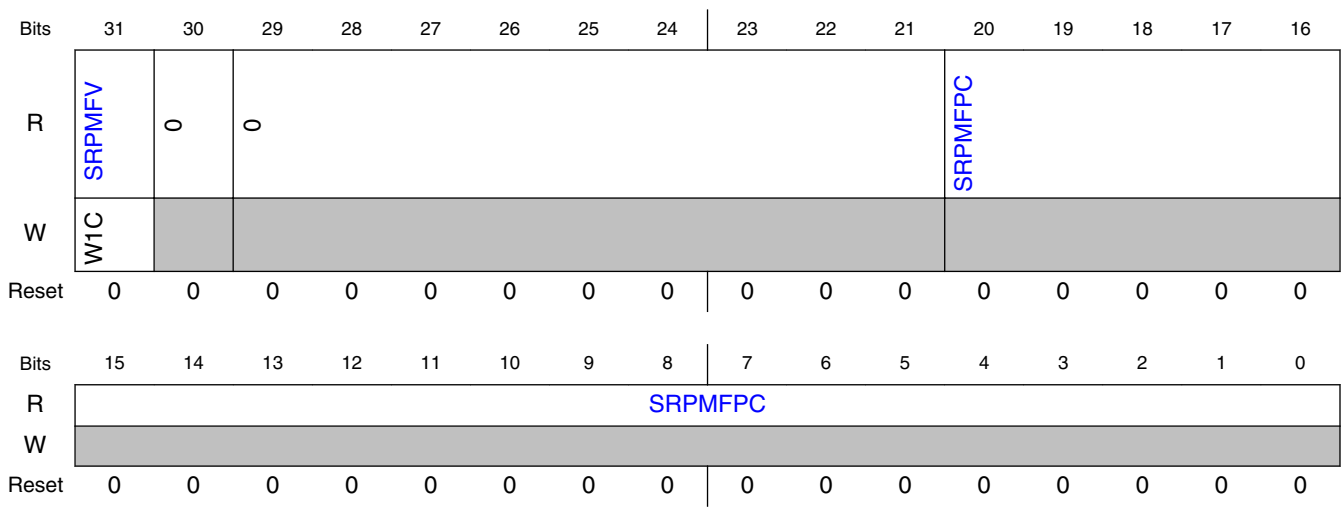
8.3.1.17.1 Offset

Register	Offset
SRPMPC	38h

8.3.1.17.2 Function

This register's fault indicators apply only to faults resulting from supervisor and user access errors when resource protection is in effect. Other faults are not indicated in this register.

8.3.1.17.3 Diagram



8.3.1.17.4 Fields

Field	Function
31 SRPMFV	Resource Protection Misaligned Fault Valid When set, this bit indicates an RP misaligned PC fault has occurred and the contents of the SRPMFPC and SRPMFOR fields are valid. A write of 1 to this bit clears both the SRPMFOR and SRPMFV bits.
30 —	Reserved
29-21 —	Reserved

Table continues on the next page...

Field	Function
20-0 SRPMFPC	Resource Protection Misaligned Faulting PC This is the 21-bit misaligned faulting PC only for a resource protection data access fault.

Chapter 9
System Integration Module (SIM)

9.1 Introduction

This specification describes the operation and functionality of the System Integration Module for this device.

9.1.1 Overview

The System Integration Module (SIM) provides a variety of system control and status functions.

The system integration module is responsible for the following control/status functions:

- Reset control and status
- User accessible Software Control Registers which reset only at power on.
- Internal clock generation
- Implementation of STOP and WAIT low power modes and related clock gating
- System status registers
- Registers for software access to the JTAG ID of the chip and suggested trim values set at the factory
- Short addressing controls
- Test registers
- External and internal peripheral signal muxing control

9.1.2 Features

The SIM interacts with a variety of other on-chip resources and provides the following services:

- Controls and sequences the release of internal reset signals.
- Generates pipelined system clocks including management of holdoffs and core stalls.
- Generates peripheral clocks configurable over power modes.

- Manages low power mode entry and exit.
- Provides clock rate controls for selected peripherals.
- Selects the active peripheral function for IO when it is not used as GPIO.
- Provides write protection for safety critical memory mapped registers.
- Controls certain DSC core functions, including the base for short addressing mode and enablement of the test access port (TAP) and debug mode entry as well as core low power modes.
- Controls voltage regulator.
- Selects CLKOUT clock source.
- Controls inter-peripheral muxing and signal relationships.

9.1.3 Modes of Operation

Since the SIM is responsible for distributing clocks and resets across the chip, it must understand the various chip operating modes and take appropriate action. These include:

- **RESET Modes:**This is the sequencing of reset deassertion as part comes out of reset.
  - **Clock Reset Mode**: The DSC processor, all peripherals, and the CLKGEN module are all in reset.
  - **System and Core Reset Mode:** Clocks activate while the DSC core and peripherals remain in reset.
  - **Core-Only Reset Mode:**DSC core in reset while peripherals are activated.

NOTE

Core-only reset mode is required to provide time for the on-chip flash interface units to load part configuration data from flash and establish the boot address.

- **RUN Mode:** This is the primary mode of operation for this device. In this mode, the processor clocks, system clocks, and all enabled peripheral clocks are operational.
- **Debug Mode:** The DSC processor is in debug mode (controlled via JTAG/EOnCE). All system and peripheral clocks with the exception of the COP and PWM's continue to run. The COP is disabled in debug mode and PWM outputs are optionally disabled (see the PWM details) to bypass undesired motor control operations.
- **WAIT Mode:** In WAIT mode, the core clock and system clocks are disabled but all enabled peripheral clocks continue to operate. The COP can optionally be stopped in WAIT mode. Similarly, the PWM outputs can optionally be switched off in WAIT mode to disable any motor from being driven.
- **STOP Mode:** In STOP mode, the core clock and system clocks are disabled. Individual peripheral clocks are also disabled unless specifically configured in the SIM to continue running in STOP mode (useful for generating STOP recovery

interrupts from selected devices). The COP can optionally be stopped in STOP mode. If desired, the OCCS may be configured to disable the PLL and/or select a lower frequency clock source prior to entering STOP mode.

When eDMA is enabled in STOP mode (via SIM_CTRL[DMAEBL] set to 011b or 111b), setting bits to logic 0s in Peripheral Clock STOP Disable Registers (SIM_SD0 ~ SIM_SD3) may not immediately gate off the clock to corresponding peripherals when STOP instruction is executed. This prevents the data loss when the DMA is transferring the data while a STOP instruction is executed. So the peripherals (enabled in SIM_PCE registers) are clocked till a DMA major loop is completed after the execution of a STOP instruction, then clocks are gated off. Also, SIM_CTRL[DMAEBL] is 011b after reset, if there is no DMA enabled at all and a STOP instruction is executed, peripherals (enabled in SIM_PCE registers) remain clocked until the completion of a DMA transfer.

NOTE

The power management controller (PMC) provides additional control of power consumption by supporting low-power states with reduced regulator drive capacity. The on-chip clock system (OCCS) module also influences power consumption by controlling the operating frequency of the system and peripheral clocks and by supporting the powering down of unused clock sources.

9.2 Memory Map and Register Descriptions

SIM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E400	Control Register (SIM_CTRL)	16	R/W	00C0h	9.2.1/165
E401	Reset Status Register (SIM_RSTAT)	16	R	0004h	9.2.2/167
E406	Most Significant Half of JTAG ID (SIM_MSHID)	16	R	See section	9.2.3/168
E407	Least Significant Half of JTAG ID (SIM_LSHID)	16	R	A61Dh	9.2.4/169
E408	Power Control Register (SIM_PWR)	16	R/W	0000h	9.2.5/170
E40A	Clock Output Select Register (SIM_CLKOUT)	16	R/W	1020h	9.2.6/172
E40B	Peripheral Clock Rate Register (SIM_PCR)	16	R/W	0400h	9.2.7/173
E40C	Peripheral Clock Enable Register 0 (SIM_PCE0)	16	R/W	0000h	9.2.8/175
E40D	Peripheral Clock Enable Register 1 (SIM_PCE1)	16	R/W	0000h	9.2.9/177

Table continues on the next page...

SIM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E40E	Peripheral Clock Enable Register 2 (SIM_PCE2)	16	R/W	0000h	9.2.10/178
E40F	Peripheral Clock Enable Register 3 (SIM_PCE3)	16	R/W	1200h	9.2.11/180
E410	Peripheral Clock STOP Disable Register 0 (SIM_SD0)	16	R/W	0000h	9.2.12/181
E411	Peripheral Clock STOP Disable Register 1 (SIM_SD1)	16	R/W	0000h	9.2.13/183
E412	Peripheral Clock STOP Disable Register 2 (SIM_SD2)	16	R/W	0000h	9.2.14/185
E413	Peripheral Clock STOP Disable Register 3 (SIM_SD3)	16	R/W	0000h	9.2.15/187
E414	I/O Short Address Location Register (SIM_IOSAHI)	16	R/W	0000h	9.2.16/189
E415	I/O Short Address Location Register (SIM_IOSALO)	16	R/W	0380h	9.2.17/190
E416	Protection Register (SIM_PROT)	16	R/W	0000h	9.2.18/191
E417	GPIOA LSBs Peripheral Select Register (SIM_GPSAL)	16	R/W	0000h	9.2.19/193
E418	GPIOB LSBs Peripheral Select Register (SIM_GPSBL)	16	R/W	0000h	9.2.20/194
E419	GPIOC LSBs Peripheral Select Register (SIM_GPSCL)	16	R/W	0000h	9.2.21/195
E41A	GPIOC MSBs Peripheral Select Register (SIM_GPSCH)	16	R/W	0000h	9.2.22/196
E41C	GPIOE LSBs Peripheral Select Register (SIM_GPSEL)	16	R/W	0000h	9.2.23/197
E41E	GPIOF LSBs Peripheral Select Register (SIM_GPSFL)	16	R/W	0000h	9.2.24/199
E41F	GPIOF MSBs Peripheral Select Register (SIM_GPSFH)	16	R/W	0000h	9.2.25/200
E422	Internal Peripheral Select Register 0 (SIM_IPS0)	16	R/W	0000h	9.2.26/200
E423	Miscellaneous Register 0 (SIM_MISC0)	16	R/W	0000h	9.2.27/202
E424	Peripheral Software Reset Register 0 (SIM_PSWR0)	16	R/W	0000h	9.2.28/203
E425	Peripheral Software Reset Register 1 (SIM_PSWR1)	16	R/W	0000h	9.2.29/204
E426	Peripheral Software Reset Register 2 (SIM_PSWR2)	16	R/W	0000h	9.2.30/206
E427	Peripheral Software Reset Register 3 (SIM_PSWR3)	16	R/W	0000h	9.2.31/207
E428	Power Mode Register (SIM_PWRMODE)	16	R/W	0000h	9.2.32/209
E445	Software Control Register (SIM_SCR0)	16	R/W	0000h	9.2.33/210
E446	Software Control Register (SIM_SCR1)	16	R/W	0000h	9.2.34/210
E447	Software Control Register (SIM_SCR2)	16	R/W	0000h	9.2.35/211
E448	Software Control Register (SIM_SCR3)	16	R/W	0000h	9.2.36/211
E449	Software Control Register (SIM_SCR4)	16	R/W	0000h	9.2.37/211
E44A	Software Control Register (SIM_SCR5)	16	R/W	0000h	9.2.38/212
E44B	Software Control Register (SIM_SCR6)	16	R/W	0000h	9.2.39/212
E44D	ADC and TMR Select Register (SIM_ADC_TMR_SEL)	16	R/W	0000h	9.2.40/212

9.2.1 Control Register (SIM_CTRL)

Address: E400h base + 0h offset = E400h

Bit	15	14	13	12	11	10	9	8
Read	0					RST_FILT	0	DMAEBL
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	DMAEBL		ONCEEBL	SWRST	STOP_DISABLE		WAIT_DISABLE	
Write								
Reset	1	1	0	0	0	0	0	0

SIM_CTRL field descriptions

Field	Description
15–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 RST_FILT	External Reset Padcell Input Filter Enable This input controls an optional analog input filter on the padcell supporting the external reset input function. When enabled, the filter removes transient signals on the input at the expense of an increased input delay. When enabled, the filter affects all input functions supported by that padcell, including GPIO. This bit is reset on POR only. The filter has two basic applications. When this pad is configured as an output function such as a GPIO output and the part is reset, the filter can remove transients that might result in a false indication of an external reset assertion in the SIM's RSTAT register. When this padcell is configured as the external reset input, the filter can filter out noise on the external reset to reduce the chance of an unintended external reset assertion. The filter delay should be considered before enabling the filter, especially for safety critical applications. 0 Input filter on external reset disabled 1 Input filter on external reset enabled
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

8–6
DMAEBL
	DMA Enable
	
	This field controls whether DMA is enabled in RUN, WAIT, RUN and WAIT, or all modes. The DMA functions as a system bus master capable of performing IO on any address in data space. It also contains memory mapped registers through which it is configured. The DMA must be enabled to function either as a system bus master or to perform IO to its memory mapped registers.
	
	NOTE: The DMA incorporates synchronous reset logic and must therefore be powered on at reset. It may subsequently be powered down at any time when not in use.
	
	NOTE: Entry to stop mode affects in-progress DMA transactions differently than does entry to wait mode:
	• When the DMA module is configured to be disabled in stop mode: If any DMA transaction is in progress during stop mode entry, the transaction will complete before the MCU gates the DMA controller's clock.
	• When the DMA module is configured to be disabled in wait mode:
	  • If any DMA transaction is in progress during wait mode entry, the transaction might be incomplete and end prematurely when the MCU gates the DMA controller's clock.
	  • To avoid data loss, the user's application must ensure no DMA transaction is in progress during the entry to wait mode.

Table continues on the next page...

SIM_CTRL field descriptions (continued)

Field	Description
	000 DMA module is disabled. 001 DMA module is enabled in run mode only. 010 DMA module is enabled in run and wait modes only. 011 DMA module is enabled in all power modes. 100 DMA module is disabled and the DMAEBL field is write protected until the next reset. 101 DMA module is enabled in run mode only and the DMAEBL field is write protected until the next reset. 110 DMA module is enabled in run and wait modes only and the DMAEBL field is write protected until the next reset. 111 DMA module is enabled in all low power modes and the DMAEBL field is write protected until the next reset.
5 ONCEEBL	OnCE Enable 0 The OnCE clock to the DSC core is enabled when the core TAP is enabled. 1 The OnCE clock to the DSC core is always enabled.
4 SWRST	SOFTWARE RESET Writing a 1 to this field causes a device reset.
3–2 STOP_DISABLE	STOP Disable 00 Stop mode is entered when the DSC core executes a STOP instruction. 01 The DSC core STOP instruction does not cause entry into stop mode. 10 Stop mode is entered when the DSC core executes a STOP instruction, and the STOP_disable field is write protected until the next reset. 11 The DSC core STOP instruction does not cause entry into stop mode, and the STOP_disable field is write protected until the next reset.
WAIT_DISABLE	WAIT Disable 00 Wait mode is entered when the DSC core executes a WAIT instruction. 01 The DSC core WAIT instruction does not cause entry into wait mode. 10 Wait mode is entered when the DSC core executes a WAIT instruction, and the WAIT_disable field is write protected until the next reset. 11 The DSC core WAIT instruction does not cause entry into wait mode, and the WAIT_disable field is write protected until the next reset.

9.2.2 Reset Status Register (SIM_RSTAT)

This register is updated upon any system reset and indicates the cause of the most recent reset. It also controls whether the COP reset vector or regular reset vector in the vector table is used. This register is asynchronously reset during power-on reset and subsequently is synchronously updated based on the level of the external reset, software reset, or COP reset inputs. It is "one-hot encoded," which means that only one reset source is indicated at any given time. When multiple reset sources assert simultaneously, the reset source with the highest precedence is indicated in this register. The precedence from highest to lowest is POR, EXTR, COP_LOR, COP_CPU, and SWR. POR is always set during a power-on reset; however, POR is cleared and EXTR is set if the external reset pin is asserted or remains asserted after the power-on reset has deasserted.

Address: E400h base + 1h offset = E401h

Bit	15	14	13	12	11	10	9	8
Read	0					BOOT_MODE_STATUS		COP_WIN
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	Reserved	SWR	COP_CPU	COP_LOR	EXTR	POR	0	
Write								
Reset	0	0	0	0	0	1	0	0

SIM_RSTAT field descriptions

Field	Description
15–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–9 BOOT_MODE_ STATUS	Boot mode. When both bits are set, indicates ROM boot. Otherwise, NVM Flash boot.
8 COP_WIN	COP Window Time-out Reset When set, this bit indicates that the previous system reset occurred as a result of a cop_window reset . It will not be set if an external reset, POR reset, COP CPU reset or COP loss of reference reset occurred. If COP_WINDOW is set as code starts executing the COP reset vector in the vector table will be used. Otherwise the normal reset vector is used.
7 Reserved	This field is reserved.
6 SWR	Software Reset

Table continues on the next page...

SIM_RSTAT field descriptions (continued)

Field	Description
	When set, this bit indicates that the previous system reset occurred as a result of a software reset (1 written to the SWRst bit of the SIM's CTRL register). SWR is not set if a COP reset, external reset, or POR also occurred.
5 COP_CPU	COP CPU Time-out Reset When set, this bit indicates that the previous system reset was caused when the computer operating properly (COP) module signaled a CPU time-out reset. COP_CPU is not set if an external reset, POR, or COP loss of reference reset also occurred. If COP_CPU is set as code starts executing, the COP reset vector in the vector table is used. Otherwise, the normal reset vector is used.
4 COP_LOR	COP Loss of Reference Reset When set, this bit indicates that the previous system reset was caused when the computer operating properly (COP) module signaled a loss of reference clock reset. COP_LOR is not set if an external or POR also occurred. If COP_LOR is set as code starts executing, the COP reset vector in the vector table is used. Otherwise, the normal reset vector is used.
3 EXTR	External Reset When set, this bit indicates that the previous system reset was caused by an external reset. EXTR is set only if the external reset pin was asserted or remained asserted after the power-on reset deasserted.
2 POR	Power-on Reset This bit is set by a power-on reset.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

9.2.3 Most Significant Half of JTAG ID (SIM_MSHID)

This read-only register returns the most significant half of the JTAG ID for the device.

Address: E400h base + 6h offset = E406h

Bit	15	14	13	12	11	10	9	8
Read	Reserved			0	1	0	1	1
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

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

* Notes:

- 0x0B32 for devices with mask set number 0N91Z or 1N91Z.
- 0xAB32 for devices with mask set number 2N91Z.

x = Undefined at reset.

SIM_MSHID field descriptions

Field	Description
15–13 Reserved	This field is reserved.
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 1.
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 1.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved.
Reserved	This field is reserved.

9.2.4 Least Significant Half of JTAG ID (SIM_LSHID)

This read-only register returns the least significant half of the JTAG ID for the device.

Address: E400h base + 7h offset = E407h

Bit	15	14	13	12	11	10	9	8
Read	Reserved	Reserved	Reserved	Reserved	0	1		0
Write								
Reset	1	0	1	0	0	1	1	0

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

SIM_LSHID field descriptions

Field	Description
15 Reserved	This field is reserved.
14 Reserved	This field is reserved.
13 Reserved	This field is reserved.
12 Reserved	This field is reserved.
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
8–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.

9.2.5 Power Control Register (SIM_PWR)

NOTE

This PWR register does not take effect when any bit in SIM_PWRMODE is set.

This register contains control fields used to enable/disable the standby and powerdown modes of the large and small voltage regulators in the Power Management Controller module. The large regulator supplies digital standard cell core logic. The register independently controls the regulator mode. However, if the flash module's FOPT[0] bit is 1 (reflecting a power mode selection via the Flash Option register), writing to these control bits does not affect the regulators.

Address: E400h base + 8h offset = E408h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								SR12STDBY		SR27PDN		SR27STDBY		LRSTDBY	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_PWR 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 SR12STDBY	Small Regulator 1.2 V Supply Standby Control This field controls the standby mode of the 1.2 V supply from the small voltage regulator. Standby mode has restricted drive capacity but substantially reduces power consumption. The field value can be optionally write protected. 00 Small regulator 1.2 V supply placed in normal mode (default). 01 Small regulator 1.2 V supply placed in standby mode. 10 Small regulator 1.2 V supply placed in normal mode and SR12STDBY is write protected until chip reset. 11 Small regulator 1.2 V supply placed in standby mode and SR12STDBY is write protected until chip reset.
5–4 SR27PDN	Small Regulator 2.7 V Supply Powerdown Control This field controls the powerdown mode of the 2.7 V supply from the small voltage regulator. Powerdown mode shuts down the 2.7 V regulated supply from the small regulator and eliminates its power consumption. Analog modules powered by this supply should themselves be powered down before entering this mode. These controls are located in the OCCS module. 00 Small regulator placed in normal mode (default). 01 Small regulator placed in powerdown mode. 10 Small regulator placed in normal mode and SR27PDN is write protected until chip reset. 11 Small regulator placed in powerdown mode and SR27PDN is write protected until chip reset.
3–2 SR27STDBY	Small Regulator 2.7 V Supply Standby Control This field controls the standby mode of the 2.7 V supply from the small voltage regulator. Standby mode has restricted drive capacity but substantially reduces power consumption. The field value can be optionally write protected. 00 Small regulator 2.7 V supply placed in normal mode (default). 01 Small regulator 2.7 V supply placed in standby mode. 10 Small regulator 2.7 V supply placed in normal mode and SR27STDBY is write protected until chip reset. 11 Small regulator 2.7 V supply placed in standby mode and SR27STDBY is write protected until chip reset.
LRSTDBY	Large Regulator Standby Control This field controls the standby mode of the primary on-device voltage regulator. Standby mode reduces overall device power consumption but places significant constraints on the allowed operating frequency. Refer to the Power Management Controller module's description for details on standby mode. The field value can optionally be write protected. 00 Large regulator placed in normal mode (default). 01 Large regulator placed in standby mode. 10 Large regulator placed in normal mode and LRSTDBY is write protected until device reset. 11 Large regulator placed in standby mode and LRSTDBY is write protected until device reset.

9.2.6 Clock Output Select Register (SIM_CLKOUT)

This register can be used to multiplex selected clocks generated inside the clock generation, SIM, and other internal modules onto the SIM CLKOUT clock output signal. This signal, in turn, is typically available to be brought out to an external pad. Glitches may be produced when the clock is enabled or switched. The delay from the clock source to the output is unspecified. The visibility of the waveform on the CLKOUT on an external pad is subject to the frequency limitations of the associated I/O cell.

Address: E400h base + Ah offset = E40Ah

Bit	15	14	13	12	11	10	9	8
Read Write	CLKODIV			CLKDIS1	Reserved		CLKOSEL1	
Reset	0	0	0	1	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	CLKOSEL1	Reserved	CLKDIS0	Reserved			CLKOSEL0	
Reset	0	0	1	0	0	0	0	0

SIM_CLKOUT field descriptions

Field	Description
15–13 CLKODIV	CLKOUT divide factor Configures dividers on the CLKOUT0 and CLKOUT1 outputs used to reduce output frequencies to levels supported by their respective pad cells. 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
12 CLKDIS1	Disable for CLKOUT1 0 CLKOUT1 output is enabled and outputs the signal indicated by CLKOSEL1 1 CLKOUT1 is disabled
11–10 Reserved	This field is reserved. Always write 0 to this field for normal operation.
9–7 CLKOSEL1	CLKOUT1 Select Selects the clock source for the CLKOUT1 pin. The internal delay to the CLKOUT1 output is unspecified. The signal at the output pad is undefined when the CLKOUT1 signal frequency exceeds the rated frequency of the IO cell. CLKOUT1 may glitch when CLKDIS1, CLKOSEL1, or CLKODIV settings are changed.

Table continues on the next page...

SIM_CLKOUT field descriptions (continued)

Field	Description
	NOTE: Propagation of CLKOUT1 to an external pad requires the proper setting of the related GPSn and GPIOn_PER fields. The following settings define the signal to be output on CLKOUT1 based on the setting of the associated CLKDIS and CLKOSEL fields. 000 Function = SYS_CLK continuous after reset 001 Function = MSTR_2X_CLK continuous after reset 010 Function = DIV2_BUS_CLK (BUS_CLK/2, used for flash memory module) continuous after reset 011 Function = MSTR_OSC (master clock) continuous after reset 100 Function = F_{IRC8M} (8 MHz / 2 MHz internal reference clock) 101 Function = Frosc200K (200 kHz relaxation oscillator clock) 110 Reserved. For normal operation, do not write 11x. 111 Reserved. For normal operation, do not write 11x.
6 Reserved	This field is reserved. Always write 0 to this bit for normal operation.
5 CLKDIS0	Disable for CLKOUT0 0 CLKOUT0 output is enabled and outputs the signal indicated by CLKOSEL0 1 CLKOUT0 is disabled
4–3 Reserved	This field is reserved. Always write 0 to this field for normal operation.
CLKOSEL0	CLKOUT0 Select Selects the clock source for the CLKOUT0 pin. The internal delay to the CLKOUT0 output is unspecified. The signal at the output pad is undefined when the CLKOUT0 signal frequency exceeds the rated frequency of the IO cell. CLKOUT0 may glitch when CLKDIS0, CLKOSEL0, or CLKODIV settings are changed. **NOTE:** Propagation of CLKOUT0 to an external pad requires the proper setting of the related GPSn and GPIOn_PER fields. The following settings define the signal to be output on CLKOUT0 based on the setting of the associated CLKDIS and CLKOSEL fields. 000 Function = SYS_CLK continuous after reset 001 Function = MSTR_2X_CLK continuous after reset 010 Function = DIV2_BUS_CLK (BUS_CLK/2, used for flash memory module) continuous after reset 011 Function = MSTR_OSC (master clock) continuous after reset 100 Function = F_{IRC8M} (8 MHz / 2 MHz internal reference clock) 101 Function = Frosc200K (200 kHz relaxation oscillator clock) 110 Reserved. For normal operation, do not write 11x. 111 Reserved. For normal operation, do not write 11x.

9.2.7 Peripheral Clock Rate Register (SIM_PCR)

By default, all peripherals are clocked at the bus clock rate (BUS_CLK). Selected peripherals can be clocked at two times this normal rate: BUS_2X_CLK. This register enables high-speed clocking for peripherals that support it.

Peripherals should not be left in an enabled or operating mode while reconfiguring their clocks using the controls in the SIM or OCCS module. PCR bits should therefore be changed only while the applicable peripheral is disabled. Refer to the peripheral's description for details.

When a peripheral operates in high-speed mode, the I/O rate to the peripheral remains limited to the system clock rate because that is the rate at which the processor operates. For peripherals with a single clock input, that clock operates at the high-speed rate and a high-speed I/O gasket is used to coordinate I/O with the processor. For peripherals with separate I/O and "run" clocks, the I/O clock operates at the normal peripheral clock rate and only the "run" clock operates at the 2x high-speed rate.

Address: E400h base + Bh offset = E40Bh

Bit	15	14	13	12	11	10	9	8
Read	0		SCI0_CR	SCI1_CR	TMR_CR	PWM_CR	LPI2C1_CR	LPI2C0_CR
Write								
Reset	0	0	0	0	0	1	0	0

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

SIM_PCR field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SCI0_CR	SCI0 Clock Rate This bit selects the clock speed for the SCI0 module. 0 SCI0 clock rate equals bus clock rate (default) 1 SCI0 clock rate equals 2X bus clock rate
12 SCI1_CR	SCI1 Clock Rate This bit selects the clock speed for the SCI1 module. 0 SCI1 clock rate equals bus clock rate (default) 1 SCI1 clock rate equals 2X bus clock rate
11 TMR_CR	TMR Clock Rate This bit selects the clock speed for the TMR module. 0 TMR clock rate equals bus clock rate (default) 1 TMR clock rate equals 2X bus clock rate
10 PWM_CR	PWM Clock Rate This bit selects the clock speed for the PWM module.

Table continues on the next page...

SIM_PCR field descriptions (continued)

Field	Description
	0 PWM clock rate equals bus clock rate 1 PWM clock rate equals 2X bus clock rate (default)
9 LPI2C1_CR	LPI2C1 Clock Rate This bit selects the clock speed for the LPI2C1 module. 0 LPI2C1 clock rate equals bus clock rate (default) 1 LPI2C1 clock rate equals 2X bus clock rate
8 LPI2C0_CR	LPI2C0 Clock Rate This bit selects the clock speed for the LPI2C0 module. 0 LPI2C0 clock rate equals bus clock rate (default) 1 LPI2C0 clock rate equals 2X bus clock rate
7 Reserved	This field is reserved.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

9.2.8 Peripheral Clock Enable Register 0 (SIM_PCE0)

The Peripheral Clock Enable registers enable or disable clocking of individual peripherals. Enabling the clocks of only peripherals that are in use can achieve significant power savings. When a peripheral's clock is disabled, no functionality, including I/O, is available to that peripheral.

Peripherals should not be left in an enabled or operating mode while their clocks are disabled or while their clocks are reconfigured using the controls in the SIM or OCCS module. PCE bits should therefore be changed only while the applicable peripheral is disabled. Refer to the peripheral's description for further details.

Enabled peripheral clocks still become disabled in STOP mode unless the peripheral's STOP Disable control in the SDn registers is set to 1.

Address: E400h base + Ch offset = E40Ch

Bit	15	14	13	12	11	10	9	8
Read	TA0	TA1	TA2	TA3	0			
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	GPIOA	GPIOB	GPIOC	GPIOD	GPIOE	GPIOF	0
Write								
Reset	0	0	0	0	0	0	0	0

SIM_PCE0 field descriptions

Field	Description
15 TA0	TMRA0 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
14 TA1	TMRA1 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
13 TA2	TMRA2 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
12 TA3	TMRA3 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
11–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 GPIOA	GPIOA IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
5 GPIOB	GPIOB IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
4 GPIOC	GPIOC IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
3 GPIOD	GPIOD IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
2 GPIOE	GPIOE IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
1 GPIOF	GPIOF IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

9.2.9 Peripheral Clock Enable Register 1 (SIM_PCE1)

The Peripheral Clock Enable registers enable or disable clocking of individual peripherals. Enabling the clocks of only peripherals that are in use can achieve significant power savings. When a peripheral's clock is disabled, no functionality, including I/O, is available to that peripheral.

Peripherals should not be left in an enabled or operating mode while their clocks are disabled or while their clocks are reconfigured using the controls in the SIM or OCCS module. PCE bits should therefore be changed only while the applicable peripheral is disabled. Refer to the peripheral's description for further details.

Enabled peripheral clocks still become disabled in STOP mode unless the peripheral's STOP Disable control in the SDn registers is set to 1.

Address: E400h base + Dh offset = E40Dh

Bit	15	14	13	12	11	10	9	8
Read	0	Reserved	DAC	SCI0	SCI1	Reserved	QSPI0	Reserved
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	LPI2C1	LPI2C0	0				
Write								
Reset	0	0	0	0	0	0	0	0

SIM_PCE1 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 Reserved	This field is reserved.
13 DAC	DAC IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
12 SCI0	SCI0 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
11 SCI1	SCI1 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
10 Reserved	This field is reserved.

Table continues on the next page...

SIM_PCE1 field descriptions (continued)

Field	Description
9 QSPI0	QSPI0 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
8 Reserved	This field is reserved.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 LPI2C1	LPI2C1 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
5 LPI2C0	LPI2C0 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

9.2.10 Peripheral Clock Enable Register 2 (SIM_PCE2)

The Peripheral Clock Enable registers enable or disable clocking of individual peripherals. Enabling the clocks of only peripherals that are in use can achieve significant power savings. When a peripheral's clock is disabled, no functionality, including I/O, is available to that peripheral.

Peripherals should not be left in an enabled or operating mode while their clocks are disabled or while their clocks are reconfigured using the controls in the SIM or OCCS module. PCE bits should therefore be changed only while the applicable peripheral is disabled. Refer to the peripheral's description for further details.

Enabled peripheral clocks still become disabled in STOP mode unless the peripheral's STOP Disable control in the SDn registers is set to 1.

Address: E400h base + Eh offset = E40Eh

Bit	15	14	13	12	11	10	9	8
Read	0			CMPA	CMPB	CMPC	CMPD	0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	CYCADC	0	CRC	Reserved	PIT0	PIT1	0	
Write								
Reset	0	0	0	0	0	0	0	0

SIM_PCE2 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 CMPA	CMPA IPBus Clock Enable (enables both CMP and 8-bit reference DAC) 0 The peripheral is not clocked. 1 The peripheral is clocked.
11 CMPB	CMPB IPBus Clock Enable (enables both CMP and 8-bit reference DAC) 0 The peripheral is not clocked. 1 The peripheral is clocked.
10 CMPC	CMPC IPBus Clock Enable (enables both CMP and 8-bit reference DAC) 0 The peripheral is not clocked. 1 The peripheral is clocked.
9 CMPD	CMPD IPBus Clock Enable (enables both CMP and 8-bit reference DAC) 0 The peripheral is not clocked. 1 The peripheral is clocked.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CYCADC	Cyclic ADC IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 CRC	CRC IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
4 Reserved	This field is reserved.
3 PIT0	Programmable Interval Timer IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
2 PIT1	Programmable Interval Timer IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

9.2.11 Peripheral Clock Enable Register 3 (SIM_PCE3)

The Peripheral Clock Enable registers enable or disable clocking of individual peripherals. Enabling the clocks of only peripherals that are in use can achieve significant power savings. When a peripheral's clock is disabled, no functionality, including I/O, is available to that peripheral.

Peripherals should not be left in an enabled or operating mode while their clocks are disabled or while their clocks are reconfigured using the controls in the SIM or OCCS module. PCE bits should therefore be changed only while the applicable peripheral is disabled. Refer to the peripheral's description for further details.

Enabled peripheral clocks still become disabled in STOP mode unless the peripheral's STOP Disable control in the SDn registers is set to 1.

Address: E400h base + Fh offset = E40Fh

Bit	15	14	13	12	11	10	9	8
Read	0			Reserved	OPAMPA	OPAMPB	ROM	Reserved
Write								
Reset	0	0	0	1	0	0	1	0

Bit	7	6	5	4	3	2	1	0
Read	PWMACH0	PWMACH1	PWMACH2	PWMACH3	0			
Write								
Reset	0	0	0	0	0	0	0	0

SIM_PCE3 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 Reserved	This field is reserved.
11 OPAMPA	OPAMPA IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
10 OPAMPB	OPAMPB IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
9 ROM	ROM IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
8 Reserved	This field is reserved.

Table continues on the next page...

SIM_PCE3 field descriptions (continued)

Field	Description
7 PWMACH0	PWMA Channel 0 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
6 PWMACH1	PWMA Channel 1 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
5 PWMACH2	PWMA Channel 2 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
4 PWMACH3	PWMA Channel 3 IPBus Clock Enable 0 The peripheral is not clocked. 1 The peripheral is clocked.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

9.2.12 Peripheral Clock STOP Disable Register 0 (SIM_SD0)

By default, peripheral clocks are disabled in stop mode to maximize power savings. The stop disable controls in the SD register override individual peripheral clocks so that they continue to operate in stop mode. Because asserting an interrupt makes the system return to run mode, this feature is provided so that selected peripherals can continue to operate in stop mode for the purpose of generating a wakeup interrupt.

For power-conscious applications, only an essential set of peripherals should be configured to remain operational in stop mode.

Peripherals should be put in a non-operating (disabled) configuration before the device enters stop mode unless their corresponding STOP Disable control is set to 1. Refer to the peripheral's description for details. IP bus reads and writes cannot be made to a module that has its clock disabled.

The SD register controls have lower priority than the PCE (Peripheral Clock Enable) register controls. If the peripheral PCE control is set to 0, the peripheral clock is disabled in all modes, including stop mode.

Address: E400h base + 10h offset = E410h

Bit	15	14	13	12	11	10	9	8
Read Write	TA0	TA1	TA2	TA3	0			
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	GPIOA	GPIOB	GPIOC	GPIOD	GPIOE	GPIOF	0
Write								
Reset	0	0	0	0	0	0	0	0

SIM_SD0 field descriptions

Field	Description
15 TA0	TMRA0 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
14 TA1	TMRA1 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
13 TA2	TMRA2 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
12 TA3	TMRA3 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
11–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 GPIOA	GPIOA IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
5 GPIOB	GPIOB IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.

Table continues on the next page...

SIM_SD0 field descriptions (continued)

Field	Description
4 GPIOC	GPIOC IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
3 GPIOD	GPIOD IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
2 GPIOE	GPIOE IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
1 GPIOF	GPIOF IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

9.2.13 Peripheral Clock STOP Disable Register 1 (SIM_SD1)

By default, peripheral clocks are disabled in stop mode to maximize power savings. The stop disable controls in the SD register override individual peripheral clocks so that they continue to operate in stop mode. Because asserting an interrupt makes the system return to run mode, this feature is provided so that selected peripherals can continue to operate in stop mode for the purpose of generating a wakeup interrupt.

For power-conscious applications, only an essential set of peripherals should be configured to remain operational in stop mode.

Peripherals should be put in a non-operating (disabled) configuration before the device enters stop mode unless their corresponding STOP Disable control is set to 1. Refer to the peripheral's description for details. IP bus reads and writes cannot be made to a module that has its clock disabled.

The SD register controls have lower priority than the PCE (Peripheral Clock Enable) register controls. If the peripheral PCE control is set to 0, the peripheral clock is disabled in all modes, including stop mode.

Address: E400h base + 11h offset = E411h

Bit	15	14	13	12	11	10	9	8
Read	0	Reserved	DAC	SCI0	SCI1	Reserved	QSPI0	Reserved
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	LPI2C1	LPI2C0	0				
Write								
Reset	0	0	0	0	0	0	0	0

SIM_SD1 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 Reserved	This field is reserved.
13 DAC	DAC IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
12 SCI0	SCI0 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
11 SCI1	SCI1 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
10 Reserved	This field is reserved.
9 QSPI0	QSPI0 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.

Table continues on the next page...

SIM_SD1 field descriptions (continued)

Field	Description
8 Reserved	This field is reserved.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 LPI2C1	LPI2C1 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode, but the LPI2C1 module will not enter stop mode.
5 LPI2C0	LPI2C0 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode, but the LPI2C0 module will not enter stop mode.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

9.2.14 Peripheral Clock STOP Disable Register 2 (SIM_SD2)

By default, peripheral clocks are disabled in stop mode to maximize power savings. The stop disable controls in the SD register override individual peripheral clocks so that they continue to operate in stop mode. Because asserting an interrupt makes the system return to run mode, this feature is provided so that selected peripherals can continue to operate in stop mode for the purpose of generating a wakeup interrupt.

For power-conscious applications, only an essential set of peripherals should be configured to remain operational in stop mode.

Peripherals should be put in a non-operating (disabled) configuration before the device enters stop mode unless their corresponding STOP Disable control is set to 1. Refer to the peripheral's description for details. IP bus reads and writes cannot be made to a module that has its clock disabled.

The SD register controls have lower priority than the PCE (Peripheral Clock Enable) register controls. If the peripheral PCE control is set to 0, the peripheral clock is disabled in all modes, including stop mode.

Address: E400h base + 12h offset = E412h

Bit	15	14	13	12	11	10	9	8
Read	0			CMPA	CMPB	CMPC	CMPD	0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	CYCADC	0	CRC	Reserved	PIT0	PIT1	0	
Write								
Reset	0	0	0	0	0	0	0	0

SIM_SD2 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 CMPA	CMPA IPBus STOP Disable (disables both CMP and 8-bit reference DAC) This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
11 CMPB	CMPB IPBus STOP Disable (disables both CMP and 8-bit reference DAC) This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
10 CMPC	CMPC IPBus STOP Disable (disables both CMP and 8-bit reference DAC) This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
9 CMPD	CMPD IPBus STOP Disable (disables both CMP and 8-bit reference DAC) This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CYCADC	Cyclic ADC IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
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_SD2 field descriptions (continued)

Field	Description
5 CRC	CRC IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
4 Reserved	This field is reserved.
3 PIT0	Programmable Interval Timer IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
2 PIT1	Programmable Interval Timer IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

9.2.15 Peripheral Clock STOP Disable Register 3 (SIM_SD3)

By default, peripheral clocks are disabled in stop mode to maximize power savings. The stop disable controls in the SD register override individual peripheral clocks so that they continue to operate in stop mode. Because asserting an interrupt makes the system return to run mode, this feature is provided so that selected peripherals can continue to operate in stop mode for the purpose of generating a wakeup interrupt.

For power-conscious applications, only an essential set of peripherals should be configured to remain operational in stop mode.

Peripherals should be put in a non-operating (disabled) configuration before the device enters stop mode unless their corresponding STOP Disable control is set to 1. Refer to the peripheral's description for details. IP bus reads and writes cannot be made to a module that has its clock disabled.

The SD register controls have lower priority than the PCE (Peripheral Clock Enable) register controls. If the peripheral PCE control is set to 0, the peripheral clock is disabled in all modes, including stop mode.

Address: E400h base + 13h offset = E413h

Bit	15	14	13	12	11	10	9	8
Read	0				OPAMPA	OPAMPB	ROM	Reserved
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	PWMACH0	PWMACH1	PWMACH2	PWMACH3	0			
Write								
Reset	0	0	0	0	0	0	0	0

SIM_SD3 field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 OPAMPA	OPAMPA IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
10 OPAMPB	OPAMPB IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
9 ROM	ROM IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
8 Reserved	This field is reserved.
7 PWMACH0	PWMA Channel 0 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
6 PWMACH1	PWMA Channel 1 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
5 PWMACH2	PWMA Channel 2 IPBus STOP Disable

Table continues on the next page...

SIM_SD3 field descriptions (continued)

Field	Description
	This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
4 PWMACH3	PWMA Channel 3 IPBus STOP Disable This bit enables peripheral clocking during stop mode to the indicated peripheral provided the corresponding PCEn bit is set to 1 so that the peripheral clock itself is enabled. 0 The peripheral is not clocked in stop mode. 1 The peripheral is clocked in stop mode.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

9.2.16 I/O Short Address Location Register (SIM_IOSAHI)

The I/O short address location registers specify the memory referenced through the I/O short address mode, which allows the instruction to specify the lower 6 bits of the address. The upper address bits are not directly controllable. The I/O short address location registers allow limited control of the full address.

With this register set, an interrupt driver can set the combined ISAL register to point to its peripheral registers and then use the I/O short addressing mode to reference them. The ISR should restore this register to its previous contents prior to returning from interrupt.

NOTE

The default value of this register points to the beginning of the slave peripheral region at E000h.

NOTE

The pipeline delay between setting this register set and using short I/O addressing with the new value is five cycles.

Address: E400h base + 14h offset = E414h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0														ISAL23_22	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_IOSAHI field descriptions

Field	Description
15–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ISAL23_22	Bits 23:22 of the address The I/O short address is calculated by concatenating the combined ISAL value with the 6-bit short address from the CPU short address opcode.

9.2.17 I/O Short Address Location Register (SIM_IOSALO)

The I/O short address location registers specify the memory referenced through the I/O short address mode, which allows the instruction to specify the lower 6 bits of the address. The upper address bits are not directly controllable. The I/O short address location registers allow limited control of the full address.

With this register set, an interrupt driver can set the combined ISAL register to point to its peripheral registers and then use the I/O short addressing mode to reference them. The ISR should restore this register to its previous contents prior to returning from interrupt.

NOTE

The default value of this register points to the beginning of the slave peripheral region at E000h.

NOTE

The pipeline delay between setting this register set and using short I/O addressing with the new value is five cycles.

Address: E400h base + 15h offset = E415h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	ISAL21_6															
Reset	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0

SIM_IOSALO field descriptions

Field	Description
ISAL21_6	Bits 21:6 of the address The I/O short address is calculated by concatenating the combined ISAL value with the 6-bit short address from the CPU short address opcode.

9.2.18 Protection Register (SIM_PROT)

This register provides write protection of selected control fields for safety critical applications. The primary purpose is to prevent unsafe conditions due to the unintentional modification of these fields between the onset of a code runaway and a reset by the COP watchdog. GPIO and Internal Peripheral Select Protection (GIPSP) write protects the registers in the SIM, XBAR, EVTG, and GPIO modules that control inter-peripheral signal multiplexing and I/O cell configuration. Peripheral Clock Enable Protection (PCEP) write protects the SIM registers that contain peripheral-specific clock controls. GDP provides write protection of the GPIO Port D registers separately from the other ports protected by the GIPSP field. Some peripherals provide additional safety features. Refer to peripheral descriptions for details.

GIPSP protects the contents of the SIM registers that control multiplexing of inter-peripheral connections (GPSn and IPSn) as well as all inter-peripheral XBAR and EVTG registers. GIPSP also write protects some registers in the GPIO module other than for port D, including the GPIOn_PER registers that select between peripheral and GPIO ownership of the I/O cell, the GPIOn_PPMODE registers the control the I/O cell's push/pull mode, and GPIOn_DRIVE registers that control the I/O cell's drive strength.

GDP provides write protection for the GPIO Port D registers, which include JTAG and reset functionality. These include GPIO_D_PER, GPIO_D_PPMODE, and GPIO_D_DRIVE as well as the GPSDL register.

PCEP write protects the SIM peripheral clock enable registers (PCEn), the SIM peripheral stop disable registers (SDn), the SIM peripheral software reset registers (PSWRn), and the SIM peripheral clock rate registers (PCR).

For flexibility, write protection control values may themselves be optionally locked (write protected). To this end, protection controls in this register have two bit values. The right bit determines the setting of the control, and the left bit determines whether the value is locked. While a protection control remains unlocked, protection can be disabled and re-enabled as desired. When a protection control is locked, its value can be altered only by a device reset, which restores its default non-locked value.

Address: E400h base + 16h offset = E416h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								PMODE		GDP		PCEP		GIPSP	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_PROT 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 PMODE	Power Mode Control Write Protection Enables write protection of the PWRMODE register. 00 Write protection off (default). 01 Write protection on. 10 Write protection off and locked until chip reset. 11 Write protection on and locked until chip reset.
5–4 GDP	GPIO Port D Protection Enables write protection of GPIO_D_PER, GPIO_D_PPMODE, and GPIO_D_DRIVE registers. This field also write protects the GPSDL register and CTRL[RST_FILT] bit. GPIO Port D contains JTAG and reset functions that are protected separately from other GPIO ports. 00 Write protection off (default). 01 Write protection on. 10 Write protection off and locked until chip reset. 11 Write protection on and locked until chip reset.
3–2 PCEP	Peripheral Clock Enable Protection Enables write protection of all fields in the PCEn, SDn, PSWRn, and PCR registers. 00 Write protection off (default). 01 Write protection on. 10 Write protection off and locked until chip reset. 11 Write protection on and locked until chip reset.
GIPSP	GPIO and Internal Peripheral Select Protection Enables write protection of GPSn and IPSn registers in the SIM. This field also write protects registers in other modules including all XBAR, EVTG, GPIOn_PER, GPIOn_PPMODE, and GPIOn_DRIVE registers. This field does not protect GPIO Port D registers or the GPSDL register; the GDP field provides that protection. 00 Write protection off (default). 01 Write protection on. 10 Write protection off and locked until chip reset. 11 Write protection on and locked until chip reset.

9.2.19 GPIOA LSBs Peripheral Select Register (SIM_GPSAL)

Most I/O pads have an associated GPIO function that, when enabled, can control and observe the I/O pad. As an alternative to the GPIO function, most I/O can be configured to connect to one of several internal peripheral functions. When the GPIOn_PER bit for an I/O is set to 0, the associated GPIO has control of the I/O to the exclusion of any internal peripheral function. The GPIOn_PER bit for an I/O must be set to 1 for the I/O to access any of its internal peripheral functions. If an I/O pad can be configured to connect to one of several peripheral functions, the GPSn field for that GPIO is used to select the active peripheral function.

NOTE

A GPIO with only one peripheral function does not require a GPS field. That peripheral function is always enabled when the PER field of the corresponding GPIO is set to 1.

In some cases there are multiple I/O pads, each of which can be configured via their GPIOn_PER and SIM GPS settings to connect to the same internal peripheral function. If more than one I/O is connected to the same peripheral output function, the peripheral output signal safely fans out to each of these I/O. However, if more than one I/O is connected to the same peripheral input signal, that peripheral input is the logical OR or AND of these multiple sources and is therefore invalid. As a result, at most one I/O should be configured to control a specific peripheral input function.

The user may opt not to use a specific peripheral input or output function. If no I/O is configured to observe a peripheral output function, then the peripheral output signal is inaccessible. If no I/O is configured to control a peripheral input signal, that peripheral input is tied to an application-appropriate constant value.

When a GPIO's internal peripheral functions include a connection to an output of an Internal Crossbar Switch (XBAR) module, that GPIO can be driven by any input to the XBAR by properly configuring the XBAR module. Similarly, when a GPIO's internal functions include a connection to an input of an Internal Crossbar Switch (XBAR) module, that GPIO can feed any device fed by the XBAR's outputs by properly configuring the XBAR module.

GPSn settings should not be altered while an affected peripheral is in an enabled (operational) configuration. See the peripheral's description for further details.

Address: E400h base + 17h offset = E417h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0															A0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_GPSAL field descriptions

Field	Description
15–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 A0	Configure GPIO A0 0 Function = ANA0/CMPA_IN3/OPAMPA_IN3; Peripheral = ADC/CMPA/OPAMPA; Direction = IN 1 Function = CMPC_O; Peripheral = CMPC; Direction = OUT

9.2.20 GPIOB LSBs Peripheral Select Register (SIM_GPSBL)

See the description of the GPSAL register.

Address: E400h base + 18h offset = E418h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0														0	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_GPSBL field descriptions

Field	Description
15–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

9.2.21 GPIOC LSBs Peripheral Select Register (SIM_GPSCL)

See the description of the GPSAL register.

Address: E400h base + 19h offset = E419h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	C7		C6		0	C5	C4		C3		C2		0			C0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_GPSCL field descriptions

Field	Description
15–14 C7	Configure GPIO C7 00 Function = SS0_B; Peripheral = SPI0; Direction = IO 01 Function = TXD0; Peripheral = SCI0; Direction = OUT 10 Function = XB_IN8; Peripheral = XBAR; Direction = IN 11 Function = XB_OUT6; Peripheral = XBAR; Direction = OUT
13–12 C6	Configure GPIO C6 00 Function = TA2; Peripheral = TMRA; Direction = IO 01 Function = XB_IN3; Peripheral = XBAR; Direction = IN 10 Function = CMP_REF; Peripheral = CMP*x*; Direction = IN 11 Function = SS0_B; Peripheral = SPI0; Direction = IO
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 C5	Configure GPIO C5 0 Function = DACA_O; Peripheral = DAC; Direction = OUT 1 Function = XB_IN7; Peripheral = XBAR; Direction = IN
9–8 C4	Configure GPIO C4 00 Function = TA1; Peripheral = TMRA; Direction = IO 01 Function = CMPB_O; Peripheral = CMPB; Direction = OUT 10 Function = XB_IN8; Peripheral = XBAR; Direction = IN 11 Function = OPAMPA_OUT; Peripheral = OPAMPA; Direction = OUT
7–6 C3	Configure GPIO C3 00 Function = TA0; Peripheral = TMRA; Direction = IO 01 Function = CMPA_O; Peripheral = CMPA; Direction = OUT 10 Function = RXD0; Peripheral = SCI0; Direction = IN 11 Function = CLKIN1; Peripheral = OCCS; Direction = IN
5–4 C2	Configure GPIO C2 00 Function = TXD0; Peripheral = SCI0; Direction = IO 01 Function = XB_OUT11; Peripheral = XBAR; Direction = OUT

Table continues on the next page...

SIM_GPSCL field descriptions (continued)

Field	Description
	10 Function = XB_IN2; Peripheral = XBAR; Direction = IN 11 Function = CLKOUT0; Peripheral = OCCS; Direction = OUT
3–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 C0	Configure GPIO C0 0 Function = EXTAL; Peripheral = OSC; Direction = IN 1 Function = CLKIN0; Peripheral = OCCS; Direction = IN

9.2.22 GPIOC MSBs Peripheral Select Register (SIM_GPSCH)

See the description of the GPSAL register.

Address: E400h base + 1Ah offset = E41Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	C15		C14		C13		C12		C11		C10		C9		C8	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_GPSCH field descriptions

Field	Description
15–14 C15	Configure GPIO C15 00 Function = SCL0; Peripheral = LPI2C0; Direction = IO 01 Function = XB_OUT5; Peripheral = XBAR; Direction = OUT 10 Function = PWMA_FAULT5; Peripheral = PWMA; Direction = IN 11 reserved
13–12 C14	Configure GPIO C14 00 Function = SDA0; Peripheral = LPI2C0; Direction = IO 01 Function = XB_OUT4; Peripheral = XBAR; Direction = OUT 10 Function = PWMA_FAULT4; Peripheral = PWMA; Direction = IN 11 reserved
11–10 C13	Configure GPIO C13 00 Function = TA3; Peripheral = TMRA; Direction = IO 01 Function = XB_IN6; Peripheral = XBAR; Direction = IN 10 Function = EWM_OUT_B; Peripheral = EWM; Direction = OUT 11 Reserved
9–8 C12	Configure GPIO C12 00 Function = SDAS0; Peripheral = LPI2C0; Direction = IO 01 Function = SDA1; Peripheral = LPI2C1; Direction = IO

Table continues on the next page...

SIM_GPSCH field descriptions (continued)

Field	Description
	10 Function = RXD1; Peripheral = SCI1; Direction = IN 11 Function = PWMA_1X; Peripheral = PWMA; Direction = IO
7–6 C11	Configure GPIO C11 00 Function = SCLS0; Peripheral = LPI2C0; Direction = IO 01 Function = SCL1; Peripheral = LPI2C1; Direction = IO 10 Function = TXD1; Peripheral = SCI1; Direction = IO 11 Function = PWMA_0X; Peripheral = PWMA; Direction = IO
5–4 C10	Configure GPIO C10 00 Function = MOSI0; Peripheral = SPI0; Direction = IO 01 Function = XB_IN5; Peripheral = XBAR; Direction = IN 10 Function = MISO0; Peripheral = SPI0; Direction = IO 11 Function = XB_OUT9; Peripheral = XBAR; Direction = OUT
3–2 C9	Configure GPIO C9 00 Function = SCLK0; Peripheral = SPI0; Direction = IO 01 Function = XB_IN4; Peripheral = XBAR; Direction = IN 10 Function = TXD0; Peripheral = SCI0; Direction = OUT 11 Function = XB_OUT8; Peripheral = XBAR; Direction = OUT
C8	Configure GPIO C8 00 Function = MISO0; Peripheral = SPI0; Direction = IO 01 Function = RXD0; Peripheral = SCI0; Direction = IN 10 Function = XB_IN9; Peripheral = XBAR; Direction = IN 11 Reserved

9.2.23 GPIOE LSBs Peripheral Select Register (SIM_GPSEL)

See the description of the GPSAL register.

Address: E400h base + 1Ch offset = E41Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	E7		E6		E5		E4		E3		E2		E1		E0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_GPSEL field descriptions

Field	Description
15–14 E7	Configure GPIO E7 00 Function = PWMA_3A; Peripheral = PWMA; Direction = IO 01 Function = XB_IN5; Peripheral = XBAR; Direction = IN

Table continues on the next page...

SIM_GPSEL field descriptions (continued)

Field	Description
	10 Reserved 11 Function = XB_OUT11; Peripheral = XBAR; Direction = OUT
13–12 E6	Configure GPIO E6 00 Function = PWMA_3B; Peripheral = PWMA; Direction = IO 01 Function = XB_IN4; Peripheral = XBAR; Direction = IN 10 Reserved 11 Function = XB_OUT10; Peripheral = XBAR; Direction = OUT
11–10 E5	Configure GPIO E5 00 Function = PWMA_2A; Peripheral = PWMA; Direction = IO 01 Function = XB_IN3; Peripheral = XBAR; Direction = IN 10 Function = SDA1; Peripheral = LPI2C1; Direction = IO 11 Function = XB_OUT9; Peripheral = XBAR; Direction = OUT
9–8 E4	Configure GPIO E4 00 Function = PWMA_2B; Peripheral = PWMA; Direction = IO 01 Function = XB_IN2; Peripheral = XBAR; Direction = IN 10 Function = SCL1; Peripheral = LPI2C1; Direction = IO 11 Function = XB_OUT8; Peripheral = XBAR; Direction = OUT
7–6 E3	Configure GPIO E3 00 Function = PWMA_1A; Peripheral = PWMA; Direction = IO 01 reserved 10 reserved 11 Function = XB_OUT7; Peripheral = XBAR; Direction = OUT
5–4 E2	Configure GPIO E2 00 Function = PWMA_1B; Peripheral = PWMA; Direction = IO 01 reserved 10 reserved 11 Function = XB_OUT6; Peripheral = XBAR; Direction = OUT
3–2 E1	Configure GPIO E1 00 Function = PWMA_0A; Peripheral = PWMA; Direction = IO 01 reserved 10 reserved 11 Function = XB_OUT5; Peripheral = XBAR; Direction = OUT
E0	Configure GPIO E0 00 Function = PWMA_0B; Peripheral = PWMA; Direction = IO 01 reserved 10 reserved 11 Function = XB_OUT4; Peripheral = XBAR; Direction = OUT

9.2.24 GPIOF LSBs Peripheral Select Register (SIM_GPSFL)

See the description of the GPSAL register.

Address: E400h base + 1Eh offset = E41Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	F7		F6		F5		F4		F3		F2		F1		F0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_GPSFL field descriptions

Field	Description
15–14 F7	Configure GPIO F7 00 Reserved 01 Function = CMPC_O; Peripheral = CMP C; Direction = OUT 10 Function = SDAS1; Peripheral = LPI2C1; Direction = IO 11 Function = XB_IN3; Peripheral = XBAR; Direction = IN
13–12 F6	Configure GPIO F6 00 Reserved 01 Function = PWMA_3X; Peripheral = PWMA; Direction = IO 10 Function = SCLS1; Peripheral = LPI2C1; Direction = IO 11 Function = XB_IN2; Peripheral = XBAR; Direction = IN
11–10 F5	Configure GPIO F5 00 Function = RXD1; Peripheral = SCI1; Direction = IN 01 Function = XB_OUT9; Peripheral = XBAR; Direction = OUT 10 Function = PWMA_1X; Peripheral = PWMA; Direction = IO 11 Function = PWMA_FAULT7; Peripheral = PWMA; Direction = IN
9–8 F4	Configure GPIO F4 00 Function = TXD1; Peripheral = SCI1; Direction = IO 01 Function = XB_OUT8; Peripheral = XBAR; Direction = OUT 10 Function = PWMA_0X; Peripheral = PWMA; Direction = IO 11 Function = PWMA_FAULT6; Peripheral = PWMA; Direction = IN
7–6 F3	Configure GPIO F3 00 Function = SDA1; Peripheral = LPI2C1; Direction = IO 01 Function = XB_OUT7; Peripheral = XBAR; Direction = OUT 10 Function = SCL0; Peripheral = LPI2C0; Direction = IO 11 reserved
5–4 F2	Configure GPIO F2 00 Function = SCL1; Peripheral = LPI2C1; Direction = IO 01 Function = XB_OUT6; Peripheral = XBAR; Direction = OUT

Table continues on the next page...

SIM_GPSFL field descriptions (continued)

Field	Description
	10 Function = SDA0; Peripheral = LPI2C0; Direction = IO 11 reserved
3–2 F1	Configure GPIO F1 00 Function = CLKOUT1; Peripheral = OCCS; Direction = OUT 01 Function = XB_IN7; Peripheral = XBAR; Direction = IN 10 Function = CMPD_O; Peripheral = CMP D; Direction = OUT 11 Reserved
F0	Configure GPIO F0 00 Function = XB_IN6; Peripheral = XBAR; Direction = IN 01 Reserved 10 Reserved 11 Function = OPAMPB_OUT; Peripheral = OPAMP B; Direction = OUT

9.2.25 GPIOF MSBs Peripheral Select Register (SIM_GPSFH)

See the description of the GPSAL register.

Address: E400h base + 1Fh offset = E41Fh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0														F8	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_GPSFH field descriptions

Field	Description
15–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
F8	Configure GPIO F8 00 Function = RXD0; Peripheral = SCI0; Direction = IN 01 Function =XB_OUT10; Peripheral = XBAR; Direction = OUT 10 Function = CMPD_O; Peripheral = CMP D; Direction = OUT 11 Function = PWMA_2X; Peripheral = PWMA; Direction = IO

9.2.26 Internal Peripheral Select Register 0 (SIM_IPS0)

The IPS register implements controls that affect the integration or communication between parts. Various circumstances require these controls.

Some peripheral inputs have the ability to be fed either by XBAR outputs or by GPIO. In these cases, an additional layer of internal multiplexing selects which source is active and feeds the peripheral input.

In other cases, peripherals have control relationships. For example, one peripheral may be configured to operate as a slave of another and require inputs to be configured appropriately based on that choice. This configuration may require the setting of control inputs or the switching of muxing relationships between the blocks.

NOTE

The TMRAn fields select which signal drives the corresponding TMRAn inputs. The choice in each case is between one or more external I/O pads and an XBAR output. When selecting the external I/O, you must set the GPIOn_PER bit for that I/O to 1 and configure that GPIO's SIM GPS field (if there is one) to feed that TMR channel. A standard requirement of GPS muxing to avoid contention is to configure, at most, one GPIO at a time to feed a specific peripheral input function. When more than one GPIO source is listed, the muxing uses the signal from the GPIO input that is currently sourcing the signal.

Address: E400h base + 22h offset = E422h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0				TA3	TA2	TA1	TA0	0						SCI1	SCI0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_IPS0 field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 TA3	Select TMRA Input 3 0 Function = GPIOC13 ; Peripheral = GPIOC; Direction = IN 1 Function = XB_OUT41; Peripheral = XBAR; Direction = IN
10 TA2	Select TMRA Input 2 0 Function = GPIOC6 ; Peripheral = GPIOC; Direction = IN 1 Function = XB_OUT40; Peripheral = XBAR; Direction = IN
9 TA1	Select TMRA Input 1 0 Function = GPIOC4; Peripheral = GPIOC; Direction = IN 1 Function = XB_OUT39; Peripheral = XBAR; Direction = IN
8 TA0	Select TMRA Input 0

Table continues on the next page...

SIM_IPS0 field descriptions (continued)

Field	Description
	0 Function = GPIOC3; Peripheral = GPIOC; Direction = IN 1 Function = XB_OUT38; Peripheral = XBAR; Direction = IN
7–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 SCI1	Select SCI1_RXD Input 0 Function = GPIOC12 or GPIOF5; Peripheral = GPIO; Direction = IN 1 Function = XB_OUT37; Peripheral = XBARA; Direction = IN
0 SCI0	Select SCI0_RXD Input 0 Function = GPIOC3 or GPIOC8 or GPIOF8; Peripheral = GPIO; Direction = IN 1 Function = XB_OUT36; Peripheral = XBAR; Direction = IN

9.2.27 Miscellaneous Register 0 (SIM_MISC0)

This register controls various integration features of the chip.

Address: E400h base + 23h offset = E423h

Bit	15	14	13	12	11	10	9	8
Read	0					LPI2C1_TRIG_SEL	LPI2C0_TRIG_SEL	MODE_STAT
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0			Reserved	SCTRL_REORDER	FAST_MODE	CLKINSEL	PIT_MSTR
Write								
Reset	0	0	0	0	0	0	0	0

SIM_MISC0 field descriptions

Field	Description
15–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 LPI2C1_TRIG_SEL	LPI2C1 master/slave trigger select 0 Selects slave trigger 1 Selects master trigger.
9 LPI2C0_TRIG_SEL	LPI2C0 master/slave trigger select 0 Selects slave trigger 1 Selects master trigger

Table continues on the next page...

SIM_MISC0 field descriptions (continued)

Field	Description
8 MODE_STAT	Mode Status bit This status bit indicates if the system is in fast mode or normal operating mode. **NOTE:** This bit itself can only be reset by POR. 0 Device in normal operating mode (core: bus frequency as 1:1) 1 Device in fast mode (core: bus frequency as 2:1)
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 Reserved	This field is reserved. **NOTE:** Do not write logic 1 to the reserved bit.
3 SCTRL_ REORDER	This bit enables the re-ordering of scan control bits of Cyclic ADC for test channels. For more details, see the ADC chapter. 0 Normal order 1 Enable the re-ordering of ADC scan control bits
2 FAST_MODE	This bit decides if the system will boot in fast mode or normal mode. Writing to this bit comes into effect during reset if it is caused by software reset. **NOTE:** This bit itself can only be reset by POR. 0 Normal operating mode (core: bus frequency as 1:1) 1 Device boots in fast mode (core:bus frequency as 2:1) after software reset
1 CLKINSEL	CLKIN Select This bit determines the GPIO port for the CLKIN input to the OCCS. 0 CLKIN0 (GPIOC0 alt1) is selected as CLKIN 1 CLKIN1 (GPIOC3 alt3) is selected as CLKIN
0 PIT_MSTR	Select Master Programmable Interval Timer (PIT) 0 PIT0 is master PIT and PIT1 is slave PIT 1 PIT1 is master PIT and PIT0 is slave PIT

9.2.28 Peripheral Software Reset Register 0 (SIM_PSWR0)

The Peripheral Software Reset registers are used to reset individual peripherals without resetting the entire chip.

Address: E400h base + 24h offset = E424h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TA	0			0	0				GPIO	0					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_PSWR0 field descriptions

Field	Description
15 TA	TMRA Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
14–12 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.
10–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 GPIO	GPIO Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

9.2.29 Peripheral Software Reset Register 1 (SIM_PSWR1)

The Peripheral Software Reset registers are used to reset individual peripherals without resetting the entire chip.

Address: E400h base + 25h offset = E425h

Bit	15	14	13	12	11	10	9	8
Read	0	Reserved	DAC	SCI0	SCI1	Reserved	QSPI0	Reserved
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	LPI2C1	LPI2C0	0				0
Write								
Reset	0	0	0	0	0	0	0	0

SIM_PSWR1 field descriptions

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

Table continues on the next page...

SIM_PSWR1 field descriptions (continued)

Field	Description
13 DAC	DAC Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
12 SCI0	SCI0 Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
11 SCI1	SCI1 Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
10 Reserved	This field is reserved.
9 QSPI0	QSPI0 Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
8 Reserved	This field is reserved.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 LPI2C1	LPI2C1 Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
5 LPI2C0	LPI2C0 Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
4–1 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.

9.2.30 Peripheral Software Reset Register 2 (SIM_PSWR2)

The Peripheral Software Reset registers are used to reset individual peripherals without resetting the entire chip.

Address: E400h base + 26h offset = E426h

Bit	15	14	13	12	11	10	9	8
Read	EWM	0		CMP	0			0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	CYCADC	0	CRC	Reserved	PIT0	PIT1	0	
Write								
Reset	0	0	0	0	0	0	0	0

SIM_PSWR2 field descriptions

Field	Description
15 EWM	EWM Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
14–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 CMP	CMP Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
11–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CYCADC	Cyclic ADC Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 CRC	CRC Software Reset This bit causes a reset of the indicated peripheral.

Table continues on the next page...

SIM_PSWR2 field descriptions (continued)

Field	Description
	0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
4 Reserved	This field is reserved.
3 PIT0	Programmable Interval Timer Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
2 PIT1	Programmable Interval Timer Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

9.2.31 Peripheral Software Reset Register 3 (SIM_PSWR3)

The Peripheral Software Reset registers are used to reset individual peripherals without resetting the entire chip.

Address: E400h base + 27h offset = E427h

Bit	15	14	13	12	11	10	9	8
Read	0				OPAMPA	OPAMPB	DMA_MUX	Reserved
Write								
Reset	0	0	0	0	0	0	0	0

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

SIM_PSWR3 field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 OPAMPA	OPAMPA Software Reset This bit causes a reset of the indicated peripheral.

Table continues on the next page...

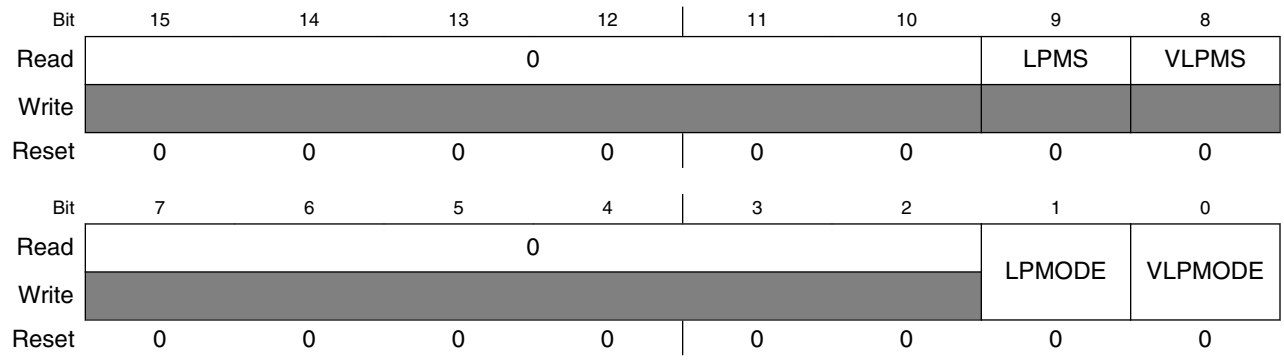
SIM_PSWR3 field descriptions (continued)

Field	Description
	0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
10 OPAMPB	OPAMPB Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
9 DMA_MUX	DMA_MUX Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
8 Reserved	This field is reserved.
7 PWMA	PWMA Software Reset This bit causes a reset of the indicated peripheral. 0 The corresponding peripheral is not reset. 1 The corresponding peripheral is reset.
6–4 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.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

9.2.32 Power Mode Register (SIM_PWRMODE)

This register controls various low power modes of the chip if the Flash Memory Module (FTFA)'s FOPT[0] bit is set (advanced power mode is enabled). All control bits in this register are write protected. See the Power Management chapter for mode details.

Address: E400h base + 28h offset = E428h



SIM_PWRMODE field descriptions

Field	Description
15–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 LPMS	LPMODE Status indicator This status bit indicates whether the chip is in LPMODE. 0 Not in LPMODE 1 In LPMODE
8 VLPMS	VLPMODE Status indicator This status bit indicates whether the chip is in VLPMODE. **NOTE:** This bit, along with the PMC's STS[SR27] bit, indicates an exit from VLPMODE. 0 Not in VLPMODE 1 In VLPMODE
7–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 LPMODE	LPMODE Entry/Exit Writing to this bit causes the device to enter LPMODE. To exit LPMODE, clear this bit. This bit can be written only if write protection is disabled (PROT[6] is 0). If both the LPMODE and VLPMODE bits are set, the VLPMODE bit has higher priority. 0 Start exit from LPMODE 1 Start entry to LPMODE

Table continues on the next page...

SIM_PWRMODE field descriptions (continued)

Field	Description
0 VLPMODE	VLPMODE Entry/Exit Writing to this bit causes the device to enter VLPMODE. To exit VLPMODE, clear this bit. This bit can be written only if write protection is disabled (PROT[6] is 0). If both the LPMODE and VLPMODE bits are set, the VLPMODE bit has higher priority. 0 Start exit from VLPMODE 1 Start entry to VLPMODE

9.2.33 Software Control Register (SIM_SCR0)

Address: E400h base + 45h offset = E445h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SCR0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SCR0 field descriptions

Field	Description
SCR0	Software Control Data This field is for general-purpose use by software. It is reset only by a power-on reset.

9.2.34 Software Control Register (SIM_SCR1)

Address: E400h base + 46h offset = E446h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SCR1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SCR1 field descriptions

Field	Description
SCR1	Software Control Data This field is for general-purpose use by software. It is reset only by a power-on reset.

9.2.35 Software Control Register (SIM_SCR2)

Address: E400h base + 47h offset = E447h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SCR2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SCR2 field descriptions

Field	Description
SCR2	Software Control Data This field is for general-purpose use by software. It is reset only by a power-on reset.

9.2.36 Software Control Register (SIM_SCR3)

Address: E400h base + 48h offset = E448h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SCR3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SCR3 field descriptions

Field	Description
SCR3	Software Control Data This field is for general-purpose use by software. It is reset only by a power-on reset.

9.2.37 Software Control Register (SIM_SCR4)

Address: E400h base + 49h offset = E449h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SCR4															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SCR4 field descriptions

Field	Description
SCR4	Software Control Data This field is for general-purpose use by software. It is reset only by a power-on reset.

9.2.38 Software Control Register (SIM_SCR5)

Address: E400h base + 4Ah offset = E44Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SCR5															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SCR5 field descriptions

Field	Description
SCR5	Software Control Data This field is for general-purpose use by software. It is reset only by a power-on reset.

9.2.39 Software Control Register (SIM_SCR6)

Address: E400h base + 4Bh offset = E44Bh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SCR6															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SCR6 field descriptions

Field	Description
SCR6	Software Control Data This field is for general-purpose use by software. It is reset only by a power-on reset.

9.2.40 ADC and TMR Select Register (SIM_ADC_TMR_SEL)

This register selects Xbar input from ADC and TMRA.

Address: E400h base + 4Dh offset = E44Dh

Bit	15	14	13	12	11	10	9	8
Read	0				0			
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0				XBAR_IN39	XBAR_IN38	XBAR_IN37	XBAR_IN36
Write								
Reset	0	0	0	0	0	0	0	0

SIM_ADC_TMR_SEL 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 XBAR_IN39	TimerA 3 or AN3 Limit. 0 TMRA3 1 ADC AN3 limit
2 XBAR_IN38	TimerA 2 or AN2 Limit. 0 TMRA2 1 ADC AN2 limit
1 XBAR_IN37	TimerA 1 or AN1 Limit. 0 TMRA1 1 ADC AN1 limit
0 XBAR_IN36	TimerA 0 or AN0 Limit. 0 TMRA0 1 ADC AN0 limit

9.3 Functional Description

9.3.1 Clock Generation Overview

The SIM uses a master clock from the OCCS module (MSTR_2X) to produce the peripheral and system (core and memory) clocks. This MSTR_2X clock input from OCCS operates at two times the system and peripheral bus rate. Peripheral and system clocks, with the exception of the RAM system clock, are generated by dividing the MSTR_2X clock by two and gating it with appropriate power mode and clock gating controls. The RAM requires a 2x system rate clock. This clock is therefore generated by gating the MSTR_2X clock with appropriate power mode and clock gating controls.

The OCCS supports several methods for deriving MSTR_2X. A master clock source (MSTR_OSC) is selected from several available sources, including an up to 50 MHz external clock input (CLKIN), a 4 MHz to 16 MHz crystal oscillator, a 200 kHz internal relaxation oscillator, or an 8 MHz / 2 MHz internal reference clock.

A high-speed clock source can be derived by multiplying MSTR_OSC frequencies between 8 MHz and 50 MHz through a PLL to a maximum allowed frequency of 400 MHz. Duty cycle correction for the high-speed clock source is achieved by a DIV2 circuit.

MSTR_2X is derived by selecting either MSTR_OSC or the high-speed clock source as the active clock source and optionally dividing it through a post-scaler. The post-scaler will support division by up to 256. As a result, the clock system has the flexibility to generate diverse MSTR_2X frequencies from 200 MHz down to below 1 kHz.

While deriving the system and peripheral clocks from MSTR_2X, the SIM provides several methods for clock gating to manage and reduce the overall power consumption of the part. These methods include low-power modes RUN/STOP/WAIT and various clock enables (PCEn registers, SDn registers, CLK_DIS, ONCE_EBL). System clocks including the core clock normally operate only in RUN mode. Peripheral clocks operate in RUN or WAIT modes when enabled by their individual clock gating controls in the SIM's PCE registers. Peripheral clocks may be individually overridden to operate in STOP mode using the SIM's SD registers—to operate select peripherals used for recovery from STOP mode back to RUN mode.

9.3.2 Power-Down Modes Overview

The DSC core operates in one of following power-down modes.

Table 9-1. Clock Operation In Power Down Modes

Mode	System Clocks	Peripheral Clocks	Description
RUN	Core and memory clocks enabled	Peripheral clocks enabled	Device is fully functional
WAIT	Core and memory clocks disabled	Peripheral clocks enabled	Core executes WAIT instruction to enter this mode. Wait mode is typically used for power conscious applications. Possible recoveries from WAIT mode to RUN mode are: 1. Any interrupt. 2. Executing a debug mode entry command using the DSC core JTAG interface. 3. Any reset (POR, external, software, COP, and so on).
STOP	Master clock generation in the OCCS remains operational, but the SIM disables the generation of system and peripheral clocks.		Core executes STOP instruction to enter this mode. Possible recoveries from stop mode to RUN mode are: 1. Interrupt from any peripheral configured in SD register to operate in STOP mode. 2. Low voltage interrupt 3. Executing a debug mode entry command using the DSC core JTAG interface. 4. Any reset.

RUN, WAIT and STOP modes provide means of enabling/disabling the peripheral and/or system clocks as a group. Peripherals must be enabled (refer to PCEn registers) to operate in any mode. Once enabled, their standard behavior is to operate in RUN and WAIT modes but to be disabled in STOP mode. However, by asserting a peripheral's STOP disable bit (refer to SDn registers), the peripheral clock continues to operate in STOP mode. This permits selected peripherals to remain operational in STOP mode to produce interrupts which can recover the part from STOP mode back into RUN mode.

System clocks are individually gated in the SIM based on their specific requirements as a function of low-power mode. Clocks specific to the DSC core processor operate only in RUN mode. The DMAEBL field in the SIM Control register determines which low power modes in which clocking to the DMA is enabled. Clocks to system bus slaves including the IPBus interface, the RAM, and the flash memory, will operate in any low power mode in which either the core processor or the DMA are enabled.

The SCIs can be configured to operate at two times the system bus rate using the SCIn_CR control bits.

RUN, WAIT and STOP modes may be combined with the power management features of the PMC, flash memory low power mode feature, and clock generation configuration of the OCCS to provide a broad palette of power control techniques.

9.3.3 STOP and WAIT Mode Disable Function

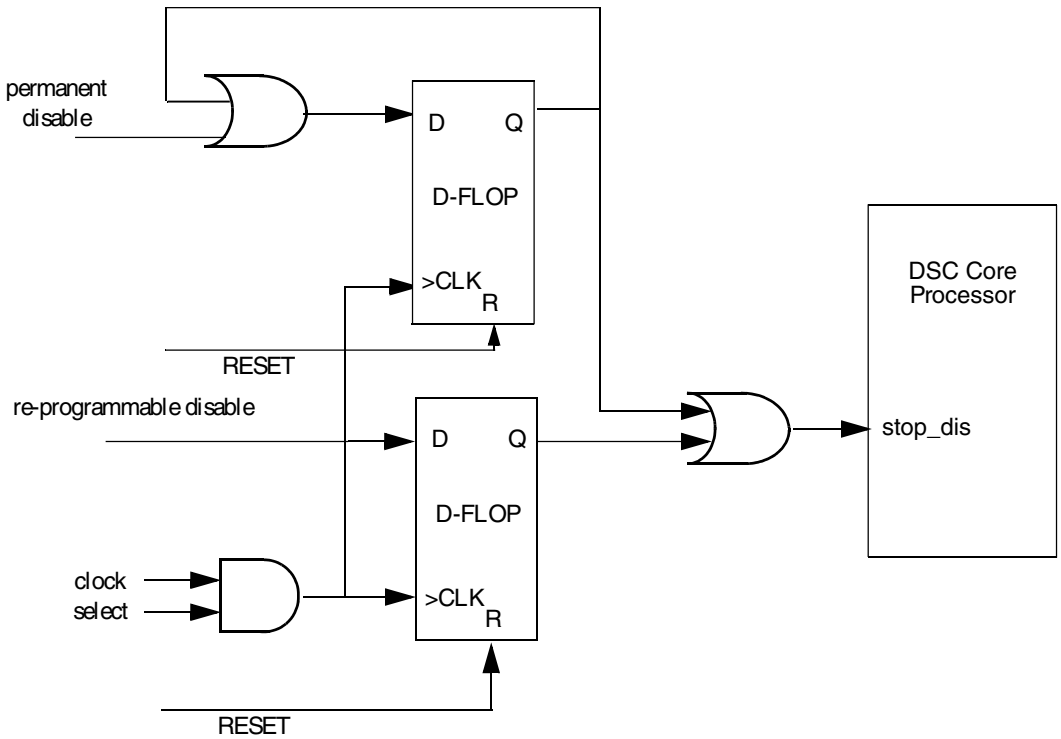


Figure 9-1. STOP Disable Circuit

The core processor supports both STOP and WAIT instructions. Both put the CPU to sleep. The peripheral bus continues to run in WAIT mode, but in STOP mode, only peripherals whose SDn control is asserted continue to run. Entry into STOP or WAIT mode does not affect the OCCS configuration and affects only the generation of system and peripheral clocks using the master clocks from the OCCS. The OCCS may be reconfigured prior to entering STOP or WAIT, if desired, to reduce master clock frequencies and thus the power utilization within the OCCS.

Some applications require the DSC core's STOP/WAIT instructions to be disabled. Control fields are provided in the CTRL register to disable WAIT and/or STOP modes. This setting can be made irrecoverable (recoverable only at the next reset) as illustrated in Figure 9-1, by setting the lock bit within these fields.

9.4 Resets

The SIM supports several sources of reset.

Table 9-2. Sources of Reset

Label	Source of Reset	Timing
EXTR	External Reset	Asynchronous
POR	Power-On-Reset (PMC)	Asynchronous
COP_CPU	COP CPU reset	Asynchronous
COP_LOR	COP Loss of Reference reset	Synchronous
SWR	Software reset (SIM)	Synchronous

9.5 Clocks

All system and peripheral clocks are derived in the SIM by dividing the MSTR_2X clock from OCCS by two. Exceptions include the RAM system clock, which is a gated version of the MSTR_2X clock, and the 2x clock option to the PWMA and SCI modules. These clocks operate at up to the MSTR_2X clock's maximum frequency. The SIM is responsible for stalling individual clocks as a response to holdoff requests from system bus slaves, low power modes, and other clock configuration parameters.

9.6 Interrupts

The SIM generates no interrupts.

Chapter 10
Interrupt Controller (INTC)

10.1 Chip-specific information for this module

10.1.1 Reset/Interrupt Vector Table

The following table shows the interrupt assignment.

Table 10-1. Interrupt Vector Table

Peripheral	Vector #	Priority Level	Vector Base Address	Interrupt	Details	Local Enable	Local Source	Description
Reserved	112	0-2	0xE0					
Reserved	111	0-2	0xDE					
Core	110	-1	0xDC	SWILP		N/A	N/A	SWILP
EWM	109	0-2	0xDA	EWM_INT		CTRL [INTEN]	N/A	EWM_ Indicate
COP	108	0-2	0xD8	COP_INT		CTRL [INTEN]	N/A	COP_ Warning
GPIO A	107	0-2	0xD6	GPIOA		IENR	IPEND	GPIO
GPIO B	106	0-2	0xD4	GPIOB		IENR	IPEND	GPIO
GPIO C	105	0-2	0xD2	GPIOC		IENR	IPEND	GPIO
GPIO D	104	0-2	0xD0	GPIOD		IENR	IPEND	GPIO
GPIO E	103	0-2	0xCE	GPIOE		IENR	IPEND	GPIO
GPIO F	102	0-2	0xCC	GPIOF		IENR	IPEND	GPIO
Reserved	101	0-2	0xCA					
Reserved	100	0-2	0xC8					
Reserved	99	0-2	0xC6					
Reserved	98	0-2	0xC4					
Reserved	97	0-2	0xC2					
Reserved	96	0-2	0xC0					

Table continues on the next page...

Table 10-1. Interrupt Vector Table (continued)

Peripheral	Vector #	Priority Level	Vector Base Address	Interrupt	Details	Local Enable	Local Source	Description
PIT 0	95	0-2	0xBE	PIT0_ROLLOVR		CTRL [PRIE]	CTRL[PRF]	Roll Over
PIT 1	94	0-2	0xBC	PIT1_ROLLOVR		CTRL [PRIE]	CTRL[PRF]	Roll Over
CMP A and OPAMP A	93	0-2	0xBA	CMPA	CMP and OPAMP OR'ed	SCR[IER]	SCR[CFR]	Rising Edge
						SCR[IEF]	SCR[CFF]	Falling Edge
				OPAMPA		CTR[LDCMIE]	CTR[LDCMF]	Load Complete
CMP B and OPAMP B	92	0-2	0xB8	CMPB	CMP and OPAMP OR'ed	SCR[IER]	SCR[CFR]	Rising Edge
						SCR[IEF]	SCR[CFF]	Falling Edge
				OPAMPB		CTR[LDCMIE]	CTR[LDCMF]	Load Completed
CMP C	91	0-2	0xB6	CMPC		SCR[IER]	SCR[CFR]	Rising Edge
						SCR[IEF]	SCR[CFF]	Falling Edge
CMP D	90	0-2	0xB4	CMPD		SCR[IER]	SCR[CFR]	Rising Edge
						SCR[IEF]	SCR[CFF]	Falling Edge
FTFA	89	0-2	0xB2	FTFA_CC		FCNFG [CCIE]	FSTAT [CCIF]	Command Complete
	88	0-2	0xB0	FTFA_RDCOL		FCNFG[RDCOLLIE]	FSTAT [RDCOLERR]	Read Collision (Access) Error
eFlex PWM A	87	0-2	0xAE	eFlexPWMA_CMP0		SM0INTEN[CMPIE]	SM0STS [CMPF]	Submodule 0 Compare
	86	0-2	0xAC	eFlexPWMA_RELOAD0		SM0INTEN[RIE]	SM0STS [RF]	Submodule 0 Reload
	85	0-2	0xAA	eFlexPWMA_CMP1		SM1INTEN[CMPIE]	SM1STS [CMPF]	Submodule 1 Compare
	84	0-2	0xA8	eFlexPWMA_RELOAD1		SM1INTEN[RIE]	SM1STS[RF]	Submodule 1 Reload
	83	0-2	0xA6	eFlexPWMA_CMP2		SM2INTEN[CMPIE]	SM2STS [CMPF]	Submodule 2 Compare
	82	0-2	0xA4	eFlexPWMA_RELOAD2		SM2INTEN[RIE]	SM2STS [RF]	Submodule 2 Reload
	81	0-2	0xA2	eFlexPWMA_CMP3		SM3INTEN[CMPIE]	SM3STS [CMPF]	Submodule 3 Compare
	80	0-2	0xA0	eFlexPWMA_RELOAD3		SM3INTEN[RIE]	SM3STS [RF]	Submodule 3 Reload
	79	0-2	0x9E	eFlexPWMA_CAP	OR'ed	SM0INTEN[CFA1IE]	SM0STS [CFA1]	Submodule 0-3 Input Captures
						SM0INTEN[CFA0IE]	SM0STS [CFA0]	

Table continues on the next page...

Table 10-1. Interrupt Vector Table (continued)

Peripheral	Vector #	Priority Level	Vector Base Address	Interrupt	Details	Local Enable	Local Source	Description
						SM0INTEN[CFB1IE]	SM0STS [CFB1]	
						SM0INTEN[CFB0IE]	SM0STS [CFB0]	
						SM0INTEN[CFX1IE]	SM0STS [CFX1]	
						SM0INTEN[CFX0IE]	SM0STS [CFX0]	
						SM1INTEN[CFA1IE]	SM1STS [CFA1]	
						SM1INTEN[CFA0IE]	SM1STS [CFB1]	
						SM1INTEN[CFB1IE]	SM1STS [CFB0]	
						SM1INTEN[CFB0IE]	SM1STS [CFA0]	
						SM1INTEN[CFX1IE]	SM1STS [CFX1]	
						SM1INTEN[CFX0IE]	SM1STS [CFX0]	
						SM2INTEN[CFA1IE]	SM2STS [CFA1]	
						SM2INTEN[CFA0IE]	SM2STS [CFA0]	
						SM2INTEN[CFB1IE]	SM2STS [CFB1]	
						SM2INTEN[CFB0IE]	SM2STS [CFB0]	
						SM2INTEN[CFX1IE]	SM2STS [CFX1]	
						SM2INTEN[CFX0IE]	SM2STS [CFX0]	
						SM3INTEN[CFA1IE]	SM3STS [CFA1]	
						SM3INTEN[CFA0IE]	SM3STS [CFA0]	
						SM3INTEN[CFB1IE]	SM3STS [CFB1]	
						SM3INTEN[CFB0IE]	SM3STS [CFB0]	
						SM3INTEN[CFX1IE]	SM3STS [CFX1]	
						SM3INTEN[CFX0IE]	SM3STS [CFX0]	

Table continues on the next page...

Table 10-1. Interrupt Vector Table (continued)

Peripheral	Vector #	Priority Level	Vector Base Address	Interrupt	Details	Local Enable	Local Source	Description
	78	0-2	0x9C	eFlexPWMA_RERR	OR'ed	SM0INTEN[REIE]	SM0STS[REF]	Submodule 0-3 Reload Errors
						SM1INTEN[REIE]	SM1STS[REF]	
						SM2INTEN[REIE]	SM2STS[REF]	
						SM3INTEN[REIE]	SM3STS[REF]	
	77	0-2	0x9A	eFlexPWMA_FAULT		FCTRL[FIE]	FSTS[FFLAG]	Fault Condition
Reserved	76	0-2	0x98					
Reserved	75	0-2	0x96					
Reserved	74	0-2	0x94					
Reserved	73	0-2	0x92					
Reserved	72	0-2	0x90					
Reserved	71	0-2	0x8E					
Reserved	70	0-2	0x8C					
Reserved	69	0-2	0x88					
Reserved	68	0-2	0x8A					
Reserved	67	0-2	0x86		O			
Reserved	66	0-2	0x84					
Reserved	65	0-2	0x82					
Reserved	64	0-2	0x80					
Reserved	63	0-2	0x7E					
LPI2C0	62	0-2	0x7C	LPI2C0	OR'ed	MIER[TDIE]	MSR[TDF]	all master and slave interrupts
						MIER[RDIE]	MSR[RDF]	
						MIER[EPIE]	MSR[EPF]	
						MIER[SDIE]	MSR[SDF]	
						MIER[NDIE]	MSR[NDF]	
						MIER[ALIE]	MSR[ALF]	
						MIER[FEIE]	MSR[FEF]	
						MIER[PLTIE]	MSR[PLTF]	
						MIER[DMIE]	MSR[DMF]	
						SIER[TDIE]	SSR[TDF]	
						SIER[RDIE]	SSR[RDF]	
						SIER[AVIE]	SSR[AVF]	
						SIER[TAIE]	SSR[TAF]	
						SIER[RSIE]	SSR[RSF]	
						MIER[SDIE]	SSR[SDF]	

Table continues on the next page...

Table 10-1. Interrupt Vector Table (continued)

Peripheral	Vector #	Priority Level	Vector Base Address	Interrupt	Details	Local Enable	Local Source	Description
						MIER[BEIE]	SSR[BEF]	
						SIER[FEIE],	SSR[FEF]	
						SIER[AM0IE]	SSR[AM0F]	
						SIER[AM1IE]	SSR[AM1F]	
						SIER[GCIE]	SSR[GCF]	
						SIER[SARIE]	SSR[SARF]	
LPI2C1	61	0-2	0x7A	LPI2C1	OR'ed	MIER[TDIE], MIER[RDIE], MIER[EPIE], MIER[SDIE], MIER[NDIE], MIER[ALIE], MIER[FEIE], MIER[PLTIE], MIER[DMIE], SIER[TDIE], SIER[RDIE], SIER[AVIE], SIER[TAIE], SIER[RSIE], MIER[SDIE], MIER[BEIE], SIER[FEIE], SIER[AM0IE], SIER[AM1IE], SIER[GCIE], SIER[SARIE]	MSR[TDF], MSR[RDF], MSR[EPF], MSR[SDF], MSR[NDF], MSR[ALF], MSR[FEF], MSR[PLTF], MSR[DMF], SSR[TDF], SSR[RDF], SSR[AVF], SSR[TAF], SSR[RSF], SSR[SDF], SSR[BEF], SSR[FEF], SSR[AM0F], SSR[AM1F], SSR[GCF], SSR[SARF]	all master and slave interrupts
QSPI	60	0-2	0x78	QSPI_RCV	OR'ed	SPSCR [SPRIE]	SPSCR [SPRF], SPFIFO RFWM]	Receiver Full
						SPSCR [ERRIR]	SPSCR [MODF, OVRF]	Mode Fault, Overflow
	59	0-2	0x76	QSPI_XMIT		SPSCR [SPTIE]	SPSCR [SPTE], SPFIFO [TFWM]	Transmitter Empty
Reserved	58	0-2	0x74					
	57	0-2	0x72					
Reserved	56	0-2	0x70					
	55	0-2	0x6E					
QSCI 0	54	0-2	0x6C	QSCI0_TDRE		CTRL1 [TEIE]	STAT[[TDRE]	Transmit Data Register Empty
	53	0-2	0x6A	QSCI0_TIDLE, _RIDLE	OR'ed	CTRL1 [TIIE], CTRL2 [RIIE]	STAT[TIDLE, RIDLE]	Transmitter and Receiver Idle

Table continues on the next page...

Table 10-1. Interrupt Vector Table (continued)

Peripheral	Vector #	Priority Level	Vector Base Address	Interrupt	Details	Local Enable	Local Source	Description
	52	0-2	0x68	QSCI0_RDFD, _OR, _RIEF	OR'ed	CTRL1 [RFIE], CTRL2[RIEIE]	STAT[RDRF, OR, RIEF]	Receive Data Register Full / Overrun / Active Edge
	51	0-2	0x66	QSCI0_RERR		CTRL1 [REIE]	STAT[OR,NF, FE,PF,LSE]	Receiver Error
QSCI 1	50	0-2	0x64	QSCI1_TDRE		CTRL1 [TEIE]	STAT[[TDRE]	Transmit Data Register Empty
	49	0-2	0x62	QSCI1_TIDLE, _RIDLE	OR'ed	CTRL1 [TIIE], CTRL2 [RIIE]	STAT[TIDLE, RIDLE]	Transmitter and Receiver Idle
	48	0-2	0x60	QSCI1_RDFD, _OR, _RIEF	OR'ed	CTRL1 [RFIE], CTRL2[RIEIE]	STAT[RDRF, OR, RIEF]	Receive Data Register Full / Overrun / Active Edge
	47	0-2	0x5E	QSCI1_RERR		CTRL1 [REIE]	STAT[OR,NF,FE,PF,LSE]	Receiver Error
Reserved	46	0-2	0x5C					
	45	0-2	0x5A					
	44	0-2	0x58					
	43	0-2	0x56					
Reserved	42	0-2	0x54					
Reserved	41	0-2	0x52					
Reserved	40	0-2	0x50					
Reserved	39	0-2	0x4E					
Reserved	38	0-2	0x4C					
Reserved	37	0-2	0x4A					
eDMA CH0	36	0-2	0x48	DMACH0		INT_HALF, INT_MAJ, DONE	NT[INT0]	eDMA CH0 Service Req
eDMA CH1	35	0-2	0x46	DMACH1		INT_HALF, INT_MAJ, DONE	INT[INT1]	eDMA CH1 Service Req
eDMA CH2	34	0-2	0x44	DMACH2		INT_HALF, INT_MAJ, DONE	INT[INT2]	eDMA CH2 Service Req
eDMA CH3	33	0-2	0x42	DMACH3		INT_HALF, INT_MAJ, DONE	INT[INT3]	eDMA CH3 Service Req
eDMA	32	0-2	0x40	DMA_ERR	OR'ed	ERR[ERR3, ERR2, ERR1, ERR0]	EEI[EEI3, EEI2, EEI1, EEI0]	DMA Error
ADC	31	0-2	0x3E	ADC_ERR		CTRL1[ZCIE, LLMTIE,HLMTIE]	STAT[ZCI, LLMT, HLMT]	ADC zero crossing, low limit, and high limit

Table continues on the next page...

Table 10-1. Interrupt Vector Table (continued)

Peripheral	Vector #	Priority Level	Vector Base Address	Interrupt	Details	Local Enable	Local Source	Description
	30	0-2	0x3C	ADC_CC0		CTRL1[EOSIE0], SCHLTEN[SCHLTEN, SCHLTEN2[SCHLTEN], EXPCFG[MAXAIE]	STAT[EOSI0], RDY[RDY], RDY2[RDY], EXPSTAT[MUXAIRQ]	ADC Scan Complete or a group of ADC Conversion Complete in any scan type, or ADCA Expansion Auxiliary Control Scan Complete.
	29[1]	0-2	0x3A	ADC_CC1		CTRL2[EOSIE1], SCHLTEN[SCHLTEN, SCHLTEN2[SCHLTEN], EXPCFG[MAXBIE]	STAT[EOSI1], RDY[RDY], RDY2[RDY], EXPSTAT[MUXBIRQ]	ADCB Scan Complete or a group of ADCB Conversion Complete or ADCB Expansion Auxiliary Control Scan Complete in non-simultaneous parallel scan mode.
TIMER A0	28	0-2	0x38	TMRA_0	TMRA0TCF	SCTRL [TCFIE]	SCTRL [TCFIE]	Timer Compare
					TMRA0TOF	SCTRL [TOFIE]	SCTRL[TOF]	Timer Overflow
					TMRA0IEF	SCTRL [IEFIE]	SCTRL[IEF]	Input Edge Flag
					TMRA0TCF1	CSCTRL [TCF1EN]	CSCTRL [TCF1]	Timer Compare Flag 1
					TMRA0TCF2	CSCTRL [TCF2EN]	CSCTRL [TCF2]	Timer Compare Flag 2
TIMER A1	27	0-2	0x36	TMRA_1	TMRA1TCF	SCTRL [TCFIE]	SCTRL [TCFIE]	Timer Compare
					TMRA1TOF	SCTRL [TOFIE]	SCTRL[TOF]	Timer Overflow
					TMRA1IEF	SCTRL [IEFIE]	SCTRL[IEF]	Input Edge Flag
					TMRA1TCF1	CSCTRL [TCF1EN]	CSCTRL [TCF1]	Timer Compare Flag 1
					TMRA1TCF2	CSCTRL [TCF2EN]	CSCTRL [TCF2]	Timer Compare Flag 2
TIMER A2	26	0-2	0x34	TMRA_2	TMRA2TCF	SCTRL [TCFIE]	SCTRL [TCFIE]	Timer Compare
					TMRA2TOF	SCTRL [TOFIE]	SCTRL [TOF]	Timer Overflow
					TMRA2IEF	SCTRL [IEFIE]	SCTRL [IEF]	Input Edge Flag
					TMRA2TCF1	CSCTRL [TCF1EN]	CSCTRL [TCF1]	Timer Compare Flag 1
					TMRA2TCF2	CSCTRL [TCF2EN]	CSCTRL [TCF2]	Timer Compare Flag 2

Table continues on the next page...

Table 10-1. Interrupt Vector Table (continued)

Peripheral	Vector #	Priority Level	Vector Base Address	Interrupt	Details	Local Enable	Local Source	Description
TIMER A3	25	0-2	0x32	TMRA_3	TMRA3TCF	SCTRL [TCFIE]	SCTRL [TCFIE]	Timer Compare
					TMRA3TOF	SCTRL [TOFIE]	SCTRL [TOF]	Timer Overflow
					TMRA3IEF	SCTRL [IEFIE]	SCTRL [IEF]	Input Edge Flag
					TMRA3TCF1	CSCTRL [TCF1EN]	CSCTRL [TCF1]	Timer Compare Flag 1
					TMRA3 TCF2	CSCTRL [TCF2EN]	CSCTRL [TCF2]	Timer Compare Flag 2
Reserved	24	0-2	0x30					
Reserved	23	0-2	0x2E					
Reserved	22	0-2	0x2C					
Reserved	21	0-2	0x2A					
OCCS	20	1-3	0x28	OCCS	OCCSLOLI1	CTRL[PLLIE1]	STAT[LCK1]	PLL Loss of Lock 1
					OCCSLOLI0	CTRL[PLLIE0]	STAT[LCK0]	PLL Loss of Lock 0
					OCCSLOCI	CTRL[LOCIE]	STAT[LOCI]	PLL Loss of Reference Clock
PMC	19	1-3	0x26	LVI1		CTRL[IOLVIE, CLVIE]	STAT[IOLVS, CLVS]	Low Voltage Interrupt
XBARA	18	1-3	0x24	XBARA		CTRL0[IEN0]| CTRL0[IEN1]| CTRL1[IEN0]| CTRL1[IEN1]	CTRL0[STS0]| CTRL0[STS1]| CTRL1[STS0]| CTRL1[STS1]	Crossbar Interrupt
Core	17	0	0x22	SWI0		N/A	N/A	SW Interrupt 0
Core	16	1	0x20	SWI1		N/A	N/A	SW Interrupt 1
Core	15	2	0x1E	SWI2		N/A	N/A	SW Interrupt 2
Reserved	14	1-3	0x1C	Reserved				
Reserved	13	1-3	0x1A	Reserved				
Reserved	12	1-3	0x18	Reserved				
Core	11	1-3	0x16	BUS_ERR		MCM_CFIER[ECFEI]	Core	Bus Error Interrupt
Core	10	1-3	0x14	RX_REG		IPR0[RX_REG]	Core	EOnCE Receive Register Full
Core	9	1-3	0x12	TX_REG		IPR0[TX_REG]	Core	EOnCE Transmit Register Empty
Core	8	1-3	0x10	TRBUF		IPR0[TRBUF]	Core	EOnCE Trace Buffer Interrupt

Table continues on the next page...

Table 10-1. Interrupt Vector Table (continued)

Peripheral	Vector #	Priority Level	Vector Base Address	Interrupt	Details	Local Enable	Local Source	Description
Core	7	1-3	0x0E	BKPT		IPR0[BKPT_U]	Core	EOnCE Breakpoint Unit
Core	6	1-3	0x0C	STPCNT		IPR0[STPCNT]	Core	EOnCE Step Counter Interrupt
Core	5	3	0x0A	MISALIGNED		N/A	Core	Misaligned Data Access
Core	4	3	0x08	OVERFLOW		N/A	Core	Hardware Stack Overflow
Core	3	3	0x06	SWI3		N/A	Core	SW Interrupt 3
Core	2	3	0x04	ILLEGAL-OP		N/A	Core	Illegal Instruction
Core	1		0x02	COP_RESET				Reserved for COP Reset Overlay
Core	0		0x00	HW_RESET				Reserved for Reset Overlay

1. This interrupt is only able to assertt in non-simultaneous parallel scan mode

COP, GPIOs, SCIs, SPIs, LPI2C, PITs, CMPs, RESET support to wake up core via interrupt controller when the clock to that module is enabled.

10.2 Introduction

The Interrupt Controller (INTC) module arbitrates among the various interrupt requests (IRQs). The module supports unique interrupt vectors and programmable interrupt priority. It signals to the DSC core when an interrupt of sufficient priority exists and to what address to jump to service this interrupt.

10.2.1 References

- DSP56800E DSC Core Reference Manual

10.2.2 Features

The INTC module design has these distinctive features:

- Programmable priority levels for each IRQ

- Two programmable fast interrupts
- Notification to System Integration Module (SIM) to restart clocks when exiting wait and stop modes
- Driving of initial address on the address bus after reset

10.2.3 Modes of Operation

The INTC module design has these major modes of operation: functional mode and wait and stop modes.

- Functional mode

  The INTC is in this mode by default.

- Wait and stop mode operation

  In wait and stop modes, the system clocks and the core are turned off. The INTC signals a pending IRQ to the System Integration Module (SIM) to restart the clocks and service the IRQ. An IRQ can wake up the core only if the IRQ is enabled prior to entering wait or stop mode.

10.2.4 Block Diagram

The following figure is a block diagram of the INTC module.

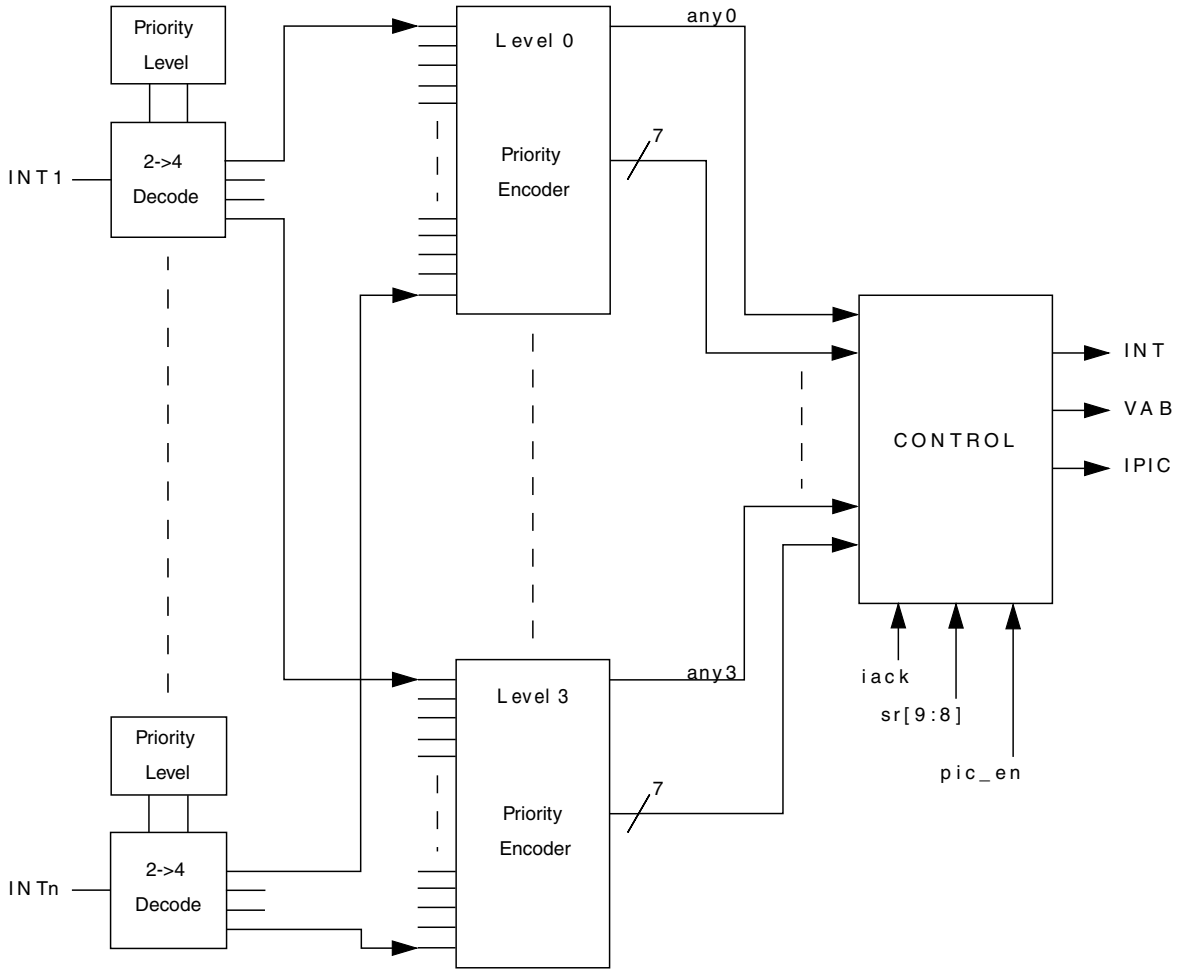


Figure 10-1. Interrupt Controller Block Diagram

10.3 Memory Map and Registers

This module uses 16-bit word addressing.

INTC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E300	Interrupt Priority Register 0 (INTC_IPR0)	16	R/W	0000h	10.3.1/230
E301	Interrupt Priority Register 1 (INTC_IPR1)	16	R/W	0000h	10.3.2/232
E302	Interrupt Priority Register 2 (INTC_IPR2)	16	R/W	0000h	10.3.3/233
E303	Interrupt Priority Register 3 (INTC_IPR3)	16	R/W	0000h	10.3.4/234
E304	Interrupt Priority Register 4 (INTC_IPR4)	16	R/W	0000h	10.3.5/235
E305	Interrupt Priority Register 5 (INTC_IPR5)	16	R/W	0000h	10.3.6/236
E306	Interrupt Priority Register 6 (INTC_IPR6)	16	R/W	0000h	10.3.7/238

Table continues on the next page...

INTC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E308	Interrupt Priority Register 8 (INTC_IPR8)	16	R/W	0000h	10.3.8/239
E309	Interrupt Priority Register 9 (INTC_IPR9)	16	R/W	0000h	10.3.9/240
E30A	Interrupt Priority Register 10 (INTC_IPR10)	16	R/W	0000h	10.3.10/241
E30B	Interrupt Priority Register 11 (INTC_IPR11)	16	R/W	0000h	10.3.11/243
E30C	Interrupt Priority Register 12 (INTC_IPR12)	16	R/W	0000h	10.3.12/244
E30D	Vector Base Address Register (INTC_VBA)	16	R/W	0000h	10.3.13/245
E30E	Fast Interrupt 0 Match Register (INTC_FIM0)	16	R/W	0000h	10.3.14/246
E30F	Fast Interrupt 0 Vector Address Low Register (INTC_FIVAL0)	16	R/W	0000h	10.3.15/246
E310	Fast Interrupt 0 Vector Address High Register (INTC_FIVAH0)	16	R/W	0000h	10.3.16/247
E311	Fast Interrupt 1 Match Register (INTC_FIM1)	16	R/W	0000h	10.3.17/247
E312	Fast Interrupt 1 Vector Address Low Register (INTC_FIVAL1)	16	R/W	0000h	10.3.18/248
E313	Fast Interrupt 1 Vector Address High Register (INTC_FIVAH1)	16	R/W	0000h	10.3.19/248
E314	IRQ Pending Register 0 (INTC_IRQP0)	16	R	FFFFh	10.3.20/248
E315	IRQ Pending Register 1 (INTC_IRQP1)	16	R	FFFFh	10.3.21/249
E316	IRQ Pending Register 2 (INTC_IRQP2)	16	R	FFFFh	10.3.22/249
E317	IRQ Pending Register 3 (INTC_IRQP3)	16	R	FFFFh	10.3.23/250
E318	IRQ Pending Register 4 (INTC_IRQP4)	16	R	FFFFh	10.3.24/250
E319	IRQ Pending Register 5 (INTC_IRQP5)	16	R	FFFFh	10.3.25/251
E31A	IRQ Pending Register 6 (INTC_IRQP6)	16	R	FFFFh	10.3.26/251
E31B	Control Register (INTC_CTRL)	16	R/W	001Ch	10.3.27/252

10.3.1 Interrupt Priority Register 0 (INTC_IPR0)

Address: E300h base + 0h offset = E300h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved		Reserved		BUS_ERR		RX_REG		TX_REG		TRBUF		BKPT		STPCNT	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR0 field descriptions

Field	Description
15–14 Reserved	This field is reserved. Do not modify this bitfield.
13–12 Reserved	This field is reserved. Do not modify this bitfield.

Table continues on the next page...

INTC_IPR0 field descriptions (continued)

Field	Description
11–10 BUS_ERR	Bus Error Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
9–8 RX_REG	EOnCE Receive Register Full Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
7–6 TX_REG	EOnCE Transmit Register Empty Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
5–4 TRBUF	EOnCE Trace Buffer Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
3–2 BKPT	EOnCE Breakpoint Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
STPCNT	EOnCE Step Counter Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1

Table continues on the next page...

INTC_IPR0 field descriptions (continued)

Field	Description
	10 IRQ Priority Level 2 11 IRQ Priority Level 3

10.3.2 Interrupt Priority Register 1 (INTC_IPR1)

Address: E300h base + 1h offset = E301h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved								OCCS		LVI1		XBARA		Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR1 field descriptions

Field	Description
15–8 Reserved	This field is reserved.
7–6 OCCS	PLL Loss of Reference or Change in Lock Status Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
5–4 LVI1	Low Voltage Detector Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
3–2 XBARA	Inter-Peripheral Crossbar Switch A (XBARA) Interrupt Priority Level These fields are used to set the interrupt priority levels for certain IRQs. These IRQs are limited to priorities 1-3 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 1 10 IRQ Priority Level 2 11 IRQ Priority Level 3
Reserved	This field is reserved.

10.3.3 Interrupt Priority Register 2 (INTC_IPR2)

Address: E300h base + 2h offset = E302h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	DMA_ERR		ADC_ERR		ADC_CC0		ADC_CC1		TMRA_0		TMRA_1		TMRA_2		TMRA_3	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR2 field descriptions

Field	Description
15–14 DMA_ERR	DMA Error Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 ADC_ERR	ADC_CYC Zero Crossing, High Limit, or Low Limit Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–10 ADC_CC0	ADC_CYC Conversion Complete Interrupt Priority Level (any scan type except converter B in non-simultaneous parallel scan mode) These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 ADC_CC1	ADC_CYC Conversion Complete Interrupt Priority Level (converter B in non-simultaneous parallel scan mode) These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 TMRA_0	Timer A Channel 0 Interrupt Priority Level

Table continues on the next page...

INTC_IPR2 field descriptions (continued)

Field	Description
	These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 TMRA_1	Timer A Channel 1 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
3–2 TMRA_2	Timer A Channel 2 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
TMRA_3	Timer A Channel 3 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

10.3.4 Interrupt Priority Register 3 (INTC_IPR3)

Address: E300h base + 3h offset = E303h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved				Reserved				DMACH0		DMACH1		DMACH2		DMACH3	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR3 field descriptions

Field	Description
15–12 Reserved	This field is reserved.

Table continues on the next page...

INTC_IPR3 field descriptions (continued)

Field	Description
11–8 Reserved	This field is reserved.
7–6 DMACH0	DMA Channel 0 Service Request Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 DMACH1	DMA Channel 1 Service Request Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
3–2 DMACH2	DMA Channel 2 Service Request Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
DMACH3	DMA Channel 3 Service Request Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

10.3.5 Interrupt Priority Register 4 (INTC_IPR4)

Address: E300h base + 4h offset = E304h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	QSCI1_RCV		QSCI1_ RERR		Reserved								Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR4 field descriptions

Field	Description
15–14 QSCI1_RCV	QSCI 1 Receive Data Register Full Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 QSCI1_RERR	QSCI 1 Receiver Error Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–4 Reserved	This field is reserved.
Reserved	This field is reserved.

10.3.6 Interrupt Priority Register 5 (INTC_IPR5)

Address: E300h base + 5h offset = E305h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved		Reserved		QSCI0_ TDRE		QSCI0_TRID LE		QSCI0_RCV		QSCI0_ RERR		QSCI1_ TDRE		QSCI1_TRID LE	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR5 field descriptions

Field	Description
15–14 Reserved	This field is reserved.
13–12 Reserved	This field is reserved.
11–10 QSCI0_TDRE	QSCI 0 Transmit Data Register Empty Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

Table continues on the next page...

INTC_IPR5 field descriptions (continued)

Field	Description
9–8 QSCI0_TRIDLE	QSCI 0 Transmitter and Receiver Idle Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 QSCI0_RCV	QSCI 0 Receive Data Register Full Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 QSCI0_RERR	QSCI0 Receiver Error Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
3–2 QSCI1_TDRE	QSCI 1 Transmit Data Register Empty Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
QSCI1_TRIDLE	QSCI 1 Transmitter and Receiver Idle Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

10.3.7 Interrupt Priority Register 6 (INTC_IPR6)

Address: E300h base + 6h offset = E306h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved				LPI2C0		LPI2C1		QSPI0_RCV		QSPI0_XMIT		Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR6 field descriptions

Field	Description
15–12 Reserved	This field is reserved.
11–10 LPI2C0	I2C0 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 LPI2C1	I2C1 Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 QSPI0_RCV	QSPI0 Receiver Full Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 QSPI0_XMIT	QSPI0 Transmitter Empty Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
Reserved	This field is reserved.

10.3.8 Interrupt Priority Register 8 (INTC_IPR8)

Address: E300h base + 8h offset = E308h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	PWMA_ RELOAD3		PWMA_CAP		PWMA_ RERR		PWMA_ FAULT		Reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR8 field descriptions

Field	Description
15–14 PWMA_ RELOAD3	PWMA Submodule 3 Reload Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 PWMA_CAP	PWMA Submodule Capture Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. The IRQ represented by this field is a logical OR of input captures for PWMA's submodules 0-3: PWMA_CAP3 | PWMA_CAP2 | PWMA_CAP1 | PWMA_CAP0. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–10 PWMA_RERR	PWMA Reload Error Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 PWMA_FAULT	PWMA Fault Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
Reserved	This field is reserved.

10.3.9 Interrupt Priority Register 9 (INTC_IPR9)

Address: E300h base + 9h offset = E309h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	FTFA_ RDCOL		PWMA_ CMP0		PWMA_ RELOAD0		PWMA_ CMP1		PWMA_ RELOAD1		PWMA_ CMP2		PWMA_ RELOAD2		PWMA_ CMP3	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR9 field descriptions

Field	Description
15–14 FTFA_RDCOL	FTFA Access Error Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 PWMA_CMP0	PWMA Submodule 0 Compare Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–10 PWMA_ RELOAD0	PWMA Submodule 0 Reload Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 PWMA_CMP1	PWMA Submodule 1 Compare Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 PWMA_ RELOAD1	PWMA Submodule 1 Reload Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default.

Table continues on the next page...

INTC_IPR9 field descriptions (continued)

Field	Description
	00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 PWMA_CMP2	PWMA Submodule 2 Compare Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
3–2 PWMA_ RELOAD2	PWMA Submodule 2 Reload Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
PWMA_CMP3	PWMA Submodule 3 Compare Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

10.3.10 Interrupt Priority Register 10 (INTC_IPR10)

Address: E300h base + Ah offset = E30Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved		PIT0_ ROLLOVR		PIT1_ ROLLOVR		CMPA_OPA MPA		CMPB_OPA MPB		CMPC		CMPD		FTFA_CC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR10 field descriptions

Field	Description
15–14 Reserved	This field is reserved.
13–12 PIT0_ROLLOVR	PIT0 Roll Over Interrupt Priority Level

Table continues on the next page...

INTC_IPR10 field descriptions (continued)

Field	Description
	These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–10 PIT1_ROLLOVR	PIT1 Roll Over Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 CMPA_OPAMPA	Comparator A / OPAMP A Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 CMPB_OPAMPB	Comparator B / OPAMP B Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 CMPC	Comparator C Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
3–2 CMPD	Comparator D Interrupt Priority Level

Table continues on the next page...

INTC_IPR10 field descriptions (continued)

Field	Description
	These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
FTFA_CC	FTFA Command Complete Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

10.3.11 Interrupt Priority Register 11 (INTC_IPR11)

Address: E300h base + Bh offset = E30Bh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	GPIOD		GPIOE		GPIOF		Reserved		Reserved						Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR11 field descriptions

Field	Description
15–14 GPIOD	GPIO D Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
13–12 GPIOE	GPIO E Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
11–10 GPIOF	GPIO F Interrupt Priority Level

Table continues on the next page...

INTC_IPR11 field descriptions (continued)

Field	Description
	These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
9–8 Reserved	This field is reserved.
7–2 Reserved	This field is reserved.
Reserved	This field is reserved.

10.3.12 Interrupt Priority Register 12 (INTC_IPR12)

Address: E300h base + Ch offset = E30Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved		Reserved		Reserved		EWM_INT		COP_INT		GPIOA		GPIOB		GPIOC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IPR12 field descriptions

Field	Description
15–14 Reserved	This field is reserved.
13–12 Reserved	This field is reserved.
11–10 Reserved	This field is reserved.
9–8 EWM_INT	External Watchdog Monitor Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
7–6 COP_INT	COP Watchdog Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0

Table continues on the next page...

INTC_IPR12 field descriptions (continued)

Field	Description
	10 IRQ Priority Level 1 11 IRQ Priority Level 2
5–4 GPIOA	GPIO A Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
3–2 GPIOB	GPIO B Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2
GPIOC	GPIO C Interrupt Priority Level These fields are used to set the interrupt priority levels for certain peripheral IRQs. These IRQs are limited to priorities 0-2 and are disabled by default. 00 IRQ disabled (default) 01 IRQ Priority Level 0 10 IRQ Priority Level 1 11 IRQ Priority Level 2

10.3.13 Vector Base Address Register (INTC_VBA)

Address: E300h base + Dh offset = E30Dh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved			VECTOR_BASE_ADDRESS												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_VBA field descriptions

Field	Description
15–13 Reserved	This field is reserved.
VECTOR_ BASE_ ADDRESS	Interrupt Vector Base Address The value in this register is used as the upper 13 bits of the interrupt vector VAB[20:0]. The lower 8 bits are determined based on the highest priority interrupt and are then appended onto the VECTOR_BASE_ADDRESS before presenting the full Vector Address Bus [VAB] to the Core.

Table continues on the next page...

INTC_VBA field descriptions (continued)

Field	Description
	NOTE: After power-on reset, the device must boot from program flash at 0x00_0000, so VBA always resets to zeros.

10.3.14 Fast Interrupt 0 Match Register (INTC_FIM0)

Address: E300h base + Eh offset = E30Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved								FAST_INTERRUPT_0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_FIM0 field descriptions

Field	Description
15–7 Reserved	This field is reserved.
FAST_ INTERRUPT_0	Fast Interrupt 0 Vector Number These values are used to declare which two IRQs will be Fast Interrupts. Fast Interrupts vector directly to a service routine based on values in the Fast Interrupt Vector Address registers without having to go to a jump table first. IRQs used as fast interrupts MUST be set to priority level 2. Unexpected results will occur if a fast interrupt vector is set to any other priority. Fast interrupts automatically become the highest priority level 2 interrupt regardless of their location in the interrupt table prior to being declared as fast interrupt. Fast interrupt 0 has priority over fast interrupt 1. To determine the vector number of each IRQ, refer to the vector table in the memory section of the system specification in this document.

10.3.15 Fast Interrupt 0 Vector Address Low Register (INTC_FIVAL0)

This low register is combined with the corresponding high register to form the 21-bit vector address for the fast interrupt that is defined in the corresponding match register.

Address: E300h base + Fh offset = E30Fh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	FI_0_VECTOR_ADDRESS_LOW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_FIVAL0 field descriptions

Field	Description
FI_0_VECTOR_ ADDRESS_LOW	Lower 16 bits of vector address for fast interrupt 0

10.3.16 Fast Interrupt 0 Vector Address High Register (INTC_FIVAH0)

This high register is combined with the corresponding low register to form the 21-bit vector address for the fast interrupt that is defined in the corresponding match register.

Address: E300h base + 10h offset = E310h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved											FI_0_VECTOR_ADDRESS_HIGH				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_FIVAH0 field descriptions

Field	Description
15–5 Reserved	This field is reserved.
FI_0_VECTOR_ ADDRESS_HIGH	Upper 5 bits of vector address for fast interrupt 0

10.3.17 Fast Interrupt 1 Match Register (INTC_FIM1)

Address: E300h base + 11h offset = E311h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved									FAST_INTERRUPT_1						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_FIM1 field descriptions

Field	Description
15–7 Reserved	This field is reserved.
FAST_ INTERRUPT_1	Fast Interrupt 1 Vector Number These values are used to declare which two IRQs will be Fast Interrupts. Fast Interrupts vector directly to a service routine based on values in the Fast Interrupt Vector Address registers without having to go to a jump table first. IRQs used as fast interrupts MUST be set to priority level 2. Unexpected results will occur if a fast interrupt vector is set to any other priority. Fast interrupts automatically become the highest priority level 2 interrupt regardless of their location in the interrupt table prior to being declared as fast interrupt. Fast interrupt 0 has priority over fast interrupt 1. To determine the vector number of each IRQ, refer to the vector table in the memory section of the system specification in this document.

10.3.18 Fast Interrupt 1 Vector Address Low Register (INTC_FIVAL1)

This low register is combined with the corresponding high register to form the 21-bit vector address for the fast interrupt that is defined in the corresponding match register.

Address: E300h base + 12h offset = E312h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	FI_1_VECTOR_ADDRESS_LOW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_FIVAL1 field descriptions

Field	Description
FI_1_VECTOR_ ADDRESS_LOW	Lower 16 bits of vector address for fast interrupt 1

10.3.19 Fast Interrupt 1 Vector Address High Register (INTC_FIVAH1)

This high register is combined with the corresponding low register to form the 21-bit vector address for the fast interrupt that is defined in the corresponding match register.

Address: E300h base + 13h offset = E313h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved											FI_1_VECTOR_ADDRESS_HIGH				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_FIVAH1 field descriptions

Field	Description
15–5 Reserved	This field is reserved.
FI_1_VECTOR_ ADDRESS_HIGH	Upper 5 bits of vector address for fast interrupt 1

10.3.20 IRQ Pending Register 0 (INTC_IRQP0)

Address: E300h base + 14h offset = E314h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PENDING[16:2]															1
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

INTC_IRQP0 field descriptions

Field	Description
15–1 PENDING[16:2]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 onward. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.

10.3.21 IRQ Pending Register 1 (INTC_IRQP1)

Address: E300h base + 15h offset = E315h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PENDING[32:17]															
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

INTC_IRQP1 field descriptions

Field	Description
PENDING[32:17]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 onward. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number

10.3.22 IRQ Pending Register 2 (INTC_IRQP2)

Address: E300h base + 16h offset = E316h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PENDING[48:33]															
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

INTC_IRQP2 field descriptions

Field	Description
PENDING[48:33]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 onward.

INTC_IRQP2 field descriptions (continued)

Field	Description
	0 IRQ pending for this vector number 1 No IRQ pending for this vector number

10.3.23 IRQ Pending Register 3 (INTC_IRQP3)

Address: E300h base + 17h offset = E317h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PENDING[64:49]															
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

INTC_IRQP3 field descriptions

Field	Description
PENDING[64:49]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 onward. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number

10.3.24 IRQ Pending Register 4 (INTC_IRQP4)

Address: E300h base + 18h offset = E318h

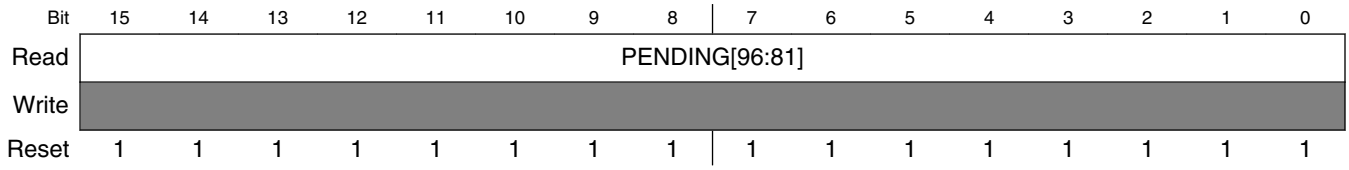
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PENDING[80:65]															
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

INTC_IRQP4 field descriptions

Field	Description
PENDING[80:65]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 onward. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number

10.3.25 IRQ Pending Register 5 (INTC_IRQP5)

Address: E300h base + 19h offset = E319h

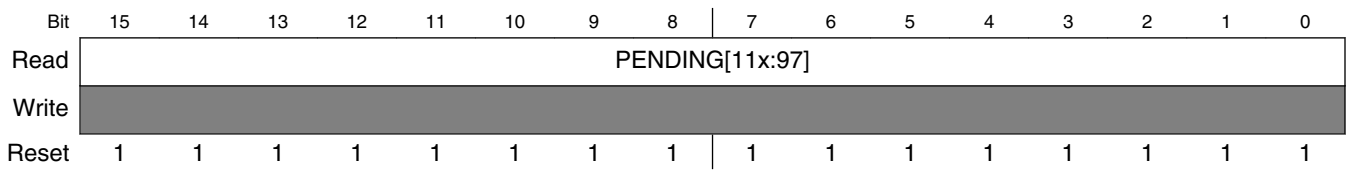


INTC_IRQP5 field descriptions

Field	Description
PENDING[96:81]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 onward. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number

10.3.26 IRQ Pending Register 6 (INTC_IRQP6)

Address: E300h base + 1Ah offset = E31Ah



INTC_IRQP6 field descriptions

Field	Description
PENDING[11x:97]	Pending IRQs These registers combine to represent the pending IRQs for interrupt vector numbers 2 onward. **NOTE:** For interrupt vector number 110, the edge triggered IRQ's status shown by the corresponding INTC_IRQPn[PENDING] bit depends on when the IRQ Pending register is read. Before the ISR is entered, the PENDING bit shows that the IRQ is pending. After the ISR is entered (also with the corresponding Interrupt pending bit cleared), the PENDING bit shows that the IRQ is not pending. 0 IRQ pending for this vector number 1 No IRQ pending for this vector number

10.3.27 Control Register (INTC_CTRL)

Address: E300h base + 1Bh offset = E31Bh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	INT	IPIC		VAB							INT_DIS	1			0	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0

INTC_CTRL field descriptions

Field	Description
15 INT	Interrupt This bit reflects the state of the interrupt to the core. 0 No interrupt is being sent to the core. 1 An interrupt is being sent to the core.
14–13 IPIC	Interrupt Priority Level These bits reflect the new interrupt priority level bits being sent to the Core. These bits indicate the priority level needed for a new IRQ to interrupt the current interrupt being sent to the Core. This field is only updated when the core jumps to a new interrupt service routine. 00 Required nested exception priority levels are 0, 1, 2, or 3. 01 Required nested exception priority levels are 1, 2, or 3. 10 Required nested exception priority levels are 2 or3. 11 Required nested exception priority level is 3.
12–6 VAB	Vector number This field shows bits [7:1] of the Vector Address Bus used at the time the last IRQ was taken. In the case of a fast interrupt, it shows the lower address bits of the jump address. This field is only updated when the core jumps to a new interrupt service routine.
5 INT_DIS	Interrupt disable This bit allows the user to disable all interrupts. 0 Normal operation. (default) 1 All interrupts disabled.
4–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

10.4 Functional Description

The Interrupt Controller is a slave on the IPS bus. It contains registers that allow each of the interrupt sources to be set to one of four priority levels (excluding certain interrupts that have fixed priority). All of the interrupt requests of a given level are priority encoded to determine the lowest numerical value of the active interrupt requests for that level. Within a given priority level, vector number 0 is the highest priority, and vector number *n*-1 (where *n* is the total number of interrupt sources) is the lowest priority.

10.4.1 Normal Interrupt Handling

After the INTC determines that an interrupt is to be serviced and which interrupt has the highest priority, an interrupt vector address is generated. Normal interrupt handling concatenates the vector base address (VBA) and the vector number to determine the vector address. In this way, an offset into the vector table is generated for each interrupt.

10.4.2 Interrupt Nesting

Interrupt exceptions may be nested to allow the servicing of an IRQ with higher priority than the current exception. The DSC core controls the masking of interrupt priority levels by setting the I0 and I1 bits in its status register (SR).

Table 10-2. Interrupt Mask Bit Settings in Core Status Register

I1 (SR[9])	I0 (SR[8])	Exceptions Permitted	Exceptions Masked
0	0	Priorities 0, 1, 2, 3	None
0	1	Priorities 1, 2, 3	Priority 0
1	0	Priorities 2, 3	Priorities 0, 1
1	1	Priority 3	Priorities 0, 1, 2

The IPIC field of the INTC module's CTRL register reflects the state of the priority level that is presented to the DSC core.

Table 10-3. Interrupt Priority Level Field Settings

IPIC	Current Interrupt Priority Level	Required Nested Exception Priority
00	No interrupt or SWILP	Priorities 0, 1, 2, 3
01	Priority 0	Priorities 1, 2, 3
10	Priority 1	Priorities 2, 3

Table continues on the next page...

Table 10-3. Interrupt Priority Level Field Settings (continued)

IPIC	Current Interrupt Priority Level	Required Nested Exception Priority
11	Priorities 2 or 3	Priority 3

10.4.3 Fast Interrupt Handling

Fast interrupt processing is described in section 9.3.2.2 of the *DSP56800E DSC Core Reference Manual*. The Interrupt Controller recognizes fast interrupts before the core does.

A fast interrupt is defined (to the INTC) by:

1. Setting the priority of the interrupt as level 2, using the appropriate field in the IPR registers.
2. Setting the FIMn register to the appropriate vector number.
3. Setting the FIVALn and FIVAHn registers with the address of the code for the fast interrupt.

When an interrupt occurs, its vector number is compared with the FIM0 and FIM1 register values. If a match occurs, and if the interrupt priority is level 2, the INTC handles the interrupt as a fast interrupt. The INTC takes the vector address from the appropriate FIVALn and FIVAHn registers, instead of generating an address that is an offset from the vector base address (VBA).

The core then fetches the instruction from the indicated vector address. If the instruction is not a JSR, the core starts its fast interrupt handling.

10.5 Interrupts

This module does not produce interrupts.

Chapter 11
Enhanced Direct Memory Access (eDMA)

11.1 Overview

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes:

- A DMA engine that performs:
  - Source address and destination address calculations
  - Data-movement operations
- Local memory containing transfer control descriptors for each of the 4 channels

11.1.1 Block diagram

The following figure illustrates the components of the eDMA system, including the eDMA module ("engine").

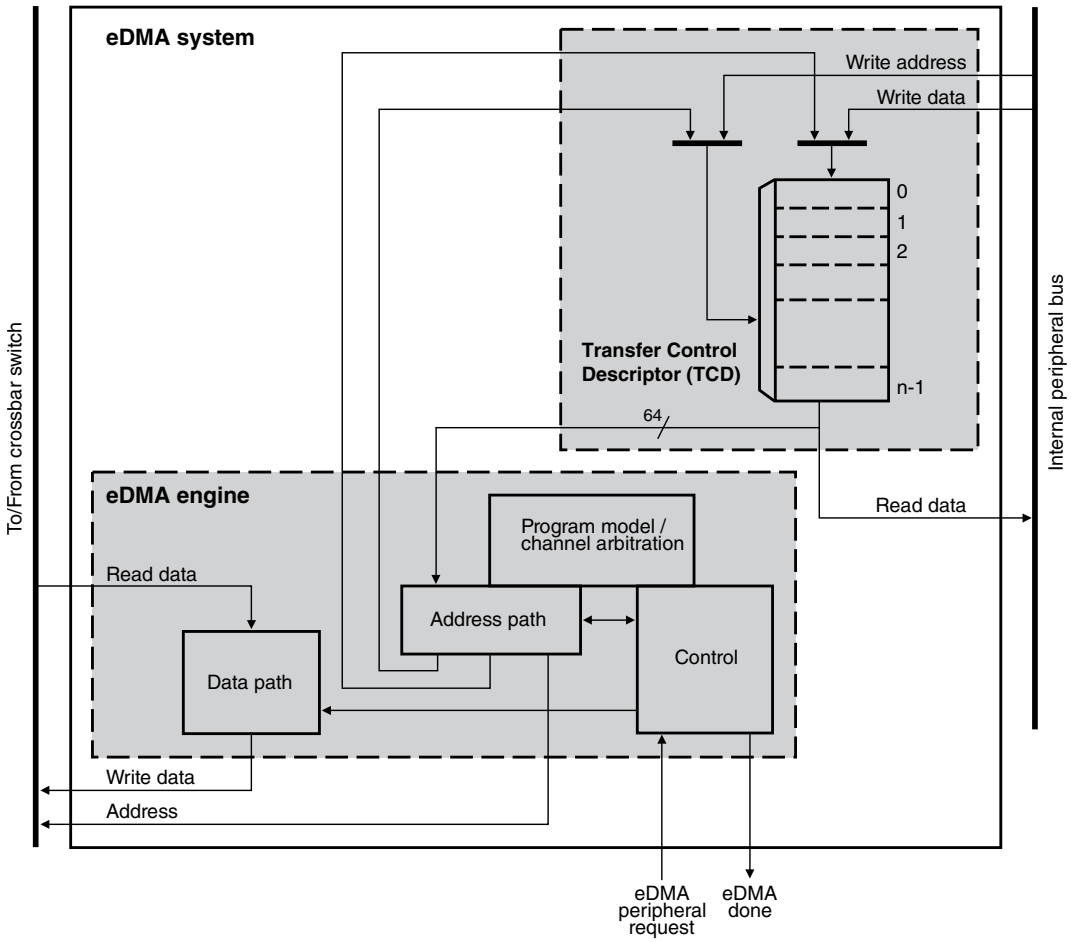


Figure 11-1. Block diagram

11.1.2 Block parts

The eDMA module comprises two major modules: the eDMA engine and the transfer-control descriptor local memory.

Table 11-1 describes the eDMA engine submodules.

Table 11-1. eDMA engine submodules

Submodule	Function
Address path	The address path block: • Provides registered versions of two channel Transfer Control Descriptors (TCDs)—channel x (normal start) and channel y (preemption start) • Manages all master bus-address calculations All channels provide the same functionality. This structure enables preemption of data transfers associated with an active channel (after completion of a read/write sequence) if the eDMA engine asserts a higher priority channel activation.

Table continues on the next page...

Table 11-1. eDMA engine submodules (continued)

Submodule	Function
	After eDMA activates a channel, it runs until the minor loop completes, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI*n*[ECP]) in which the eDMA engine can preempt a large data move operation to minimize the time another channel stalls. When the eDMA engine selects a channel to execute, it reads the contents of the channel TCD from local memory and loads it into one of the following: • The address path channel x registers (normal start) • The address path channel y registers (preemption start) After the minor loop execution completes, the address path hardware writes the new values for the TCD*n*_{SADDR, DADDR, CITER} back to local memory. If the major iteration count completes, the eDMA engine performs additional processing, including: • Final address pointer updates • Reloading the TCD*n*_CITER field • A possible fetch of the next TCD*n* from memory as part of a scatter/gather operation.
Data path	The data path block implements the bus master read/write data path. It includes a data buffer and the necessary multiplex logic to support any required data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output. The address and data path modules directly support the two-stage pipelined internal bus. The address path module represents the first stage of the bus pipeline (address phase). The data path module implements the second stage of the pipeline (data phase).
Programming model/ channel arbitration	This block implements: • The first section of the eDMA programming model • Channel arbitration logic The programming model registers connect to the chip's internal peripheral bus. The eDMA peripheral request inputs and interrupt request outputs also connect to this block (via control logic).
Control	The control block provides all control functions for the eDMA engine. For data transfers in which the source size (SSIZE) and destination size (DSIZE) are equal, the eDMA engine performs a series of source read/destination write operations until it has transferred the number of bytes specified in the minor loop byte count (NBYTES). For TCDs in which the source and destination sizes are not equal, the eDMA engine executes multiple accesses of the smaller size data for each reference of the larger size. For example, if the source size (SSIZE) references 16-bit data and the destination size (DSIZE) is 32-bit data, eDMA performs two reads, then one 32-bit write.

Table 11-2 explains the partitioning of the TCD local memory.

Table 11-2. Transfer control descriptor memory

Submodule	Description
Memory controller	The Memory controller logic implements the required dual-ported controller, managing accesses from the eDMA engine as well as references from the internal peripheral bus. If simultaneous accesses occur, the eDMA engine receives priority and the peripheral transaction stalls.
Memory array	The Memory array provides TCD storage for the transfer profile for each channel.

11.1.3 Features

The eDMA module is a highly programmable data-transfer engine optimized to minimize any required intervention from the host processor. Use it for applications where you statically know the size of the data to be transferred and do not define the size within the transferred data itself. The eDMA module features:

- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source and destination addresses and transfer size
  - Support for enhanced addressing modes
- 4-channel implementation performs complex data transfers with minimal intervention from a host processor
  - Connections to the crossbar switch (AXBS) for bus mastering the data movement
- TCD supports two-deep, nested transfer operations
  - 32-byte TCD stored in local memory for each channel
  - An inner data transfer loop defined by a minor byte transfer count
  - An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
  - Explicit software initiation
  - Initiation via a channel-to-channel linking mechanism for continuous transfers
  - Peripheral-paced hardware requests, one per channel
- Fixed-priority and round-robin channel arbitration
- Channel completion notification via programmable interrupt requests
  - One interrupt per channel. eDMA engine can generate an interrupt when major iteration count completes
  - Programmable error terminations per channel and logically summed together to form one error interrupt to the interrupt controller
- Programmable support for scatter/gather DMA processing
- Support for complex data structures

NOTE

In the discussion of this module, *n* is the channel number.

11.2 Functional description

The operation of the eDMA is described in the following subsections.

11.2.1 Modes of operation

eDMA operates in the following modes:

Table 11-3. Modes of operation

Mode	Description
Normal	In Normal mode, eDMA transfers data from a source to a destination. The source and destination can be a memory block or an I/O block capable of operation with eDMA. A *service request* initiates a transfer of a specific number of bytes (NBYTES) as specified in the TCD. • The *minor loop* is the sequence of read and write operations that transfers the NBYTES of data for a service request. • Each service request executes one iteration of the major loop, transferring NBYTES of data.
Debug	DMA operation is configurable in Debug mode via Control (CR) • If CR[EDBG] = 0, eDMA continues to operate normally when the chip is in debug mode. • If CR[EDBG] = 1, eDMA stops transferring data when the chip enters debug mode. If a channel is active when eDMA enters Debug mode, eDMA continues operation until the channel retires.
Wait	Before entering Wait mode, eDMA attempts to complete any transfer that is in progress. After the transfer completes, the chip enters Wait mode.

11.2.2 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments.

As shown in the following diagram, the first segment involves the channel activation:

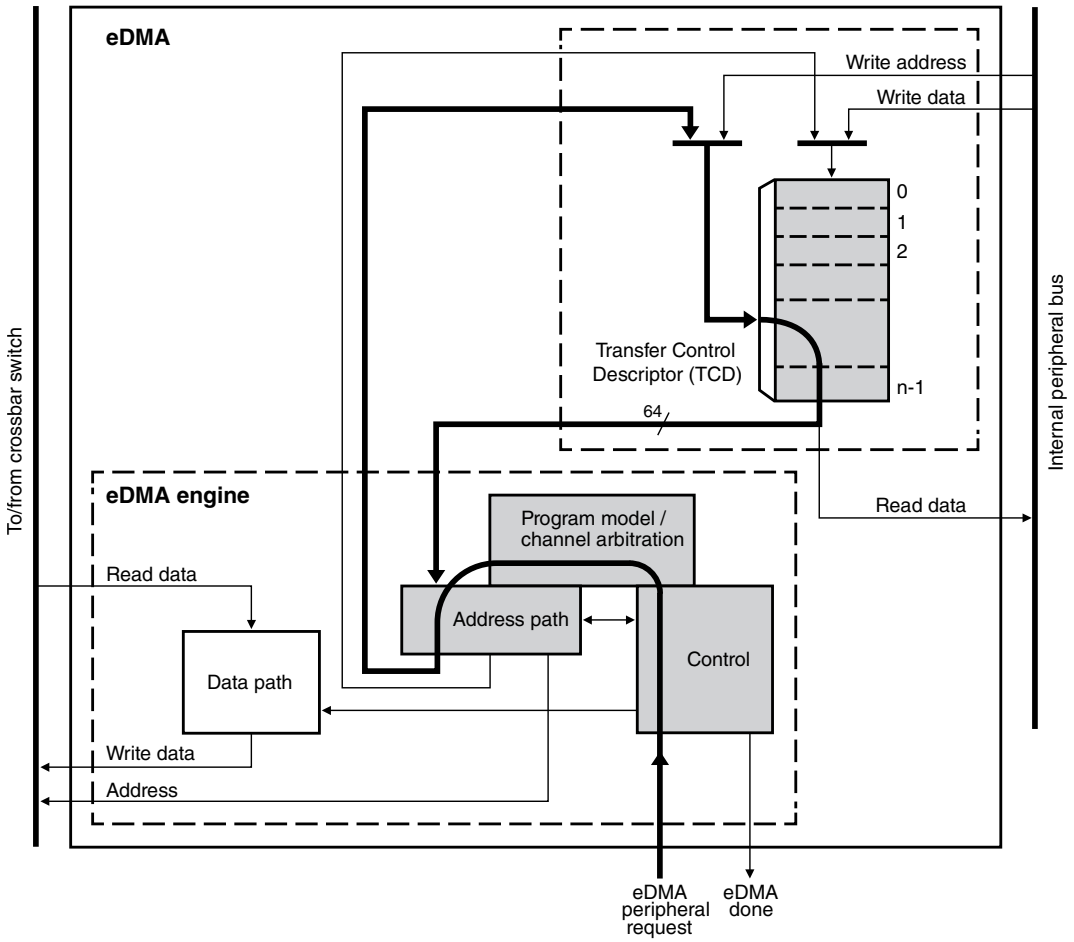


Figure 11-2. eDMA operation, part 1

This example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel activation via software and the TCD*n*_CSR[START] bit follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration. In the next cycle, the channel arbitration executes, using either the fixed-priority or round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the required address to access the local memory for TCD*n*. Next, the TCD memory is accessed and the required descriptor read from the local memory and loaded into the eDMA engine's internal register file. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the internal register file.

The following diagram illustrates the second part of the basic data flow:

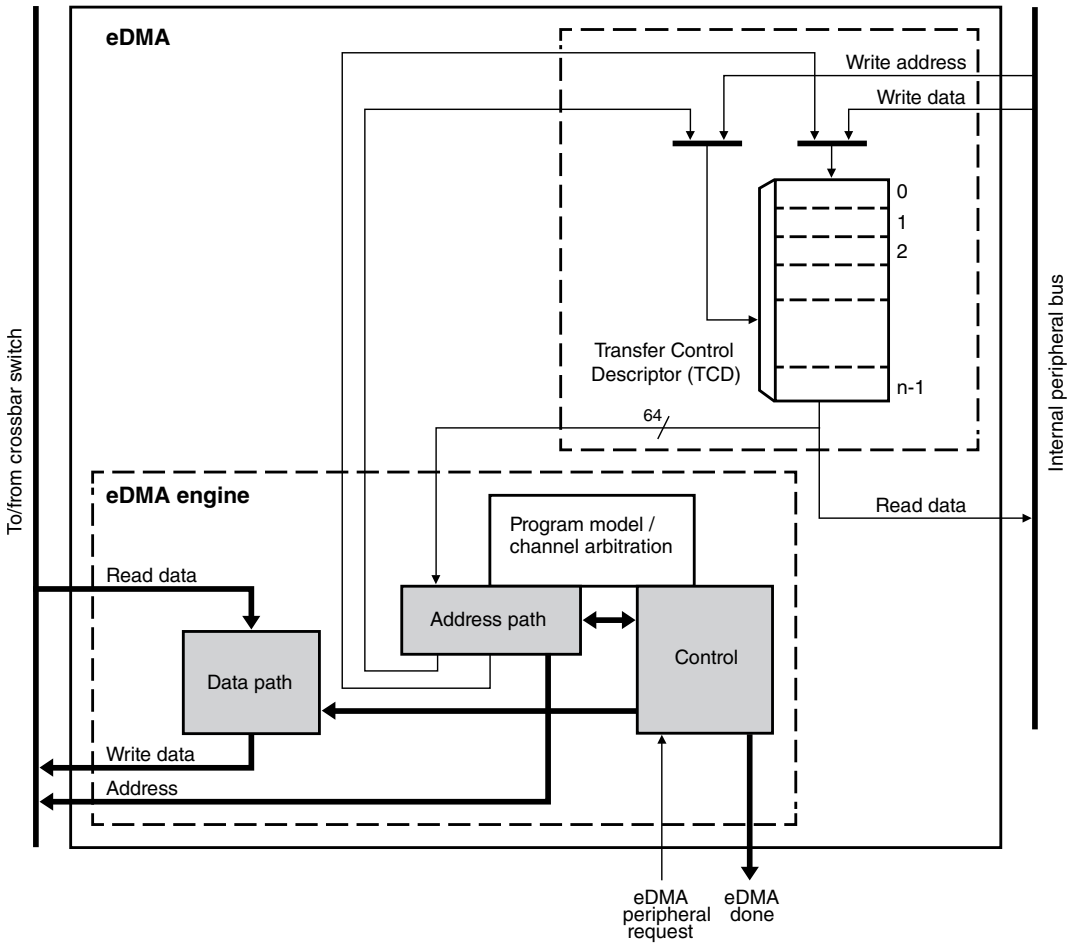


Figure 11-3. eDMA operation, part 2

The modules associated with the data transfer (address path, data path, and control) execute sequentially through the required source reads and destination writes to perform the data movement. The source reads are initiated and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the minor byte count has transferred.

After the minor byte count has moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs the required updates to certain fields in the appropriate TCD, for example, SADDR, DADDR, CITER. If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Assertion of an optional interrupt request also occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor (if scatter/gather is enabled). The updates to the TCD memory and the assertion of an interrupt request are shown in the following diagram.

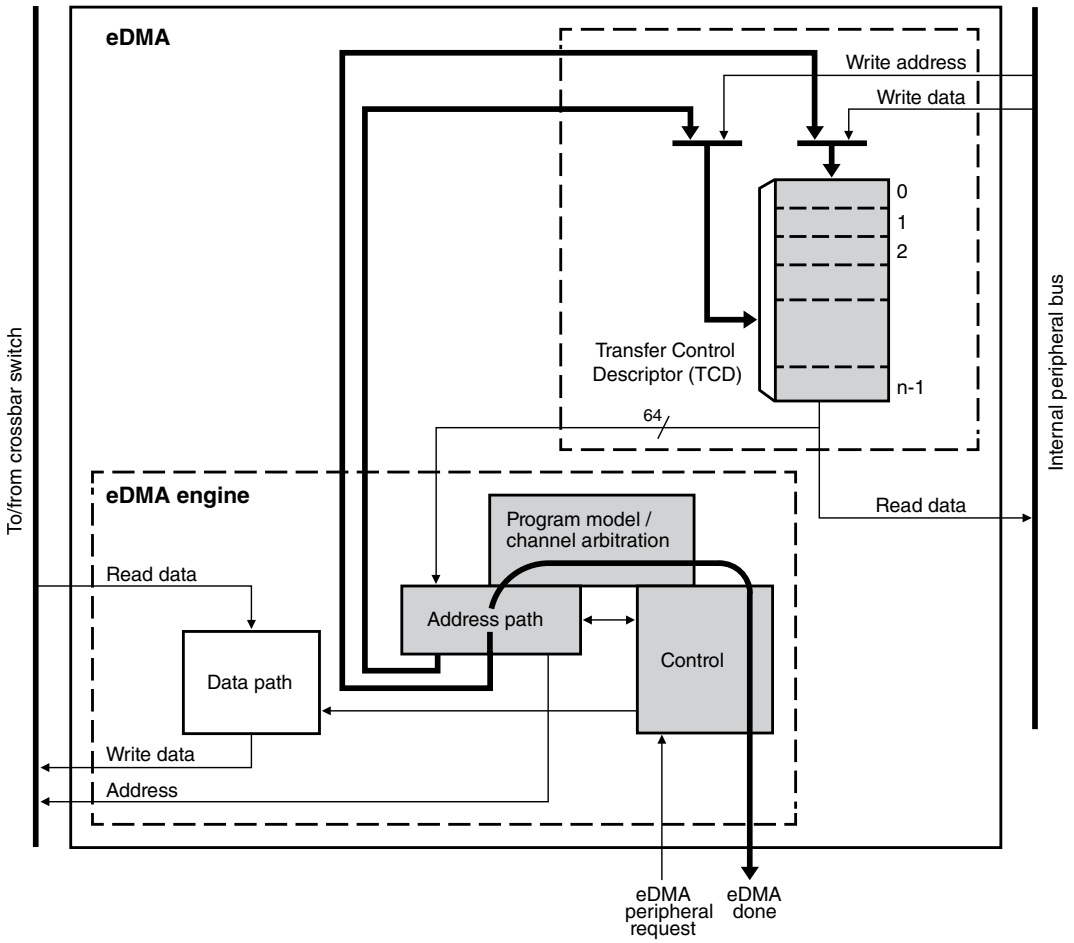


Figure 11-4. eDMA operation, part 3

11.2.3 Fault reporting and handling

Channel errors are reported in the Error Status register (DMAx_ES) and can be caused by:

- A configuration error, which is an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed-Arbitration mode, or
- An error termination to a bus master read or write cycle

A configuration error is reported when the starting source or destination address, source or destination offsets, minor loop byte count, or the transfer size represent an inconsistent state. Each of these possible causes is detailed below:

- The addresses and offsets must be aligned on 0-modulo transfer size boundaries.
- The minor loop byte count must be a multiple of the source and destination transfer sizes.

- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively.
- In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal. All channel priority levels must be unique when fixed arbitration mode is enabled.

NOTE

When two channels have the same priority, a channel priority error exists and is reported in the Error Status register. However, the channel number is not reported in the Error Status register. When all of the channel priorities within a group are not unique, the channel number selected by arbitration is undetermined.

To aid in Channel Priority Error (CPE) debug, set the Halt On Error bit in the DMA's Control register. If all channel priorities within a group are not unique, the DMA is halted after the CPE error is recorded. The DMA remains halted and does not process any channel service requests. After all of the channel priorities are set to unique numbers, the DMA may be enabled again by clearing the HALT bit.

- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn_CITER[ELINK] bit does not equal the TCDn_BITER[ELINK] bit.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, report as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion, when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system bus error occurs, the channel terminates after the next transfer. Due to pipeline effect, the next transfer is already in progress when the bus error is received by the eDMA. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the

data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

A transfer may be canceled by software with the CR[CX] bit. When a cancel transfer request is recognized, the DMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the Error Status register (DMAx_ES) is updated with the canceled channel number and ECX is set. The TCD of a canceled channel contains the source and destination addresses of the last transfer saved in the TCD. If the channel needs to be restarted, you must re-initialize the TCD because the aforementioned fields no longer represent the original parameters. When a transfer is canceled by the error cancel transfer mechanism, the channel number is loaded into DMA_ES[ERRCHN] and ECX and VLD are set. In addition, an error interrupt may be generated if enabled.

NOTE

The cancel transfer request enables you to stop a large data transfer when the full data transfer is no longer needed. The cancel transfer bit does not abort the channel. It simply stops the transferring of data and then retires the channel through its normal shutdown sequence. The application software must manage the context of the cancel. If an interrupt is desired (or not), then the interrupt should be enabled (or disabled) before the cancel request. The application software must clean up the transfer control descriptor because the full transfer did not occur.

The occurrence of any error causes the eDMA engine to stop normal processing of the active channel immediately (it goes to its error processing states and the transaction to the system bus still has pipeline effect), and the appropriate channel bit in the eDMA error register is asserted. At the same time, the details of the error condition are loaded into the Error Status register (DMAx_ES). The major loop complete indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request, are not affected when an error is detected. After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

11.2.4 Channel preemption

Channel preemption is enabled on a per-channel basis by setting DCHPRIn[ECP]. Channel preemption enables the executing channel's data transfers to temporarily be suspended in favor of starting a higher priority channel. After the preempting channel has completed its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended and the higher priority channel is serviced. Nested preemption, that is, attempting to preempt a preempting channel, is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is available only when fixed arbitration is selected.

A channel's ability to preempt another channel can be disabled by setting DCHPRIn[DPA]. When a channel's preemption ability is disabled, that channel cannot suspend a lower priority channel's data transfer, regardless of the lower priority channel's ECP setting. This enables a pool of low priority, large data-moving channels to be defined. These low priority channels can be configured to not preempt each other, thus preventing a low priority channel from consuming the preempt slot normally available to a true, high priority channel.

11.3 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

11.3.1 eDMA initialization

To initialize the eDMA:

1. Write to the CR if a configuration other than the default is desired.
2. Write the channel priority levels to the DCHPRI*n* registers if a configuration other than the default is desired.
3. Enable error interrupts in the EEI register if desired.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the ERQ register.
6. Request channel service via either:
   - Software: setting TCD*n*_CSR[START]
   - Hardware: slave device asserting its eDMA peripheral request signal

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels in the programming model. The eDMA engine reads the entire TCD, including the TCD control and status fields, as shown in Table 11-4, for the selected channel into its internal address path module.

As the TCD is read, the first transfer is initiated on the system bus, unless a configuration error is detected. Transfers from the source, as defined by TCD*n*_SADDR, to the destination, as defined by TCD*n*_DADDR, continue until the number of bytes specified by TCD*n*_NBYTES have been transferred.

When the transfer is complete, the eDMA engine's local TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post-processing executes, such as interrupts, major loop channel linking, and scatter/gather operations, if enabled.

Table 11-4. TCD control and status fields

TCD*n*_CSR field name	Description
START	Control bit to start channel explicitly when using a software-initiated DMA service (automatically cleared by hardware)
ACTIVE	Status bit indicating the channel is currently in execution
DONE	Status bit indicating major loop completion (cleared by software when using a software-initiated DMA service)
DREQ	Control bit to disable DMA request at end of major loop completion when using a hardware-initiated DMA service
BWC	Control bits for throttling bandwidth control of a channel
ESG	Control bit to enable scatter/gather feature
INTHALF	Control bit to enable interrupt when major loop is half complete
INTMAJOR	Control bit to enable interrupt when major loop completes

The following figure shows how each DMA request initiates one minor-loop transfer, or iteration, without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

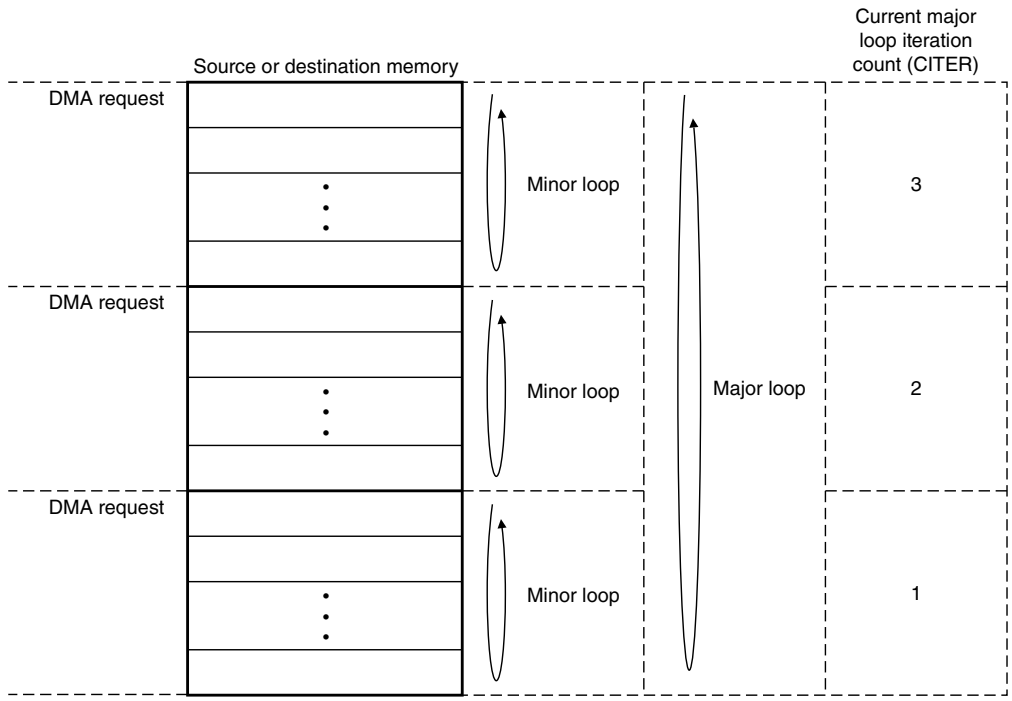


Figure 11-5. Example of multiple loop iterations

The following figure lists the memory array terms and how the TCD settings interrelate.

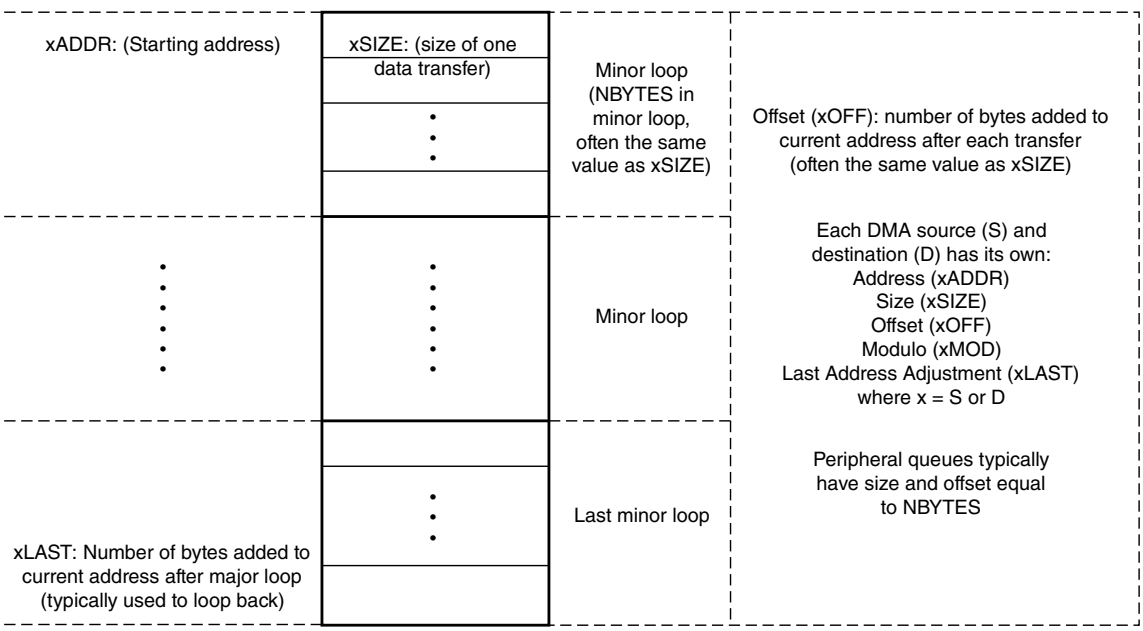


Figure 11-6. Memory array terms

11.3.2 Programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per-channel basis with the exception of channel priority error (ES[CPE]).

For all error types other than channel priority error, the channel number causing the error is recorded in the Error Status register (DMAx_ES). If the error source is not removed before the next activation of the problem channel, the error is detected and recorded again.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest channel priority with an active request is selected, but the lowest numbered channel with that priority is selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts, and error reporting are associated with the selected channel.

11.3.3 Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

11.3.3.1 Fixed channel arbitration

In this mode, the channel service request from the highest priority channel is selected to execute.

11.3.3.2 Round-robin channel arbitration

Channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels.

11.3.4 DMA transfer examples

This section presents examples of how to perform DMA transfers with the eDMA.

11.3.4.1 Single request

To perform a simple transfer of n bytes of data with one activation, set the major loop to one (TCD*n*_CITER = TCD*n*_BITER = 1). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, TCD*n*_CSR[DONE] is set and an interrupt generates if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has an 8-bit memory port located at 0x1000. The destination memory has a 32-bit port located at 0x2000. The address offsets are programmed in increments to match the transfer size: one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
TCDn_DLAST_SGA= -16
TCDn_CSR[INTMAJOR] = 1
TCDn_CSR[START] = 1 (Should be written last after all other fields have been initialized)
All other TCDn fields = 0
```

This generates the following event sequence:

1. User write to the TCD*n*_CSR[START] bit requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32 bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32 bits to location 0x2004 → second iteration of the minor loop.
   e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f. Write 32 bits to location 0x2008 → third iteration of the minor loop.
   g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.

h. Write 32 bits to location 0x200C → last iteration of the minor loop → major loop complete.

6. The eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 1 (TCD*n*_BITER).
7. The eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1.
8. The channel retires and the eDMA goes idle or services the next channel.

11.3.4.2 Multiple requests

The following example transfers 32 bytes via two hardware requests, but is otherwise the same as the previous example. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop, transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the ERQ register, the slave device initiates channel service requests.

```
TCDn_CITER = TCDn_BITER = 2
TCDn_SLAST = -32
TCDn_DLAST_SGA = -32
```

This would generate the following sequence of events:

1. First hardware, that is, the eDMA peripheral, requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file.
5. The source to destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32 bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32 bits to location 0x2004 → second iteration of the minor loop.
   e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f. Write 32 bits to location 0x2008 → third iteration of the minor loop.
   g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
   h. Write 32 bits to location 0x200C → last iteration of the minor loop.
6. eDMA engine writes: TCD*n*_SADDR = 0x1010, TCD*n*_DADDR = 0x2010, TCD*n*_CITER = 1.
7. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0.

8. The channel retires → one iteration of the major loop. The eDMA goes idle or services the next channel.
9. Second hardware, that is, eDMA peripheral, requests channel service.
10. The channel is selected by arbitration for servicing.
11. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
12. eDMA engine reads channel TCD data from local memory to internal register file.
13. The source to destination transfers are executed as follows:
    a. Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013.
    b. Write 32 bits to location 0x2010 → first iteration of the minor loop.
    c. Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017.
    d. Write 32 bits to location 0x2014 → second iteration of the minor loop.
    e. Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B.
    f. Write 32 bits to location 0x2018 → third iteration of the minor loop.
    g. Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F.
    h. Write 32 bits to location 0x201C → last iteration of the minor loop → major loop complete.
14. eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 2 (TCD*n*_BITER).
15. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[n] = 1.
16. The channel retires → major loop complete. The eDMA goes idle or services the next channel.

11.3.4.3 Using the modulo feature

The modulo feature of the eDMA provides the ability to implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address+offset calculation. All upper address bits remain the same as in the original value. A setting of zero for this field disables the modulo feature.

The following table shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567*x*) retain their original value. In this example the source address is set to 0x12345670, the offset is set to 4 bytes, and the MOD field is set to 4, allowing for a 2^4 byte (16-byte) size queue.

Table 11-5. Modulo example

Transfer number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

11.3.5 Monitoring transfer descriptor status

This section discusses how to monitor eDMA status.

11.3.5.1 Testing for minor loop completion

There are two methods to test for minor loop completion when using software-initiated service requests. The first is to read the TCD*n*_CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test TCD*n*_CSR[START] and TCD*n*_CSR[ACTIVE]. The minor-loop-complete condition is indicated by both bits reading zero after TCD*n*_CSR[START] was set. Polling the TCD*n*_CSR[ACTIVE] bit may be inconclusive, because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

Stage	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	1	0	0	Channel service request via software
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

The best method to test for minor-loop completion when using service requests initiated by hardware, that is, peripherals, is to read the TCD*n*_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status bits execute the following sequence for a hardware-activated channel:

Stage	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	0	0	0	Channel service request via hardware (peripheral request asserted)
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

For both activation types, the major-loop-complete status is explicitly indicated via the TCD*n*_CSR[DONE] bit.

The TCD*n*_CSR[START] bit is cleared automatically when the channel begins execution regardless of how the channel activates.

11.3.5.2 Reading the transfer descriptors of active channels

The eDMA reads back the true TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_NBYTES values if read when a channel executes. The true values of SADDR, DADDR, and NBYTES are the values the eDMA engine currently uses in its internal register file and not the values in the TCD local memory for that channel. The addresses, SADDR and DADDR, and NBYTES, which decrement to zero as the transfer progresses, can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

11.3.5.3 Checking channel preemption status

Preemption is available only when fixed arbitration is selected as the channel arbitration mode. A preemptive situation is one in which a preempt-enabled channel runs and a higher priority request becomes active. When the eDMA engine is not operating in fixed channel arbitration mode, determination of the actively running relative priority outstanding requests become undefined. Channel priorities are treated as equal, that is, constantly rotating, when Round-Robin Arbitration mode is selected.

The TCD*n*_CSR[ACTIVE] bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one major loop iteration. If two TCD*n*_CSR[ACTIVE] bits are set simultaneously in the global TCD map, a higher priority channel is actively preempting a lower priority channel.

11.3.6 Channel linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD*n*_CSR[START] bit of another channel (or itself), thus initiating a service request for that channel. When properly enabled, the EDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD*n*_CITER[ELINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, the initial fields of:

```
TCDn_CITER[ELINK] = 1
TCDn_CITER[LINKCH] = 0xC
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJOR_ELINK] = 1
TCDn_CSR[MAJOR_LINKCH] = 0x3
```

executes as:

1. Minor loop done → set TCD2_CSR[START] bit.
2. Minor loop done → set TCD2_CSR[START] bit.
3. Minor loop done → set TCD2_CSR[START] bit.
4. Minor loop done, major loop done → set TCD3_CSR[START] bit.

When minor loop linking is enabled (TCD*n*_CITER[ELINK] = 1), the TCD*n*_CITER[CITER] field uses a nine-bit vector to form the current iteration count. When minor loop linking is disabled (TCD*n*_CITER[ELINK] = 0), the TCD*n*_CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD*n*_CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

Note

The TCD*n*_CITER[ELINK] bit and the TCD*n*_BITER[ELINK] bit must be equal or a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop, half-way done interrupt point.

The following table summarizes how a DMA channel can link to another DMA channel, that is, use another channel's TCD, at the end of a loop.

Table 11-6. Channel linking parameters

Desired link behavior	TCD control field name	Description
Link at end of minor loop	CITER[ELINK]	Enable channel-to-channel linking on minor loop completion (current iteration)
	CITER[LINKCH]	Link channel number when linking at end of minor loop (current iteration)
Link at end of major loop	CSR[MAJOR_ELINK]	Enable channel-to-channel linking on major loop completion
	CSR[MAJOR_LINKCH]	Link channel number when linking at end of major loop

11.3.7 Dynamic programming

This section provides recommended methods to change the programming model during channel execution.

11.3.7.1 Dynamically changing the channel priority

The following two options are recommended for dynamically changing channel priority levels:

1. Switch to Round-Robin Channel Arbitration mode, change the channel priorities, then switch back to Fixed Arbitration mode
2. Disable all the channels, change the channel priorities, then enable the appropriate channels.

11.3.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting TCDn_CSR[MAJORELINK] during channel execution (see the diagram in TCD structure). This field is read from the TCD local memory at the end of channel execution, thus enabling you to enable the feature during channel execution.

Because you can change the configuration during execution, a coherency model is needed. Consider the scenario where you attempt to execute a dynamic channel link by enabling TCDn_CSR[MAJORELINK] at the same time the eDMA engine is retiring the channel. TCDn_CSR[MAJORELINK] would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The following coherency model is recommended when executing a dynamic channel link request.

1. Write one to TCDn_CSR[MAJORELINK].
2. Read back TCDn_CSR[MAJORELINK].
3. Test the TCDn_CSR[MAJORELINK] request status:
   - If TCDn_CSR[MAJORELINK] = 1, the dynamic link attempt was successful.
   - If TCDn_CSR[MAJORELINK] = 0, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces TCDn_CSR[MAJORELINK] to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

NOTE

You must clear TCDn_CSR[DONE] before writing TCDn_CSR[MAJORELINK]. The eDMA engine automatically clears TCDn_CSR[DONE] after a channel begins execution.

11.3.7.3 Dynamic scatter/gather

Scatter/gather is the process of automatically loading a new TCD into a channel. It enables a DMA channel to use multiple TCDs; this enables a DMA channel to scatter the DMA data to multiple destinations or gather it from multiple sources. When scatter/gather is enabled and the channel has finished its major loop, a new TCD is fetched from system memory and loaded into that channel's descriptor location in eDMA programmer's model, thus replacing the current descriptor.

Because you are able to change the configuration during execution, a coherency model is needed. Consider the scenario where you attempt to execute a dynamic scatter/gather operation by enabling the TCDn_CSR[ESG] bit at the same time the eDMA engine is retiring the channel. The ESG bit would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the MAJORLINKCH field and the ESG bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD MAJOR[ELINK] and ESG bits to zero on any writes to a channel's TCD word 7 if that channel's TCD[DONE] bit is set, indicating the major loop is complete.

NOTE

The user must clear the TCDn_CSR[DONE] bit before writing the MAJORELINK or ESG bits. The TCDn_CSR[DONE] bit is cleared automatically by the eDMA engine after a channel begins execution.

11.3.7.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request.

When TCDn_CSR[MAJORELINK] is zero, TCDn_CSR[MAJORLINKCH] is not used by the eDMA. In this case, MAJORLINKCH may be used for other purposes. This method uses the MAJORLINKCH field as a TCD identification (ID).

1. When the descriptors are built, write a unique TCD ID in TCDn_CSR[MAJORLINKCH] for each TCD associated with a channel using dynamic scatter/gather.
2. Write one to TCDn_CSR[DREQ].

   Should a dynamic scatter/gather attempt fail, setting the DREQ bit prevents a future hardware activation of the channel. This stops the channel from executing with a destination address (DADDR) that was calculated using a scatter/gather address (written in the next step) instead of a DLAST_SGA final offset value.

3. Write the TCDn_DLASTSGA register with the scatter/gather address.
4. Write one to TCDn_CSR[ESG].
5. Read back the 16-bit TCD control/status field.
6. Test the ESG request status and MAJORLINKCH value in the TCDn_CSR register:

   If ESG = 1, the dynamic link attempt was successful.

   If ESG = 0 and MAJORLINKCH (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring).

   If ESG = 0 and MAJORLINKCH (ID) changed, the dynamic link attempt was successful (the new TCD's ESG value cleared the ESG bit).

11.3.7.3.2 Method 2 (channel using major loop channel linking)

For a channel using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request. This method uses the TCD[DLAST_SGA] field as a TCD identification (ID).

1. Write one to TCDn_CSR[DREQ].

Should a dynamic scatter/gather attempt fail, setting TCDn_CSR[DREQ] prevents a future hardware activation of the channel. This stops the channel from executing with a destination address (DADDR) that was calculated using a scatter/gather address (written in the next step) instead of a DLAST_SGA final offset value.

2. Write the TCDn_DLAST_SGA register with the scatter/gather address.
3. Write one to TCDn_CSR[ESG].
4. Read back TCDn_CSR[ESG].
5. Test the ESG request status:

   If ESG = 1, the dynamic link attempt was successful.

   If ESG = 0, read the 32-bit TCDn_DLAST_SGA field.

   If ESG = 0 and TCDn_DLAST_SGA did not change, the attempted dynamic link did not succeed (the channel was already retiring).

   If ESG = 0 and TCDn_DLAST_SGA changed, the dynamic link attempt was successful (the new TCD's ESG value cleared the ESG bit).

11.3.8 Suspend/resume a DMA channel with active hardware service requests

The DMA enables you to move data from memory or peripheral registers to another location in memory or peripheral registers without CPU interaction. After the DMA and peripherals have been configured and are active, it is rare to suspend a peripheral's service request dynamically. In this scenario, there are certain restrictions to disabling a DMA hardware service request. For coherency, a specific procedure must be followed. This section provides guidance on how to coherently suspend and resume a Direct Memory Access (DMA) channel when the DMA is triggered by a slave module such as the Serial Peripheral Interface (SPI), ADC, or other module.

11.3.8.1 Suspend an active DMA channel

To suspend an active DMA channel:

1. Stop the DMA service request at the peripheral first. Confirm it has been disabled by reading back the appropriate register in the peripheral.
2. Check the DMA's Hardware Request Status register (DMA_HRSn) to ensure there is no service request to the DMA channel being suspended. Then disable the hardware service request by clearing the ERQ bit on appropriate DMA channel.

11.3.8.2 Resume a DMA channel

To resume a DMA channel:

1. Enable the DMA service request on the appropriate channel by setting the relevant ERQ bit.
2. Enable the DMA service request at the peripheral.

For example, assume the SPI is set as a master for transmitting data via a DMA service request when the SPI_TXFIFO has an empty slot. The DMA transfers the next command and data to the TXFIFO upon the request. You must suspend the DMA/SPI transfer loop and perform the following steps:

1. Disable the DMA service request at the source by writing zero to SPI_RSER[TFFF_RE]. Confirm that SPI_RSER[TFFF_RE] is zero.
2. Ensure there is no DMA service request from the SPI by verifying that DMA_HRS[HRS*n*] is zero for the appropriate channel. If no service request is present, disable the DMA channel by clearing the channel's ERQ bit. If a service request is present, wait until the request has been processed and the HRS bit reads zero.

11.4 Memory map/register definition

The eDMA programming model consists of registers that provide:

- Control and status functions
- Channel configuration functions
- TCD definition functions

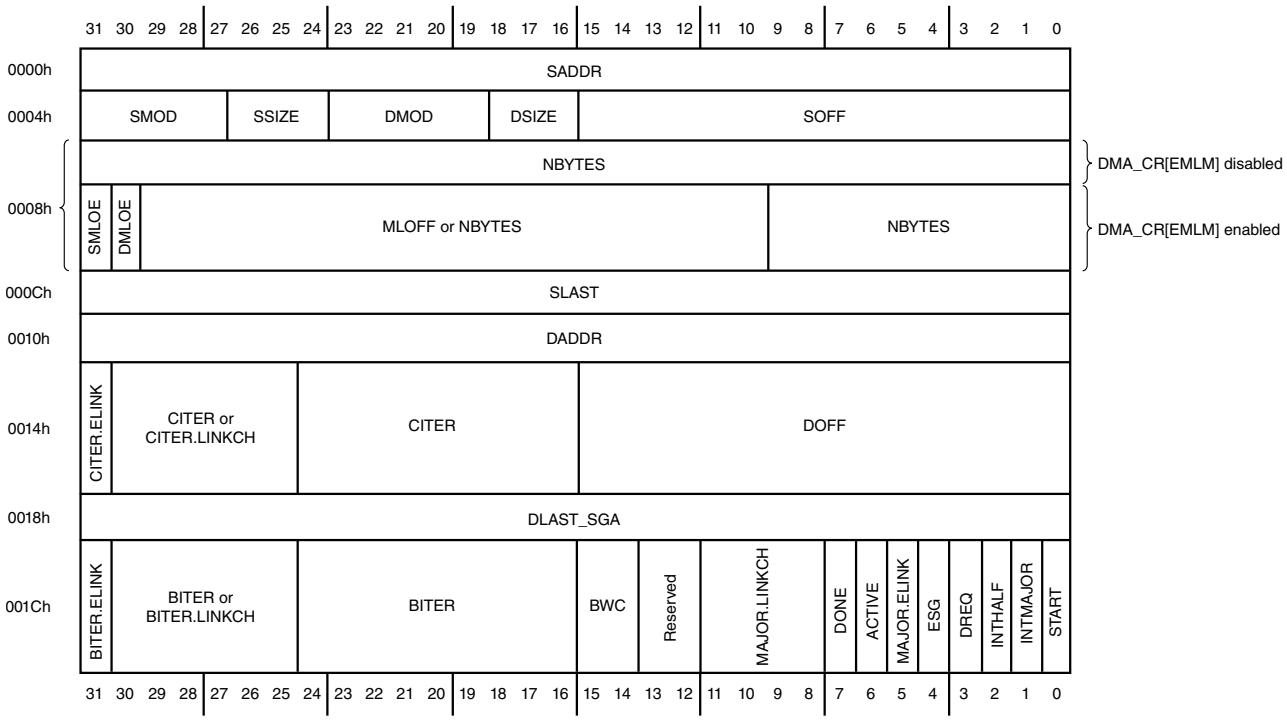
11.4.1 TCD memory

Each channel requires a 32-byte TCD to define the desired data movement operation. The channel descriptors are in local memory in sequential order: channel 0, channel 1, ... channel 3. Each TCD*n* definition comprises 11 registers of 16 or 32 bits.

11.4.2 TCD initialization

Before activating a channel, you must initialize its TCD with the appropriate transfer profile.

11.4.3 TCD structure



11.4.4 Reserved memory and fields

- Reading reserved fields in a register returns the value of zero.
- The eDMA ignores writes to reserved bits in a register.
- Reading or writing a reserved memory location generates a bus error.

11.4.5 DMA register descriptions

NOTE

The base address and offsets for these registers are presented in terms of bytes.

11.4.5.1 DMA memory map

DMA0 base address: 1_9000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control (CR)	32	RW	Table 11-6
4h	Error Status (ES)	32	RO	0000_0000h
Ch	Enable Request (ERQ)	32	RW	0000_0000h
14h	Enable Error Interrupt (EEI)	32	RW	0000_0000h
18h	Clear Enable Error Interrupt (CEEI)	8	WORZ	00h
19h	Set Enable Error Interrupt (SEEI)	8	WORZ	00h
1Ah	Clear Enable Request (CERQ)	8	WORZ	00h
1Bh	Set Enable Request (SERQ)	8	WORZ	00h
1Ch	Clear DONE Status Bit (CDNE)	8	WORZ	00h
1Dh	Set START Bit (SSRT)	8	WORZ	00h
1Eh	Clear Error (CERR)	8	WORZ	00h
1Fh	Clear Interrupt Request (CINT)	8	WORZ	00h
24h	Interrupt Request (INT)	32	W1C	0000_0000h
2Ch	Error (ERR)	32	W1C	0000_0000h
34h	Hardware Request Status (HRS)	32	RO	0000_0000h
44h	Enable Asynchronous Request in Stop (EARS)	32	RW	0000_0000h
100h	Channel Priority (DCHPRI3)	8	RW	03h
101h	Channel Priority (DCHPRI2)	8	RW	02h
102h	Channel Priority (DCHPRI1)	8	RW	01h
103h	Channel Priority (DCHPRI0)	8	RW	00h
1000h - 1060h	TCD Source Address (TCD0_SADDR - TCD3_SADDR)	32	RW	Table 11-6
1004h - 1064h	TCD Signed Source Address Offset (TCD0_SOFF - TCD3_SOFF)	16	RW	Table 11-6
1006h - 1066h	TCD Transfer Attributes (TCD0_ATTR - TCD3_ATTR)	16	RW	Table 11-6
1008h - 1068h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD0_NBYTES_MLNO - TCD3_NBYTES_MLNO)	32	RW	Table 11-6
1008h - 1068h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD0_NBYTES_MLOFFNO - TCD3_NBYTES_MLOFFNO)	32	RW	Table 11-6
1008h - 1068h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD0_NBYTES_MLOFFYES - TCD3_NBYTES_MLOFFYES)	32	RW	Table 11-6
100Ch - 106Ch	TCD Last Source Address Adjustment (TCD0_SLAST - TCD3_SLAST)	32	RW	Table 11-6
1010h - 1070h	TCD Destination Address (TCD0_DADDR - TCD3_DADDR)	32	RW	Table 11-6
1014h - 1074h	TCD Signed Destination Address Offset (TCD0_DOFF - TCD3_DOFF)	16	RW	Table 11-6
1016h - 1076h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD3_CITER_ELINKNO)	16	RW	Table 11-6
1016h - 1076h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD3_CITER_ELINKYES)	16	RW	Table 11-6
1018h - 1078h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD0_DLASTSGA - TCD3_DLASTSGA)	32	RW	Table 11-6

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
101Ch - 107Ch	TCD Control and Status (TCD0_CSR - TCD3_CSR)	16	RW	Table 11-6
101Eh - 107Eh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_BITER_ELINKNO - TCD3_BITER_ELINKNO)	16	RW	Table 11-6
101Eh - 107Eh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_BITER_ELINKYES - TCD3_BITER_ELINKYES)	16	RW	Table 11-6

11.4.5.2 Control (CR)

11.4.5.2.1 Offset

Register	Offset
CR	0h

11.4.5.2.2 Function

This register defines the basic operating configuration of the eDMA module.

You can configure arbitration to use either a fixed-priority or a round-robin scheme. For fixed-priority arbitration, eDMA selects and executes the highest-priority channel that requests service. The channel priority registers assign the priorities (see the Channel Priority (DCHPRI0 - DCHPRI3) registers). For round-robin arbitration, the eDMA engine ignores channel priorities and cycles through channels from high to low channel number without regard to priority.

NOTE

For correct operation, you must write to this register only when the eDMA channels are inactive—that is, when TCD*n*_CSR[ACTIVE] = 0.

Minor loop offsets are address-offset values to be added to the final source address (TCDn_SADDR) or destination address (TCDn_DADDR) when the minor loop completes. When you enable minor loop offsets, eDMA adds the minor loop offset (MLOFF) value to the final source address (TCDn_SADDR), to the final destination address (TCDn_DADDR), or to both, before it writes the addresses back into the TCD. If the major loop is complete, eDMA ignores the minor loop offset, and uses the major loop address offsets (TCDn_SLAST and TCDn_DLAST_SGA) to compute the next TCDn_SADDR and TCDn_DADDR values.

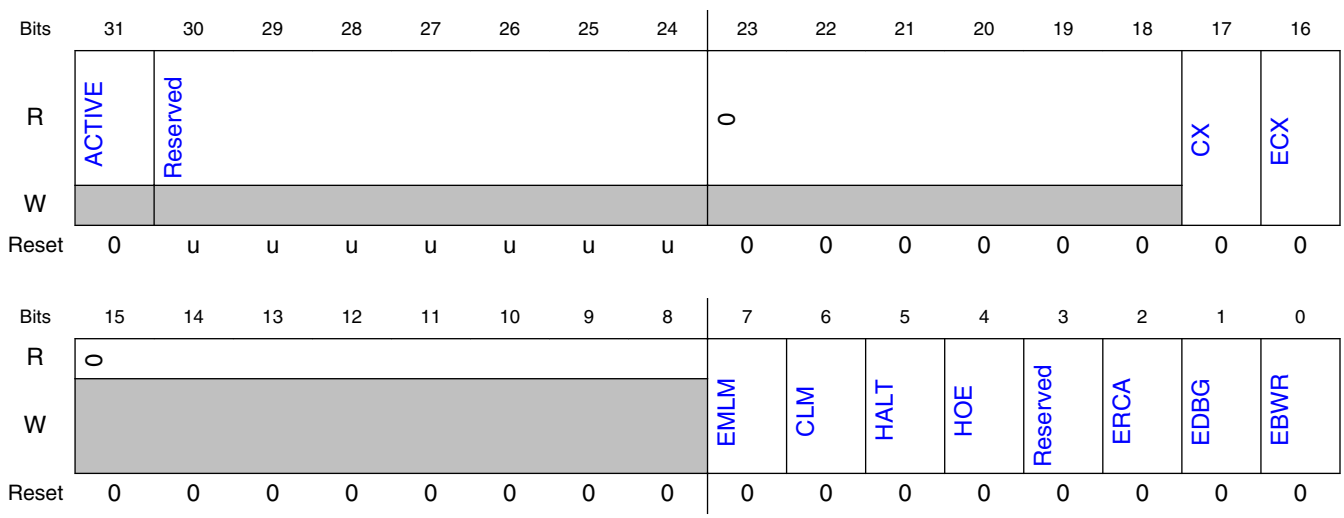
Enabling minor loop mapping (EMLM = 1) redefines TCDn word2. eDMA uses a portion of TCDn word2 for multiple fields:

- A source enable field (SMLOE) to specify the minor loop offset is to be applied to the source address (TCDn_SADDR) when the minor loop completes
- A destination enable field (DMLOE) to specify the minor loop offset to be applied to the destination address (TCDn_DADDR) when the minor loop completes
- The sign extended minor loop offset value (MLOFF).

eDMA uses the same offset value (MLOFF) for both source and destination minor loop offsets. When you enable either minor loop offset (SMLOE = 1 or DMLOE = 1), the NBYTES field reduces in size to 10 bits. When you disable both minor loop offsets (SMLOE = 0 and and DMLOE = 0), the NBYTES field is a 30-bit vector.

When you disable minor loop mapping (EMLM = 0), the NBYTES field contains all 32 bits of TCDn word2.

11.4.5.2.3 Diagram



11.4.5.2.4 Fields

Field	Function
31 ACTIVE	eDMA Active Status 0b - eDMA is idle 1b - eDMA is executing a channel
30-24 —	Reserved
23-18 —	Reserved

Table continues on the next page...

Field	Function
17 CX	Cancel Transfer When you write 1 to this field, the following actions take place: • Stop the executing channel • Force the minor loop to finish. The cancellation takes effect after the last write of the current read/write sequence. This field is automatically written with 0 after the cancellation completes. The cancellation retires the channel normally as if the minor loop completed. 0b - Normal operation 1b - Cancel the remaining data transfer
16 ECX	Error Cancel Transfer When you write a 1 to this field, the following actions take place: • Stop the executing channel • Force the minor loop to finish. The cancellation takes effect after the last write of the current read/write sequence. This field is automatically reset to 0 after the cancellation completes. In addition to cancelling the transfer, eDMA: • Treats the cancel as an error condition • Updates the Error Status register (DMAx_ES) • Optionally generates an error interrupt. 0b - Normal operation 1b - Cancel the remaining data transfer
15-8 —	Reserved
7 EMLM	Enable Minor Loop Mapping When the value of this field is 0, TCDn.word2 is a 32-bit NBYTES field. When the value of this field is 1, TCDn.word2 includes: • Individual enable fields • An offset field • The NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The NBYTES field reduces in size when either offset is enabled. 0b - Disabled 1b - Enabled
6 CLM	Continuous Link Mode When the value of this field is 0, a minor loop channel link made to itself goes through channel arbitration before being activated again. When the value of this field is 1, a minor loop channel link made to itself does not go through channel arbitration before being activated again. When the minor loop completes, the channel activates again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop. **NOTE:** Do not use continuous link mode with a channel linking to itself if there is only one minor loop iteration per service request, for example, if the channel's NBYTES value is the same as either the source or destination size. The same data transfer profile can be achieved by simply increasing the NBYTES value, which provides more efficient, faster processing. 0b - Continuous link mode is off 1b - Continuous link mode is on
5 HALT	Halt eDMA Operations When this field is 1 the following actions take place: • eDMA stalls the start of any new channels • Executing channels are allowed to complete.

Table continues on the next page...

Field	Function
	When you write 0 to this field, channel execution resumes. 0b - Normal operation 1b - eDMA operations halted
4 HOE	Halt On Error When this field is 1, any error causes the eDMA engine to write 1 to the HALT field. Subsequently, the eDMA engine ignores all service requests until you write 0 to the HALT field. 0b - Normal operation 1b - Error causes HALT field to be automatically set to 1
3 —	Reserved
2 ERCA	Enable Round Robin Channel Arbitration When you write 1 to this field, eDMA uses round robin arbitration for channel selection. Otherwise, eDMA uses fixed priority arbitration for channel selection. 0b - Fixed priority arbitration 1b - Round robin arbitration
1 EDBG	Enable Debug When this field is 0 and the chip enters Debug mode, eDMA continues operation. When this field is 1, entry of the chip into Debug mode causes the eDMA to stall the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the chip exits Debug mode or you write 0 to this field. 0b - When the chip is in Debug mode, the eDMA continues to operate. 1b - When the chip is in debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete.
0 EBWR	Enable Buffered Writes 0b - Buffered writes are disabled 1b - Buffered writes are enabled

11.4.5.3 Error Status (ES)

11.4.5.3.1 Offset

Register	Offset
ES	4h

11.4.5.3.2 Function

The ES register provides information concerning the most-recently recorded channel error. Channel errors can be caused by:

- A configuration error, that is:
  - An illegal setting in the transfer-control descriptor
  - An illegal priority register setting in fixed arbitration

- An error termination to a bus master read or write cycle
- A cancel transfer with error field that is 1 when a transfer is canceled via the corresponding cancel transfer control field

See Fault reporting and handling for more details.

11.4.5.3.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	CPE	0				ERRCHN		SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

11.4.5.3.4 Fields

Field	Function
31 VLD	Logical OR of all ERR status fields 0b - No ERR fields are 1 1b - At least one ERR field has a value of 1, indicating a valid error exists that has not been cleared
30-17 —	Reserved
16 ECX	Transfer Canceled 0b - No canceled transfers 1b - The most-recently recorded entry was a canceled transfer initiated by the error cancel transfer field
15 —	Reserved
14 CPE	Channel Priority Error 0b - No channel priority error. 1b - The most-recently recorded error was a configuration error in the channel priorities. Channel priorities are not unique.
13-10 —	Reserved
9-8 ERRCHN	Error Channel Number or Canceled Channel Number The channel number of the most-recently recorded error, excluding CPE errors or most-recently recorded error canceled transfer.
7 SAE	Source Address Error 0b - No source address configuration error.

Table continues on the next page...

Field	Function
	1b - The most-recently recorded error was a configuration error detected in the TCDn_SADDR field. TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE].
6 SOE	Source Offset Error 0b - No source offset configuration error. 1b - The most-recently recorded error was a configuration error detected in the TCDn_SOFF field. TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE].
5 DAE	Destination Address Error 0b - No destination address configuration error. 1b - The most-recently recorded error was a configuration error detected in the TCDn_DADDR field. TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE].
4 DOE	Destination Offset Error 0b - No destination offset configuration error. 1b - The most-recently recorded error was a configuration error detected in the TCDn_DOFF field. TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE].
3 NCE	NBYTES/CITER Configuration Error 0b - No NBYTES/CITER configuration error. 1b - The most-recently recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields. TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE], or TCDn_CITER[CITER] = 0, or TCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK].
2 SGE	Scatter/Gather Configuration Error When 1, this field indicates the most-recently recorded error was a configuration error detected in the TCDn_DLASTSGA field. eDMA checks This field at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG] is enabled. TCDn_DLASTSGA is not on a 32-byte boundary. 0b - No scatter/gather configuration error. 1b - The most-recently recorded error was a configuration error detected in the TCDn_DLASTSGA field.
1 SBE	Source Bus Error 0b - No source bus error. 1b - The most-recently recorded error was a bus error on a source read.
0 DBE	Destination Bus Error 0b - No destination bus error. 1b - The most-recently recorded error was a bus error on a destination write.

11.4.5.4 Enable Request (ERQ)

11.4.5.4.1 Offset

Register	Offset
ERQ	Ch

11.4.5.4.2 Function

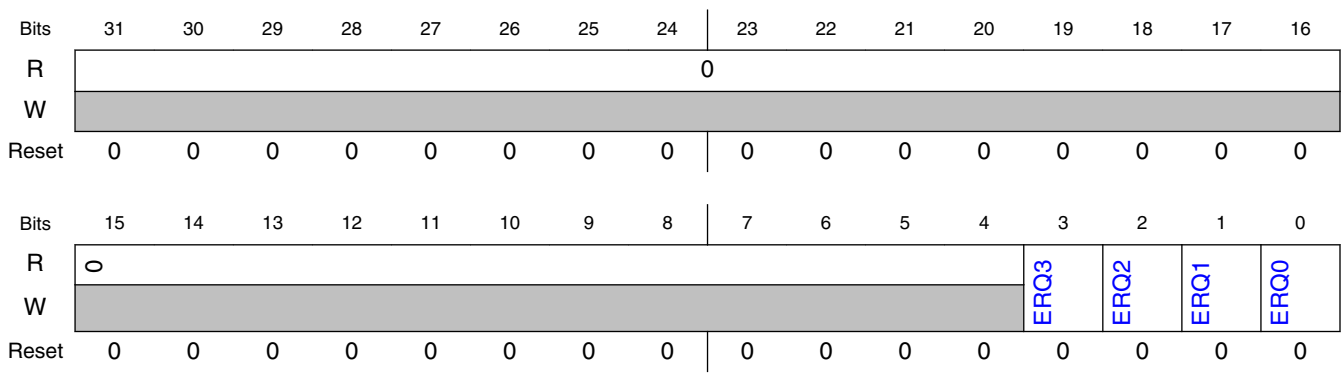
The ERQ register provides a bit map for the 4 channels to enable the request signal for each channel. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ registers. These registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to this register.

DMA request input signals and this enable request field must be set to 1 before a channel's hardware service request is accepted. The state of the DMA enable request field does not affect a channel service request made explicitly through software or a linked channel request.

NOTE

Disable a channel's hardware service request at the source before writing 0 to the channel's ERQ field.

11.4.5.4.3 Diagram



11.4.5.4.4 Fields

Field	Function
31-4 —	Reserved
3 ERQ3	Enable DMA Request 3 0b - The DMA request signal for channel 3 is disabled 1b - The DMA request signal for channel 3 is enabled
2 ERQ2	Enable DMA Request 2 0b - The DMA request signal for channel 2 is disabled 1b - The DMA request signal for channel 2 is enabled
1 ERQ1	Enable DMA Request 1 0b - The DMA request signal for channel 1 is disabled 1b - The DMA request signal for channel 1 is enabled

Table continues on the next page...

Field	Function
0 ERQ0	Enable DMA Request 0 0b - The DMA request signal for channel 0 is disabled 1b - The DMA request signal for channel 0 is enabled

11.4.5.5 Enable Error Interrupt (EEI)

11.4.5.5.1 Offset

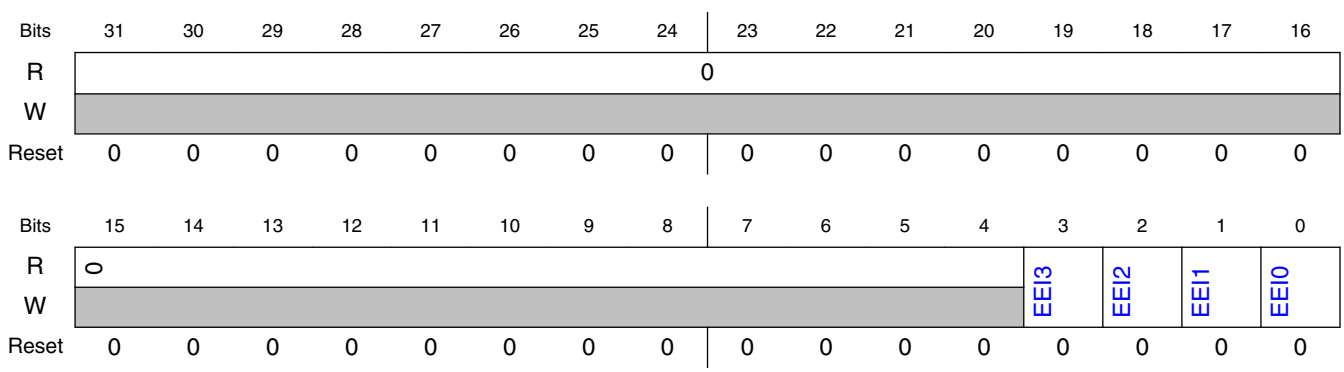
Register	Offset
EEI	14h

11.4.5.5.2 Function

The EEI register provides a bit map for the 4 channels to enable the error interrupt signal for each channel. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI registers. These registers are provided so that the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI register.

The DMA error indicator and the error interrupt enable field must be set to 1 before an error interrupt request for a given channel is sent to the interrupt controller.

11.4.5.5.3 Diagram



11.4.5.5.4 Fields

Field	Function
31-4 —	Reserved
3 EEI3	Enable Error Interrupt 3 0b - An error on channel 3 does not generate an error interrupt 1b - An error on channel 3 generates an error interrupt request
2 EEI2	Enable Error Interrupt 2 0b - An error on channel 2 does not generate an error interrupt 1b - An error on channel 2 generates an error interrupt request
1 EEI1	Enable Error Interrupt 1 0b - An error on channel 1 does not generate an error interrupt 1b - An error on channel 1 generates an error interrupt request
0 EEI0	Enable Error Interrupt 0 0b - An error on channel 0 does not generate an error interrupt 1b - An error on channel 0 generates an error interrupt request

11.4.5.6 Clear Enable Error Interrupt (CEEI)

11.4.5.6.1 Offset

Register	Offset
CEEI	18h

11.4.5.6.2 Function

The CEEI provides a simple memory-mapped mechanism to write 0 to a given field in the EEI register to disable the error interrupt for a given channel. The data value on a register write causes the corresponding field in the EEI register to be written to 0. Writing 1 to the CAEE field provides a global clear to 0 function, forcing the EEI contents to be written to 0, disabling all DMA request inputs.

If the NOP field is written with 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field set to 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

11.4.5.6.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0					0	
W	NOP	CAEE	0				CEEI	
Reset	0	0	0	0	0	0	0	0

11.4.5.6.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation, ignore the other fields in this register
6 CAEE	Clear All Enable Error Interrupts 0b - Write 0 only to the EEI field specified in the CEEI field 1b - Write 0 to all fields in EEI
5-2 —	Reserved
1-0 CEEI	Clear Enable Error Interrupt Writes 0 to the corresponding field in EEI

11.4.5.7 Set Enable Error Interrupt (SEEI)

11.4.5.7.1 Offset

Register	Offset
SEEI	19h

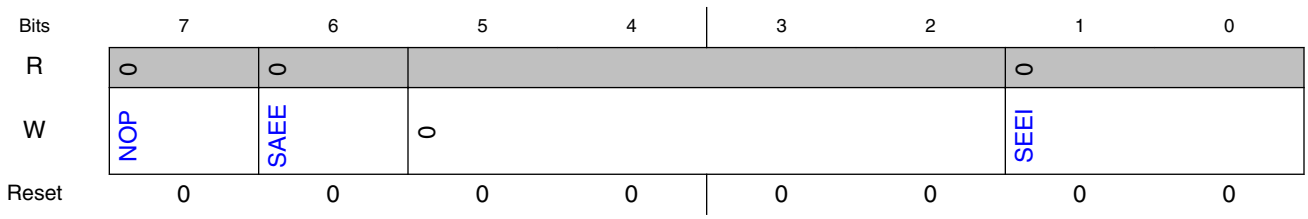
11.4.5.7.2 Function

The SEEI register provides a simple memory-mapped mechanism to write 1 to a given field in the EEI register to enable the error interrupt for a given channel. The data value on a register write causes the corresponding field in the EEI to be written to 1. Writing 1 to the SAEE field provides a global set to 1 function, forcing the entire EEI register contents to be written with 1.

If the NOP field is 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field set to 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

11.4.5.7.3 Diagram



11.4.5.7.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation, ignore the other fields in this register
6 SAEE	Set All Enable Error Interrupts 0b - Write 1 only to the EEI field specified in the SEEI field 1b - Writes 1 to all fields in EEI
5-2 —	Reserved
1-0 SEEI	Set Enable Error Interrupt Writes 1 to the corresponding field in EEI

11.4.5.8 Clear Enable Request (CERQ)

11.4.5.8.1 Offset

Register	Offset
CERQ	1Ah

11.4.5.8.2 Function

The CERQ provides a simple memory-mapped mechanism to write 0 to a given field in the ERQ register to disable the DMA request for a given channel. The data value on a register write causes the corresponding field in the ERQ register to be written with 0. Setting the CAER field provides a global clear to 0 function, forcing the entire contents of the ERQ register to be written with 0, disabling all DMA request inputs.

If the NOP field is 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field written with 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

NOTE

Disable a channel's hardware service request at the source before writing 0 to the channel's ERQ field.

11.4.5.8.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0					0	
W	NOP	CAER	0				CERQ	
Reset	0	0	0	0	0	0	0	0

11.4.5.8.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation, ignore the other fields in this register
6 CAER	Clear All Enable Requests 0b - Write 0 to only the ERQ field specified in the CERQ field 1b - Write 0 to all fields in ERQ
5-2 —	Reserved
1-0 CERQ	Clear Enable Request Writes 0 to the corresponding field in ERQ.

11.4.5.9 Set Enable Request (SERQ)

11.4.5.9.1 Offset

Register	Offset
SERQ	1Bh

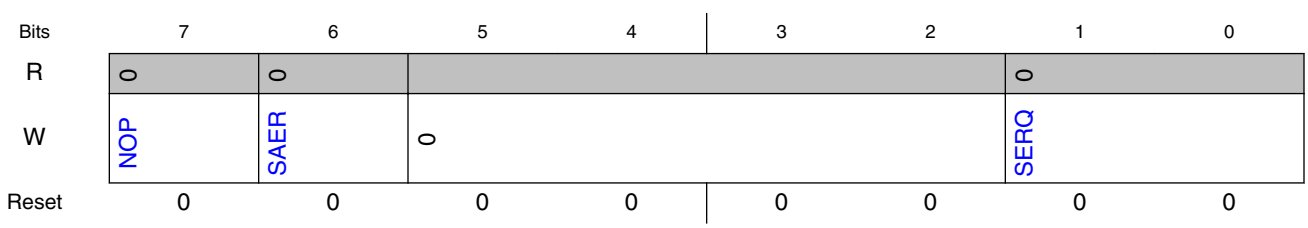
11.4.5.9.2 Function

The SERQ provides a simple memory-mapped mechanism to write 1 to a given field in the ERQ register to enable the DMA request for a given channel. The data value on a register write causes the corresponding field in the ERQ register to be set. Writing 1 to the SAER field provides a global set to 1 function, forcing the entire contents of ERQ register to be 1.

If the NOP field is 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field written with 1 so that these registers are not affected by the write.

Reads of this register returns all zeroes.

11.4.5.9.3 Diagram



11.4.5.9.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation, ignore the other fields in this register
6 SAER	Set All Enable Requests 0b - Write 1 to only the ERQ field specified in the SERQ field 1b - Write 1 to all fields in ERQ
5-2	Reserved

Table continues on the next page...

Field	Function
—	
1-0 SERQ	Set Enable Request Writes 1 to the corresponding field in ERQ.

11.4.5.10 Clear DONE Status Bit (CDNE)

11.4.5.10.1 Offset

Register	Offset
CDNE	1Ch

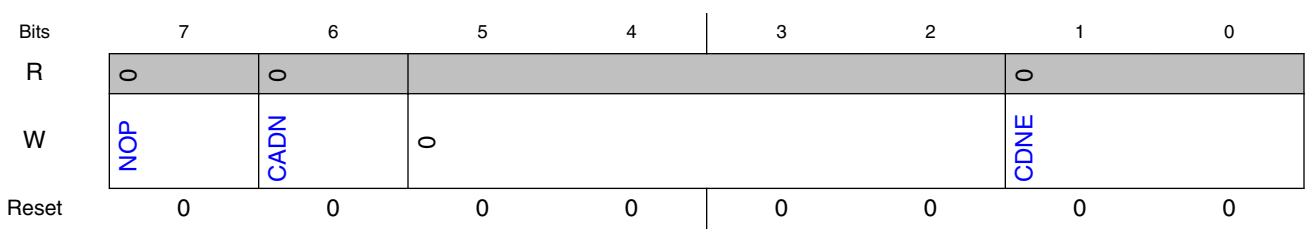
11.4.5.10.2 Function

The CDNE provides a simple memory-mapped mechanism to write 0 to the DONE field in the TCD of the given channel. The data value on a register write causes the DONE field in the corresponding TCD to be written with 0. Writing 1 to the CADN field provides a global clear function, forcing all DONE fields to be written with 0.

If the NOP field is 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field written with 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

11.4.5.10.3 Diagram



11.4.5.10.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation; all other fields in this register are ignored.
6 CADN	Clears All DONE fields 0b - Writes 0 to only the TCDn_CSR[DONE] field specified in the CDNE field 1b - Writes 0 to all bits in TCDn_CSR[DONE]
5-2 —	Reserved
1-0 CDNE	Clear DONE field Writes 0 to the corresponding field in TCDn_CSR[DONE]

11.4.5.11 Set START Bit (SSRT)

11.4.5.11.1 Offset

Register	Offset
SSRT	1Dh

11.4.5.11.2 Function

The SSRT register provides a simple memory-mapped mechanism to write 1 to the START field in the TCD of the given channel. The data value on a register write causes the START field in the corresponding TCD to be written with 1. Writing 1 to the SAST field provides a global set to 1 function, forcing all START fields to be written with 1.

If the NOP field is 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field written with 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

11.4.5.11.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0					0	
W	NOP	SAST	0				SSRT	
Reset	0	0	0	0	0	0	0	0

11.4.5.11.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation; all other fields in this register are ignored.
6 SAST	Set All START fields (activates all channels) 0b - Write 1 to only the TCDn_CSR[START] field specified in the SSRT field 1b - Write 1 to all bits in TCDn_CSR[START]
5-2 —	Reserved
1-0 SSRT	Set START field Sets the corresponding field in TCDn_CSR[START]

11.4.5.12 Clear Error (CERR)

11.4.5.12.1 Offset

Register	Offset
CERR	1Eh

11.4.5.12.2 Function

The CERR provides a simple memory-mapped mechanism to write 0 to a given field in the ERR register to disable the error condition field for a given channel. The given value on a register write causes the corresponding field in the ERR register to be written with 0. Writing 1 to the CAEI field provides a global clear to 0 function, forcing the ERR register contents to be written with 0, clearing all channel error indicators. If the NOP field is 1, the command is ignored. This enables you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

11.4.5.12.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0					0	
W	NOP	CAEI	0				CERR	
Reset	0	0	0	0	0	0	0	0

11.4.5.12.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation; all other fields in this register are ignored.
6 CAEI	Clear All Error Indicators 0b - Write 0 to only the ERR field specified in the CERR field 1b - Write 0 to all fields in ERR
5-2 —	Reserved
1-0 CERR	Clear Error Indicator Writes 0 to the corresponding field in ERR

11.4.5.13 Clear Interrupt Request (CINT)

11.4.5.13.1 Offset

Register	Offset
CINT	1Fh

11.4.5.13.2 Function

The CINT register provides a simple, memory-mapped mechanism to clear a given field in the INT register to disable the interrupt request for a given channel. The given value on a register write causes the corresponding field in the INT register to be cleared. Setting the CAIR field provides a global clear function, forcing the entire contents of the INT to be cleared, disabling all DMA interrupt requests.

If the NOP field is 1, the command is ignored. This enables you to set a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP field set to 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

11.4.5.13.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0					0	
W	NOP	CAIR	0				CINT	
Reset	0	0	0	0	0	0	0	0

11.4.5.13.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation; all other fields in this register are ignored.
6 CAIR	Clear All Interrupt Requests 0b - Clear only the INT field specified in the CINT field 1b - Clear all bits in INT
5-2 —	Reserved
1-0 CINT	Clear Interrupt Request Clears the corresponding field in INT

11.4.5.14 Interrupt Request (INT)

11.4.5.14.1 Offset

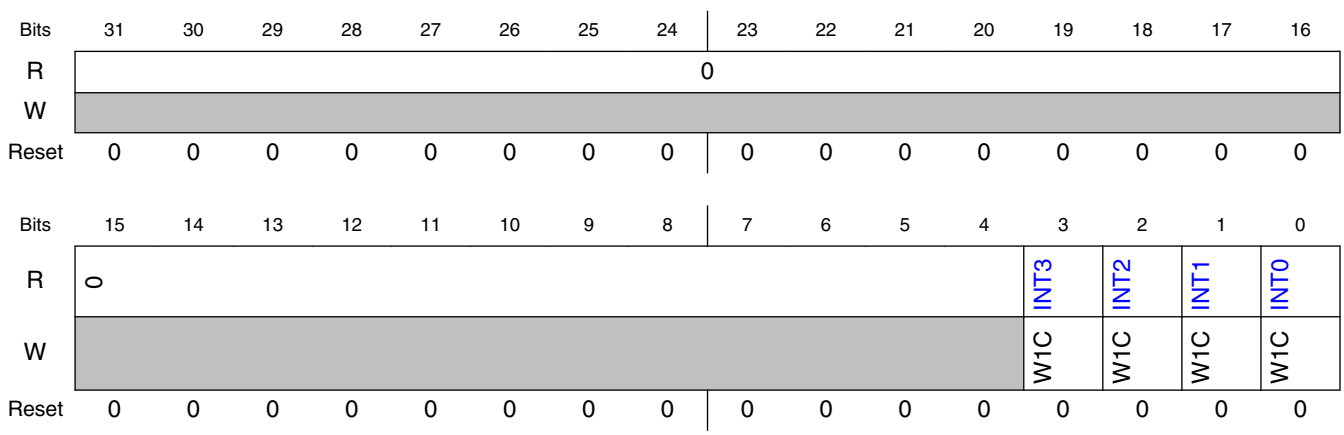
Register	Offset
INT	24h

11.4.5.14.2 Function

The INT register provides a bit map for the 4 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptors, the eDMA engine generates an interrupt on data transfer completion. The outputs of this register are directly routed to the interrupt controller. During the interrupt-service routine associated with any given channel, it is the software's responsibility to write 0 to the appropriate bit, negating the interrupt request. Typically, a write to the CINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT register. On writes to INT, a 1 in any bit position clears the corresponding channel's interrupt request. A 0 in any bit position has no effect on the corresponding channel's current interrupt status. The CINT register is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT register.

11.4.5.14.3 Diagram



11.4.5.14.4 Fields

Field	Function
31-4 —	Reserved
3 INT3	Interrupt Request 3 0b - The interrupt request for channel 3 is cleared 1b - The interrupt request for channel 3 is active
2 INT2	Interrupt Request 2 0b - The interrupt request for channel 2 is cleared 1b - The interrupt request for channel 2 is active
1	Interrupt Request 1

Table continues on the next page...

Field	Function
INT1	0b - The interrupt request for channel 1 is cleared 1b - The interrupt request for channel 1 is active
0 INT0	Interrupt Request 0 0b - The interrupt request for channel 0 is cleared 1b - The interrupt request for channel 0 is active

11.4.5.15 Error (ERR)

11.4.5.15.1 Offset

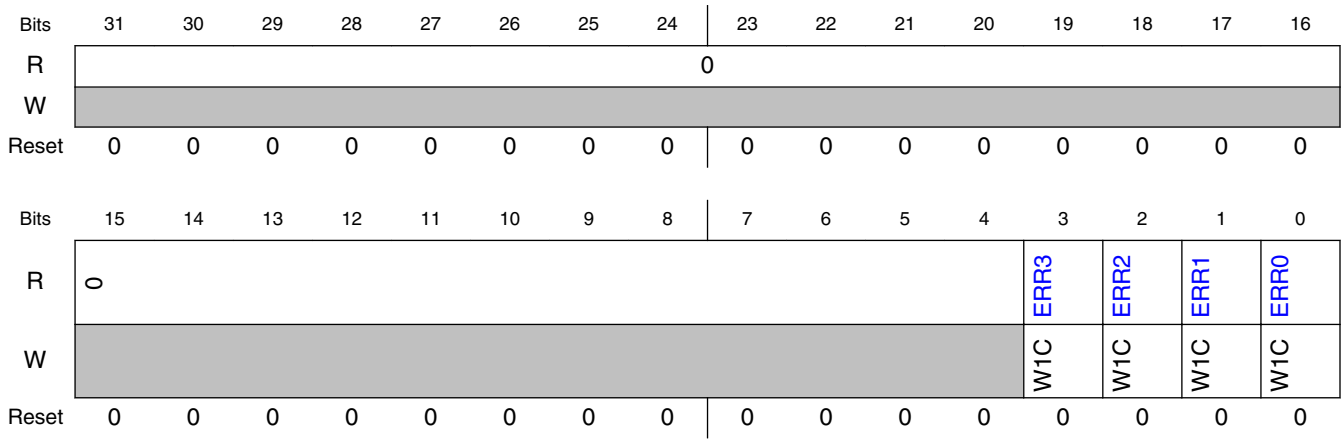
Register	Offset
ERR	2Ch

11.4.5.15.2 Function

The ERR register provides a bit map for the 4 channels, signaling the presence of an error for each channel. The eDMA engine signals the occurrence of an error condition by setting the appropriate field in this register. The outputs of this register are enabled by the contents of the EEI register, and then routed to the interrupt controller. During the execution of the interrupt service routine associated with any DMA errors, it is software's responsibility to reset the appropriate bit to 0, negating the error-interrupt request. Typically, a write to the CERR in the interrupt service routine is used for this purpose. The normal DMA channel completion indicators (setting the TCD DONE field to 1 and the possible generation of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI fields. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a 1 in any bit position clears the corresponding channel's error status. A 0 in any bit position has no effect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be reset to 0.

11.4.5.15.3 Diagram



11.4.5.15.4 Fields

Field	Function
31-4 —	Reserved
3 ERR3	Error In Channel 3 0b - No error in this channel has occurred 1b - An error in this channel has occurred
2 ERR2	Error In Channel 2 0b - No error in this channel has occurred 1b - An error in this channel has occurred
1 ERR1	Error In Channel 1 0b - No error in this channel has occurred 1b - An error in this channel has occurred
0 ERR0	Error In Channel 0 0b - No error in this channel has occurred 1b - An error in this channel has occurred

11.4.5.16 Hardware Request Status (HRS)

11.4.5.16.1 Offset

Register	Offset
HRS	34h

11.4.5.16.2 Function

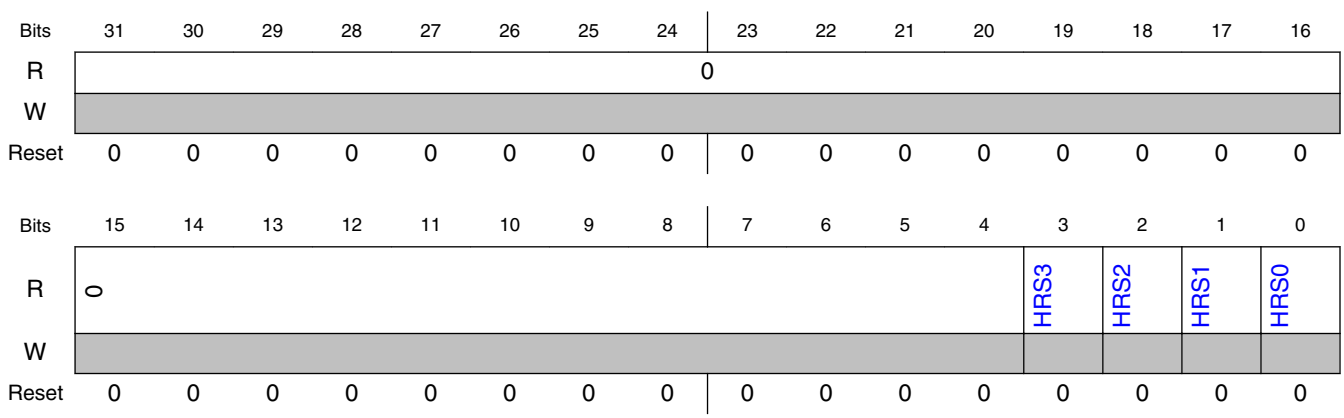
The HRS register provides a bit map for the DMA channels, signaling the presence of a hardware request for each channel. The hardware request status bits reflect the current state of the register and qualified (via the ERQ fields) DMA request signals, as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

NOTE

These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ bits.

Each HRS field for its respective channel is 1 when a hardware request is present on the channel. After the request is completed and channel is free, the HRS field is automatically changed to 0 by hardware.

11.4.5.16.3 Diagram



11.4.5.16.4 Fields

Field	Function
31-4 —	Reserved
3 HRS3	Hardware Request Status Channel 3 0b - A hardware service request for channel 3 is not present 1b - A hardware service request for channel 3 is present
2 HRS2	Hardware Request Status Channel 2 0b - A hardware service request for channel 2 is not present 1b - A hardware service request for channel 2 is present
1	Hardware Request Status Channel 1 0b - A hardware service request for channel 1 is not present

Table continues on the next page...

Field	Function
HRS1	1b - A hardware service request for channel 1 is present
0 HRS0	Hardware Request Status Channel 0 0b - A hardware service request for channel 0 is not present 1b - A hardware service request for channel 0 is present

11.4.5.17 Enable Asynchronous Request in Stop (EARS)

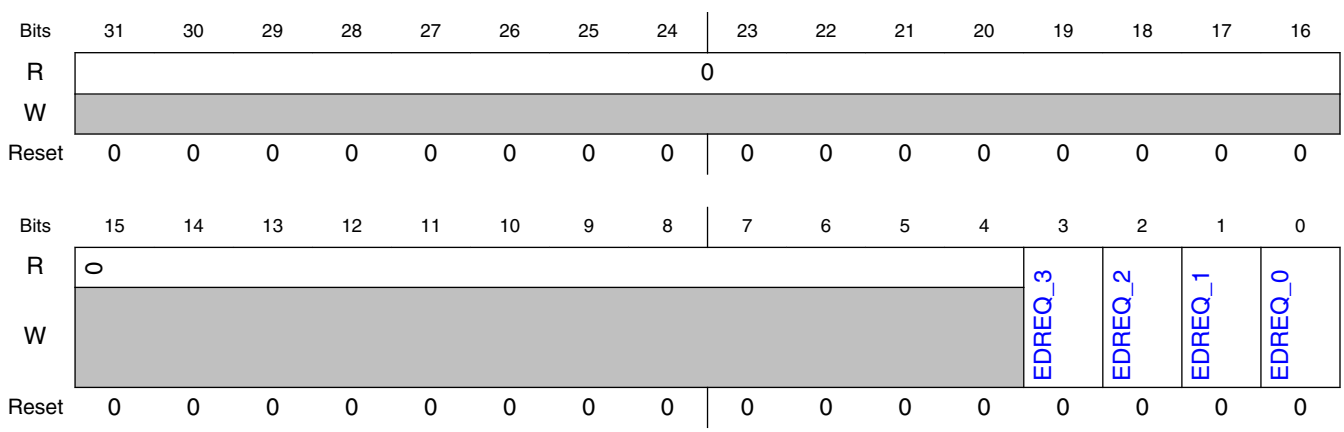
11.4.5.17.1 Offset

Register	Offset
EARS	44h

11.4.5.17.2 Function

The EARS register is used to enable or disable the DMA requests in Enable Request (ERQ) by AND'ing the bits of these two registers.

11.4.5.17.3 Diagram



11.4.5.17.4 Fields

Field	Function
31-4 —	Reserved
3	Enable asynchronous DMA request in stop mode for channel 3.

Table continues on the next page...

Field	Function
EDREQ_3	0b - Disable asynchronous DMA request for channel 3 1b - Enable asynchronous DMA request for channel 3
2 EDREQ_2	Enable asynchronous DMA request in stop mode for channel 2. 0b - Disable asynchronous DMA request for channel 2 1b - Enable asynchronous DMA request for channel 2
1 EDREQ_1	Enable asynchronous DMA request in stop mode for channel 1. 0b - Disable asynchronous DMA request for channel 1 1b - Enable asynchronous DMA request for channel 1
0 EDREQ_0	Enable asynchronous DMA request in stop mode for channel 0. 0b - Disable asynchronous DMA request for channel 0 1b - Enable asynchronous DMA request for channel 0

11.4.5.18 Channel Priority (DCHPRI0 - DCHPRI3)

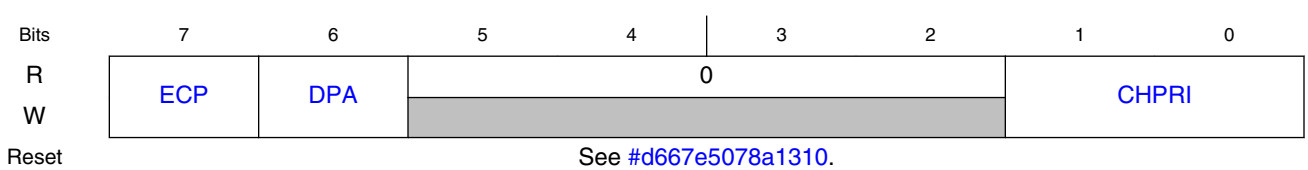
11.4.5.18.1 Offset

Register	Offset
DCHPRI3	100h
DCHPRI2	101h
DCHPRI1	102h
DCHPRI0	103h

11.4.5.18.2 Function

When fixed-priority channel arbitration is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel. The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, and so on. Software must program the channel priorities with unique values; otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 3.

11.4.5.18.3 Diagram



11.4.5.18.4 Register reset values

Register	Reset value
DCHPRI0	00h
DCHPRI1	01h
DCHPRI2	02h
DCHPRI3	03h

11.4.5.18.5 Fields

Field	Function
7 ECP	Enable Channel Preemption. This field resets to 0. 0b - Channel n cannot be suspended by a higher priority channel's service request 1b - Channel n can be temporarily suspended by the service request of a higher priority channel
6 DPA	Disable Preempt Ability. This field resets to 0. 0b - Channel n can suspend a lower priority channel 1b - Channel n cannot suspend any channel, regardless of channel priority
5-2 —	Reserved
1-0 CHPRI	Channel n Arbitration Priority Channel priority when fixed-priority arbitration is enabled.

11.4.5.19 TCD Source Address (TCD0_SADDR - TCD3_SADDR)

11.4.5.19.1 Offset

Register	Offset
TCD0_SADDR	1000h
TCD1_SADDR	1020h
TCD2_SADDR	1040h
TCD3_SADDR	1060h

11.4.5.19.2 Function

This register contains the source address of the transfer.

11.4.5.19.3 Diagram

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

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

11.4.5.19.4 Fields

Field	Function
31-0 SADDR	Source Address Memory address pointing to the source data.

11.4.5.20 TCD Signed Source Address Offset (TCD0_SOFF - TCD3_SOFF)

11.4.5.20.1 Offset

Register	Offset
TCD0_SOFF	1004h
TCD1_SOFF	1024h
TCD2_SOFF	1044h
TCD3_SOFF	1064h

11.4.5.20.2 Diagram

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

11.4.5.20.3 Fields

Field	Function
15-0 SOFF	Source address signed offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

11.4.5.21 TCD Transfer Attributes (TCD0_ATTR - TCD3_ATTR)

11.4.5.21.1 Offset

Register	Offset
TCD0_ATTR	1006h
TCD1_ATTR	1026h
TCD2_ATTR	1046h
TCD3_ATTR	1066h

11.4.5.21.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SMOD					SSIZE			DMOD					DSIZE		
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

11.4.5.21.3 Fields

Field	Function
15-11 SMOD	Source Address Modulo Any non-zero value in this field defines a specific address range specified to be the value after SADDR + SOFF calculation is performed on the original register value. Setting this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range. 0_0000b - Source address modulo feature is disabled 0_0001b-1_1111b - Value defines address range used to set up circular data queue
10-8	Source data transfer size

Table continues on the next page...

Field	Function
SSIZE	**NOTE:** 1. Using a reserved value causes a configuration error. 2. The eDMA defaults to privileged data access for all transactions. 000b - 8-bit 001b - 16-bit 010b - 32-bit 011b - Reserved 100b - 16-byte 101b - Reserved 110b - Reserved 111b - Reserved
7-3 DMOD	Destination Address Modulo See the SMOD definition.
2-0 DSIZE	Destination data transfer size See the SSIZE definition.

11.4.5.22 TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD0_NBYTES_MLNO - TCD3_NBYTES_MLNO)

11.4.5.22.1 Offset

Register	Offset
TCD0_NBYTES_MLNO	1008h
TCD1_NBYTES_MLNO	1028h
TCD2_NBYTES_MLNO	1048h
TCD3_NBYTES_MLNO	1068h

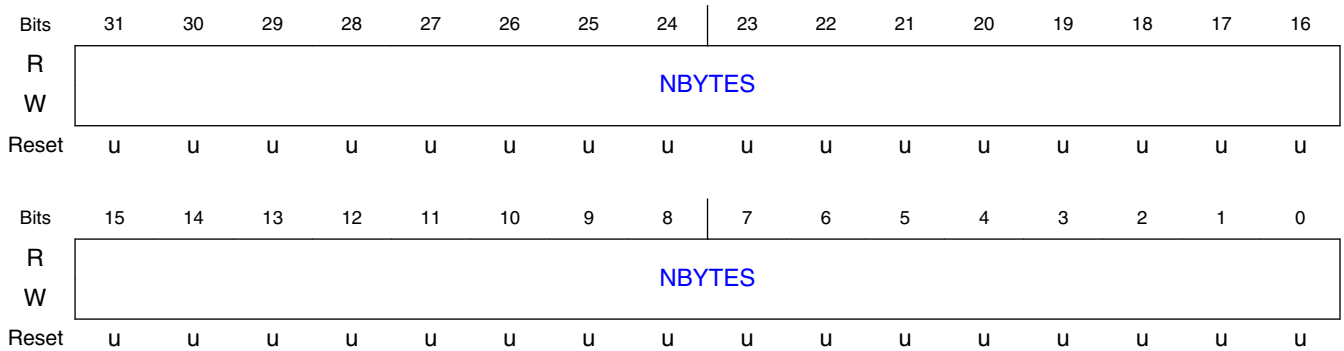
11.4.5.22.2 Function

This register, or one of the next two registers (TCD_NBYTES_MLOFFNO, TCD_NBYTES_MLOFFYES), that defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, is enabled but not used for this channel, or is enabled and used.

TCD word 2 is defined as follows if minor loop mapping is disabled (CR[EMLM] = 0).

If minor loop mapping is enabled, see the TCD_NBYTES_MLOFFNO and TCD_NBYTES_MLOFFYES register descriptions for the definition of TCD word 2.

11.4.5.22.3 Diagram



11.4.5.22.4 Fields

Field	Function
31-0 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes are performed until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, and the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. **NOTE:** An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer.

11.4.5.23 TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD0_NBYTES_MLOFFNO - TCD3_NBYTES_MLOFFNO)

11.4.5.23.1 Offset

Register	Offset
TCD0_NBYTES_MLOFFNO	1008h
TCD1_NBYTES_MLOFFNO	1028h
TCD2_NBYTES_MLOFFNO	1048h
TCD3_NBYTES_MLOFFNO	1068h

11.4.5.23.2 Function

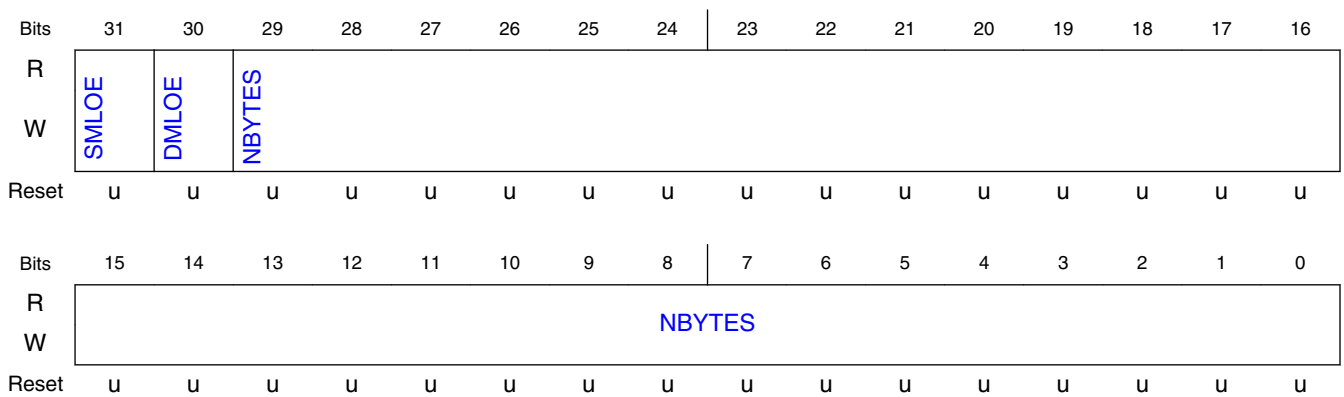
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFYES), that defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, is enabled but not used for this channel, or is enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- SMLOE = 0 and DMLOE = 0

If minor loop mapping is enabled and SMLOE = 1 or DMLOE = 1, refer to the TCD_NBYTES_MLOFFYES register description. If minor loop mapping is disabled, refer to the TCD_NBYTES_MLNO register description.

11.4.5.23.3 Diagram



11.4.5.23.4 Fields

Field	Function
31 SMLOE	Source Minor Loop Offset Enable Specifies whether the minor loop offset is applied to the source address when the minor loop completes. 0b - The minor loop offset is not applied to the SADDR 1b - The minor loop offset is applied to the SADDR
30 DMLOE	Destination Minor Loop Offset Enable Specifies whether the minor loop offset is applied to the destination address when the minor loop completes. 0b - The minor loop offset is not applied to the DADDR 1b - The minor loop offset is applied to the DADDR
29-0	Minor Byte Transfer Count

Field	Function
NBYTES	Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes are performed until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, and the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

11.4.5.24 TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD0_NBYTES_MLOFFYES - TCD3_NBYTES_MLOFFYES)

11.4.5.24.1 Offset

Register	Offset
TCD0_NBYTES_MLOFFYES	1008h
TCD1_NBYTES_MLOFFYES	1028h
TCD2_NBYTES_MLOFFYES	1048h
TCD3_NBYTES_MLOFFYES	1068h

11.4.5.24.2 Function

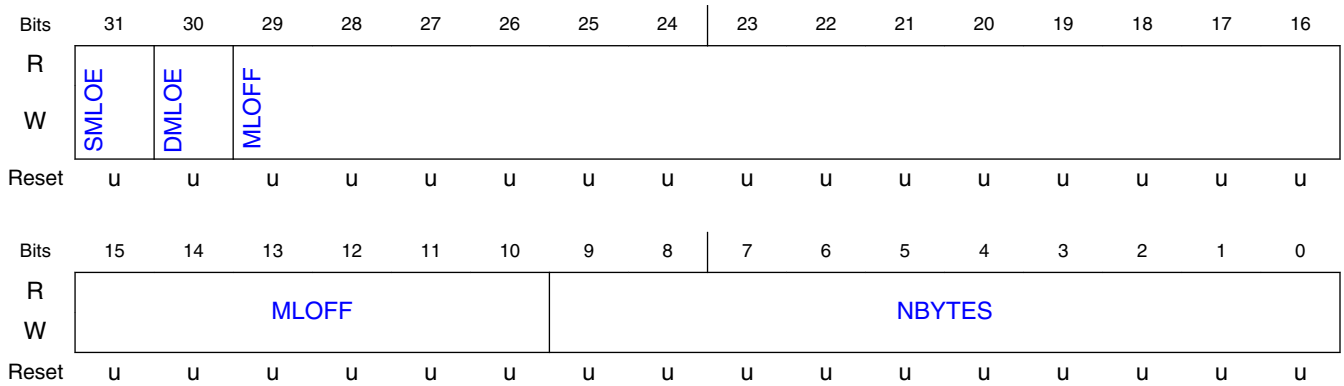
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFNO), that defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, is enabled but not used for this channel, or is enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- Minor loop offset is enabled (SMLOE or DMLOE = 1)

If minor loop mapping is enabled and SMLOE = 0 and DMLOE = 0, refer to the TCD_NBYTES_MLOFFNO register description. If minor loop mapping is disabled, refer to the TCD_NBYTES_MLNO register description.

11.4.5.24.3 Diagram



11.4.5.24.4 Fields

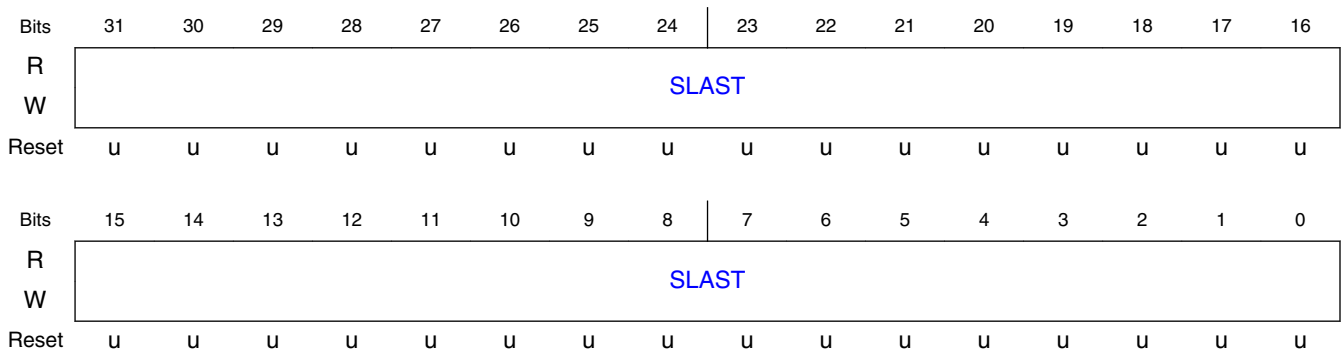
Field	Function
31 SMLOE	Source Minor Loop Offset Enable Specifies whether the minor loop offset is applied to the source address when the minor loop completes. 0b - The minor loop offset is not applied to the SADDR 1b - The minor loop offset is applied to the SADDR
30 DMLOE	Destination Minor Loop Offset Enable Specifies whether the minor loop offset is applied to the destination address when the minor loop completes. 0b - The minor loop offset is not applied to the DADDR 1b - The minor loop offset is applied to the DADDR
29-10 MLOFF	If SMLOE = 1 or DMLOE = 1, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes.
9-0 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes are performed until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, and the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

11.4.5.25 TCD Last Source Address Adjustment (TCD0_SLAST - TCD3_SLAST)

11.4.5.25.1 Offset

Register	Offset
TCD0_SLAST	100Ch
TCD1_SLAST	102Ch
TCD2_SLAST	104Ch
TCD3_SLAST	106Ch

11.4.5.25.2 Diagram



11.4.5.25.3 Fields

Field	Function
31-0 SLAST	Last Source Address Adjustment Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure. This register uses two's complement notation; the overflow bit is discarded.

11.4.5.26 TCD Destination Address (TCD0_DADDR - TCD3_DADDR)

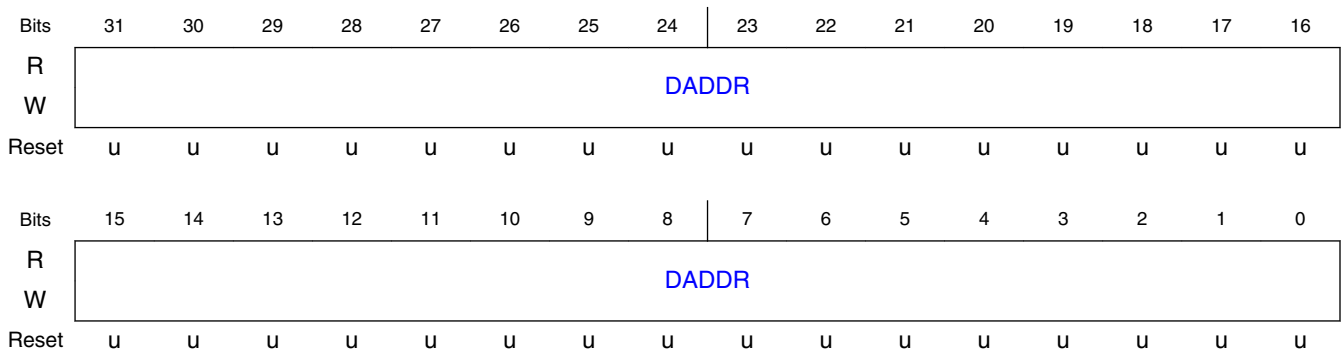
11.4.5.26.1 Offset

Register	Offset
TCD0_DADDR	1010h
TCD1_DADDR	1030h
TCD2_DADDR	1050h
TCD3_DADDR	1070h

11.4.5.26.2 Function

This register contains the destination address of the transfer.

11.4.5.26.3 Diagram



11.4.5.26.4 Fields

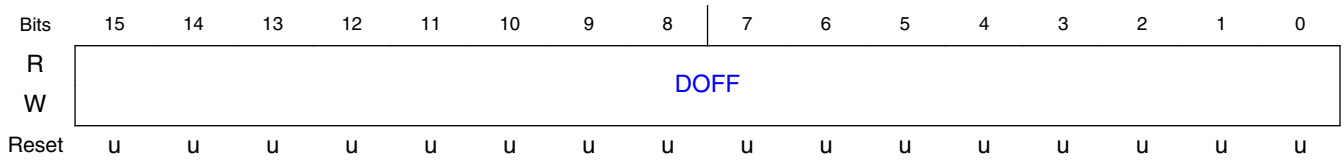
Field	Function
31-0 DADDR	Destination Address Memory address pointing to the destination data.

11.4.5.27 TCD Signed Destination Address Offset (TCD0_DOFF - TCD3_DOFF)

11.4.5.27.1 Offset

Register	Offset
TCD0_DOFF	1014h
TCD1_DOFF	1034h
TCD2_DOFF	1054h
TCD3_DOFF	1074h

11.4.5.27.2 Diagram



11.4.5.27.3 Fields

Field	Function
15-0 DOFF	Destination Address Signed Offset Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

11.4.5.28 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD3_CITER_ELINKNO)

11.4.5.28.1 Offset

Register	Offset
TCD0_CITER_ELINKNO	1016h
TCD1_CITER_ELINKNO	1036h
TCD2_CITER_ELINKNO	1056h
TCD3_CITER_ELINKNO	1076h

11.4.5.28.2 Function

This register contains the minor-loop channel-linking configuration and the channel's current iteration count. It is the same register as TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD3_CITER_ELINKYES), but its fields are defined differently based on the state of the ELINK field. If the ELINK field is 0, this register is defined as follows.

11.4.5.28.3 Diagram

11.4.5.28.4 Fields

Field	Function
15 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this field enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets TCDn_CSR[START] of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This field must be equal to BITER[ELINK]; otherwise, a configuration error is reported. 0b - Channel-to-channel linking is disabled 1b - Channel-to-channel linking is enabled
14-0 CITER	Current Major Iteration Count This field is the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations. It optionally generates an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** 1. When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. 2. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

11.4.5.29 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD3_CITER_ELINKYES)

11.4.5.29.1 Offset

Register	Offset
TCD0_CITER_ELINKYES	1016h

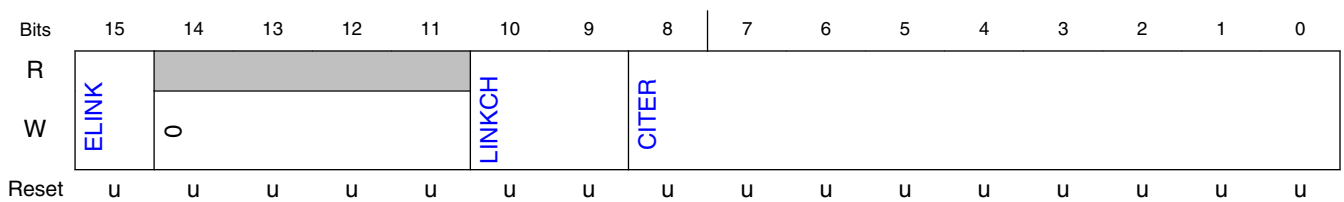
Table continues on the next page...

Register	Offset
TCD1_CITER_ELINKYES	1036h
TCD2_CITER_ELINKYES	1056h
TCD3_CITER_ELINKYES	1076h

11.4.5.29.2 Function

This register contains the minor-loop channel-linking configuration and the channel's current iteration count. It is the same register as TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD3_CITER_ELINKNO), but its fields are defined differently based on the state of the ELINK field. If the ELINK field is 1, this register is defined as follows.

11.4.5.29.3 Diagram



11.4.5.29.4 Fields

Field	Function
15 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this field enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets TCDn_CSR[START] of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This field must be equal to BITER[ELINK]; otherwise, a configuration error is reported. 0b - Channel-to-channel linking is disabled 1b - Channel-to-channel linking is enabled
14-11 —	Reserved
10-9 LINKCH	Minor Loop Link Channel Number

Table continues on the next page...

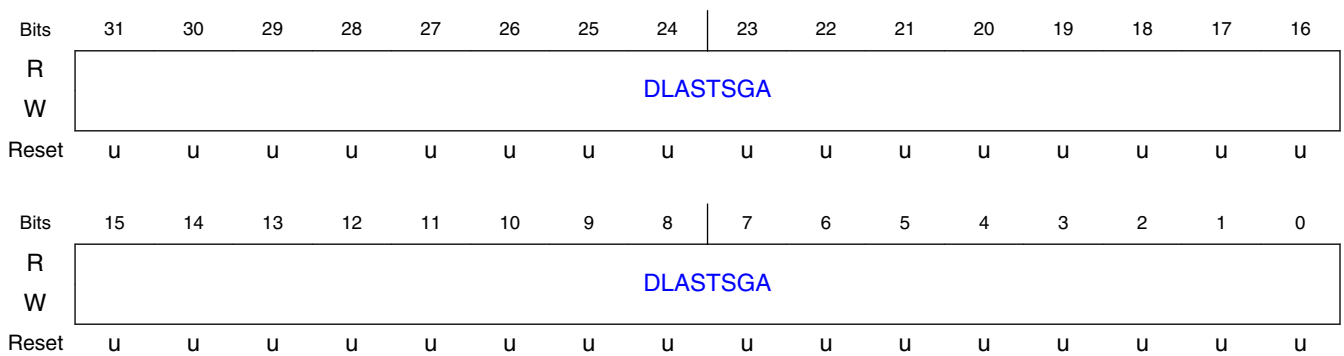
Field	Function
	If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by this field, by setting that channel's TCDn_CSR[START].
8-0 CITER	Current Major Iteration Count This field is the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations. It optionally generates an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** 1. When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. 2. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

11.4.5.30 TCD Last Destination Address Adjustment/Scatter Gather Address (TCD0_DLASTSGA - TCD3_DLASTSGA)

11.4.5.30.1 Offset

Register	Offset
TCD0_DLASTSGA	1018h
TCD1_DLASTSGA	1038h
TCD2_DLASTSGA	1058h
TCD3_DLASTSGA	1078h

11.4.5.30.2 Diagram



11.4.5.30.3 Fields

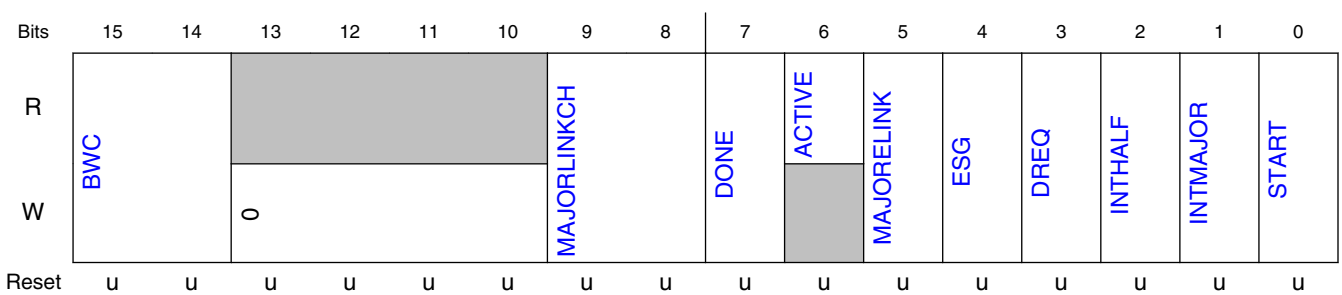
Field	Function
31-0 DLASTSGA	Destination last address adjustment, or next memory address TCD for channel (scatter/gather) If (TCDn_CSR[ESG] = 0) then: • This is the adjustment value added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure. • This field uses two's complement notation for the final destination address adjustment. Otherwise: • This address points to the beginning of a 0-modulo 32-byte region containing the next TCD to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo 32-byte; otherwise a configuration error is reported.

11.4.5.31 TCD Control and Status (TCD0_CSR - TCD3_CSR)

11.4.5.31.1 Offset

Register	Offset
TCD0_CSR	101Ch
TCD1_CSR	103Ch
TCD2_CSR	105Ch
TCD3_CSR	107Ch

11.4.5.31.2 Diagram



11.4.5.31.3 Fields

Field	Function
15-14	Bandwidth Control

Table continues on the next page...

Field	Function
BWC	Throttles the amount of bus bandwidth consumed by the eDMA. Generally, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch. **NOTE:** If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency. **NOTE:** When executing a large, zero wait-stated memory-to-memory transfer, insert bandwidth control using the TCD_CSR[BWC] bits to avoid: • Starvation of another master accessing the memory. • Any delay in writing a TCD during the transfer. 00b - No eDMA engine stalls 01b - Reserved 10b - eDMA engine stalls for 4 cycles after each R/W 11b - eDMA engine stalls for 8 cycles after each R/W
13-10 —	Reserved
9-8 MAJORLINKCH	Major Loop Link Channel Number If (MAJORELINK = 0) then: • No channel-to-channel linking, or chaining, is performed after the major loop counter is exhausted. Otherwise: • After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's START bit.
7 DONE	Channel Done This field indicates whether the eDMA has completed the major loop. The eDMA engine sets the value of this field to 1 when the CITER count reaches zero. The value of this field is reset to 0 by the hardware (when the channel is activated) or by software. **NOTE:** This field must be 0 to write the MAJORELINK or ESG fields.
6 ACTIVE	Channel Active This field indicates whether the channel is currently in execution. The eDMA sets the value of this field to 1 when channel service begins, and resets it to 0 as the minor loop completes or when any error condition is detected.
5 MAJORELINK	Enable channel-to-channel linking on major loop complete As the channel completes the major loop, this field controls linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets TCDn_CSR[START] of the specified channel. **NOTE:** To support the dynamic linking coherency model, this field is forced to zero when written to when TCDn_CSR[DONE] is set. 0b - Channel-to-channel linking is disabled 1b - Channel-to-channel linking is enabled
4 ESG	Enable Scatter/Gather Processing As the channel completes the major loop, this field controls scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo 32-bit address containing a 32-byte data structure loaded as the TCD into local memory. **NOTE:** To support the dynamic scatter/gather coherency model, this field is forced to zero when written to when TCDn_CSR[DONE] is set. 0b - The current channel's TCD is normal format 1b - The current channel's TCD specifies a scatter gather format

Table continues on the next page...

Field	Function
3 DREQ	Disable Request If the value of this field is 1, eDMA hardware automatically writes 0 to the corresponding ERQ field when the current major iteration count reaches zero. 0b - The channel's ERQ field is not affected 1b - The channel's ERQ field value changes to 0 when the major loop is complete
2 INTHALF	Enable an interrupt when major counter is half complete. If the value of this field is 1, the channel generates an interrupt request by setting the appropriate field in the INT register when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered, also known as ping-pong, schemes or other types of data movement where the processor needs an early indication of the transfer's progress. **NOTE:** If BITER = 1, do not use INTHALF. Use INTMAJOR instead. 0b - Half-point interrupt is disabled 1b - Half-point interrupt is enabled
1 INTMAJOR	Enable an interrupt when major iteration count completes. If the value of this field is 1, the channel generates an interrupt request by setting the appropriate field in the INT when the current major iteration count reaches zero. 0b - End of major loop interrupt is disabled 1b - End of major loop interrupt is enabled
0 START	Channel Start If the value of this field is 1, the channel is requesting service. eDMA hardware automatically writes 0 to this field after the channel begins execution. 0b - Channel is not explicitly started 1b - Channel is explicitly started via a software initiated service request

11.4.5.32 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_BITER_ELINKNO - TCD3_BITER_ELINKNO)

11.4.5.32.1 Offset

Register	Offset
TCD0_BITER_ELINKNO	101Eh
TCD1_BITER_ELINKNO	103Eh
TCD2_BITER_ELINKNO	105Eh
TCD3_BITER_ELINKNO	107Eh

11.4.5.32.2 Function

If TCDn_BITER[ELINK] is 0, the TCDn_BITER register is defined as follows.

11.4.5.32.3 Diagram

11.4.5.32.4 Fields

Field	Function
15 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this field enables linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets TCDn_CSR[START] of the specified channel. If channel linking is disabled, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0b - Channel-to-channel linking is disabled 1b - Channel-to-channel linking is enabled
14-0 BITER	Starting Major Iteration Count As the TCD is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

11.4.5.33 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_BITER_ELINKYES - TCD3_BITER_ELINKYES)

11.4.5.33.1 Offset

Register	Offset
TCD0_BITER_ELINKYES	101Eh
TCD1_BITER_ELINKYES	103Eh

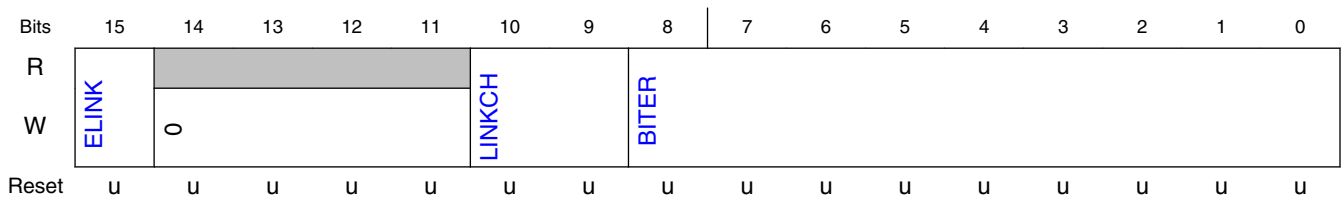
Table continues on the next page...

Register	Offset
TCD2_BITER_ELINKYES	105Eh
TCD3_BITER_ELINKYES	107Eh

11.4.5.33.2 Function

If TCDn_BITER[ELINK] is 1, the TCDn_BITER register is defined as follows.

11.4.5.33.3 Diagram



11.4.5.33.4 Fields

Field	Function
15 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this field enables linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets TCDn_CSR[START] of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0b - Channel-to-channel linking is disabled 1b - Channel-to-channel linking is enabled
14-11 —	Reserved
10-9 LINKCH	Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field, by setting that channel's TCDn_CSR[START]. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.
8-0 BITER	Starting major iteration count

Field	Function
	As the TCD is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

Chapter 12
DMA Channel Multiplexer (DMAMUX)

12.1 Chip-specific information for this module

12.1.1 eDMA

This device includes a eDMA request mux that allows up to 63 DMA request signals to be mapped to any of the 4 DMA channels. Because of the mux, there is no hard correlation between any of the DMA request sources and a specific DMA channel. Some of the modules support Asynchronous DMA operation, as indicated by the last column in the following DMA source assignment table.

Table 12-1. DMA Channel Assignment

Source Number	Source Module	Logic	Source Description	Async DMA capable
0	–		Channel Disabled **NOTE:** Configuring a DMA Channel to seelct source 0 or any of the reserved sources diable that DMA channel.	
1	–		Reserved	
2	SCI0_RF		Receive	Yes
3	SCI0_TE		Transmit	Yes
4	SCI1_RF		Receive	Yes
5	SCI1_TE		Transmit	Yes
6	–		Reserved	
7	–		Reserved	
8	–		Reserved	
9	–		Reserved	
10	–		Reserved	

Table continues on the next page...

Table 12-1. DMA Channel Assignment (continued)

Source Number	Source Module	Logic	Source Description	Async DMA capable
11	–		Reserved	
12	SPI0_RF		SPI Receive Full	
13	SPI0_TE		SPI Transmit Empty	
14	–		Reserved	
15	–		Reserved	
16	–		Reserved	
17	–		Reserved	
18	LPI2C0_M_TX	OR'ed all req	LPI2C0_M_TX_Req	
	LPI2C0_S_Tx		LPI2C0_S_TX_Req	
	LPI2C0_M_RX		LPI2C0_M_RX_Req	
	LPI2C0_S_RX		LPI2C0_S_RX_Req	
19	LPI2C1_M_TX	OR'ed all req	LPI2C1_M_TX_Req	
	LPI2C1_S_Tx		LPI2C1_S_TX_Req	
	LPI2C1_M_RX		LPI2C1_M_RX_Req	
	LPI2C1_S_RX		LPI2C1_S_RX_Req	
20	–		Reserved	
21	–		Reserved	
22	eFlexPWMA0_CP		Submodule 0 Capture DMA Req	
23	eFlexPWMA1_CP		Submodule 1 Capture DMA Req	
24	eFlexPWMA2_CP		Submodule 2 Capture DMA Req	
25	eFlexPWMA3_CP		Submodule 3 Capture DMA Req	
26	eFlexPWMA0_WR	OR'ed all req	SubModule0 Value write DMA Req	
	eFlexPWMA1_WR		SubModule1 Value write DMA Req	
	eFlexPWMA2_WR		SubModule2 Value write DMA Req	
	eFlexPWMA3_WR		SubModule3 Value write DMA Req	
27	eFlexPWMA0_WR		SubModule0 Value write DMA Req	
28	eFlexPWMA1_WR		SubModule1 Value write DMA Req	
29	eFlexPWMA2_WR		SubModule2 Value write DMA Req	
30	eFlexPWMA3_WR		SubModule3 Value write DMA Req	
31	–		Reserved	
32	TMRA0_CP	OR'ed all req	TMRA0 Capture	

Table continues on the next page...

Table 12-1. DMA Channel Assignment (continued)

Source Number	Source Module	Logic	Source Description	Async DMA capable
	TMRA0_CMP1		TMRA0 Compare1	
33	TMRA0_CMP2		TMRA0 Compare2	
34	TMRA1_CP	OR'ed all req	TMRA1 Capture	
	TMRA1_CMP1		TMRA1 Compare1 T	
35	TMRA1_CMP2		MRA1 Compare2	
36	TMRA2_CP	OR'ed all req	TMRA2 Capture	
	TMRA2_CMP1		TMRA2 Compare1	
37	TMRA2_CMP2		TMRA2 Compare2	
38	TMRA3_CP	OR'ed all req	TTMRA3 Capture	
	TMRA3_CMP1		TMRA3 Compare1	
39	TMRA3_CMP2		TMRA3 Compare2	
40	–		Reserved	
41	–		Reserved	
42	–		Reserved	
43	–		Reserved	
44	–		Reserved	
45	–		Reserved	
46	–		Reserved	
47	–		Reserved	
48	ADCA_ES		ADCA End of Scan or ADCA Conversion Result Ready [1]	
49	ADCB_ES		ADCB End of Scan or ADCB Conversion Result Ready [1]	
50	DACA_FIFO (8 deep)		12bit DAC FIFO Water Mark	
51	–		Reserved	
52	CMPA		CMP Toggle	Yes
53	CMPB		CMP Toggle	Yes
54	CMPC		CMP Toggle	Yes
55	CMPD		CMP Toggle	Yes
56	XBAR_DSC0		XBAR DMA Req 0	Yes
57	XBAR_DSC1		XBAR DMA Req 1	Yes
58	XBAR_DSC2		XBAR DMA Req 2	yes
59	XBAR_DSC3		XBAR DMA Req 3	Yes
60	DMA_MUX		Always enabled	
61	DMA_MUX		Always enabled	
62	DMA_MUX		Always enabled	
63	DMA_MUX		Always enabled	

1. See ADC_CTRL3[DMASRC] for details.

12.2 Introduction

12.2.1 Overview

The Direct Memory Access Multiplexer (DMAMUX) routes DMA sources, called slots, to any of the 4 DMA channels. See the chip-specific information to know the detailed source numbers. This process is illustrated in the following figure.

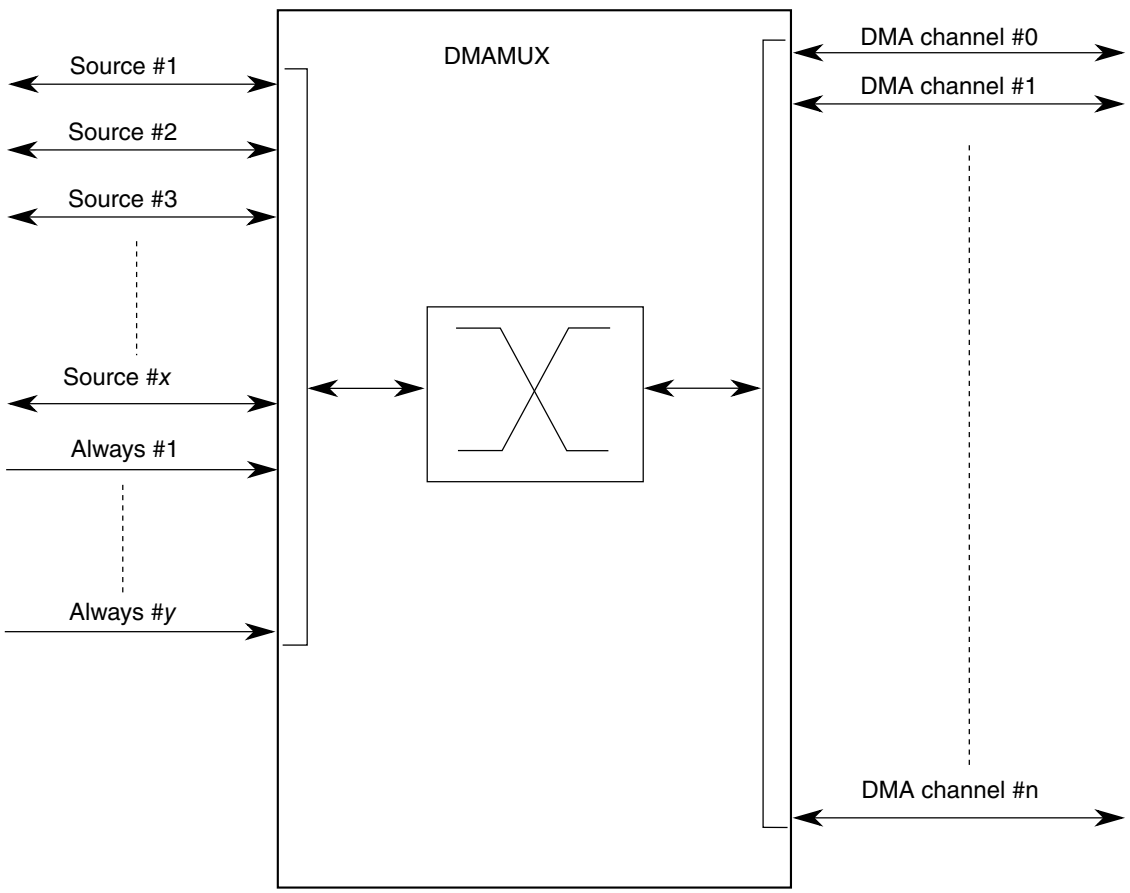


Figure 12-1. DMAMUX block diagram

12.2.2 Features

Following are the features of DMAMUX module.

- Up to 59 peripheral slots and up to 4 always-on slots can be routed to 4 channels.
- 4 independently selectable DMA channel routers.
- Each channel router can be assigned to one of the possible peripheral DMA slots or to one of the always-on slots.

12.2.3 Modes of operation

The DMAMUX module supports the following operating modes.

- Disabled mode: In this mode, the DMA channel is disabled. Because disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMA channel MUX. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place.
- Normal mode: In this mode, a DMA source is routed directly to the specified DMA channel. The operation of the DMAMUX in this mode is completely transparent to the system.

12.3 Functional description

The primary purpose of the DMAMUX is to provide flexibility in the system's use of the available DMA channels.

As such, configuration of the DMAMUX is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in Enabling and configuring sources is followed, the configuration of the DMAMUX may be changed during the normal operation of the system.

The DMAMUX channels implement only the normal routing functionality.

12.3.1 Always-enabled DMA sources

In addition to the peripherals that can be used as DMA sources, there are 4 additional DMA sources that are always enabled. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the sources that are always enabled provide no such "throttling" of the data transfers. These sources are most useful in the following cases:

- Performing DMA transfers to/from GPIO—Moving data from/to one or more GPIO pins.
- Performing DMA transfers from memory to memory: Moving data from memory to memory, typically as fast as possible, sometimes with software activation.

- Performing DMA transfers from memory to the external bus, or vice-versa: Similar to memory to memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation: Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, an always-enabled DMA source can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the minor loop require that a new start event be sent. This can either be a new software activation, or a transfer request from the DMA channel MUX. The options for doing this are as follows.

- Transfer all data in a single minor loop: By configuring the DMA to transfer all of the data in a single minor loop (that is, major loop counter = 1), no reactivation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will impose on the system. For this option, the DMA channel must be disabled in the DMA channel MUX.
- Use explicit software reactivation: In this option, the DMA is configured to transfer the data using both minor and major loops, but the processor is required to reactivate the channel by writing to the DMA registers after every minor loop. For this option, the DMA channel must be disabled in the DMA channel MUX.
- Use an always-enabled DMA source: In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMA channel MUX does the channel reactivation. For this option, the DMA channel should be enabled and pointing to an "always enabled" source.

12.3.2 DMA sources with cancel rewind capability

If the DMA sources (or peripherals) determine the current packet being received is corrupt or incomplete, then they signal the DMA to discard the packet through DMA Channel MUX. Discarding the packet involves two procedures: terminating the current data transfer and restoring the DMA's program state back to its beginning—thus being ready for a new packet. This allows the processor to handle other critical tasks (or stay in low-power mode) while the DMA automatically discards the bad packet. Cancel rewind capability is supported on each of the DMA channels and for every DMA source (or peripheral) except "always-enabled" DMA sources.

12.4 Initialization/application information

This section provides instructions for initializing the DMA channel MUX.

12.4.1 Reset

The reset state of each individual bit is shown in Memory map/register definition. In summary, after reset, all channels are disabled and must be explicitly enabled before use.

12.4.2 Enabling and configuring sources

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
:
:
*CHCFG1 = 0x00;                                  /* Clear all the fields of CHCFG1 register */
*CHCFG1 = 0xC5;                                  /* ENBL = 1  DMA Channel is enabled */
                                                 /* TRIG = 1  Triggering is enabled */
                                                 /* SOURCE = 5  DMA Source 5 is selected */
```

To enable a source, the following steps can be used:

1. Determine the DMA channel with which the source will be associated.
2. Clear the CHCFG[ENBL] field of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that CHCFG[ENBL] is set.

To configure source #5 transmit for use with DMA channel 1, as an example, the following steps can be used:

1. Write 0x00 to CHCFG1.
2. Configure channel 1 in the DMA, including enabling the channel.
3. Write 0x85 to CHCFG1.

The following code example illustrates steps 1 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
:
:
*CHCFG1 = 0x00;                                  /* Clear all the fields of CHCFG1 register */
*CHCFG1 = 0x85;                                  /* ENBL = 1  DMA Channel is enabled */
                                                 /* TRIG = 0  Triggering is disabled */
                                                 /* SOURCE = 5  DMA Source 5 is selected */
```

To disable a source:

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCFG registers. Additionally, some module-specific configuration may be necessary. See the eDMA chapter for more details.

To switch the source of a DMA channel:

1. Disable the DMA channel in the DMA and reconfigure the channel for the new source.
2. Clear the CHCFG[ENBL] bit of the DMA channel.
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] field is set.

To switch DMA channel 8 from source #5 transmit to source #7 transmit:

1. In the DMA configuration registers, disable DMA channel 8 and reconfigure it to handle the transfers to peripheral slot 7. This example assumes channel 8 doesn't have triggering capability.
2. Write 0x00 to CHCFG8.
3. Write 0x87 to CHCFG8.

The following code example illustrates steps 2 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
:
:
*CHCFG8 = 0x00;                                 /* Clear all the fields of CHCFG1 register */
*CHCFG8 = 0x87;                                 /* ENBL = 1  DMA Channel is enabled */
                                                /* TRIG = 0  Triggering is disabled */
                                                /* SOURCE = 7  DMA Source 7 is selected */
```

12.5 Memory map/register definition

This section provides a detailed description of all memory-mapped registers in the DMAMUX.

12.5.1 DMAMUX register descriptions

NOTE

The base address and offsets for these registers are presented in terms of bytes.

12.5.1.1 DMAMUX memory map

DMAMUX base address: 1_C760h

Offset	Register	Width (In bits)	Access	Reset value
0h - 3h	Channel Configuration register (CHCFG0 - CHCFG3)	8	RW	00h

12.5.1.2 Channel Configuration register (CHCFG0 - CHCFG3)

12.5.1.2.1 Offset

Register	Offset
CHCFG0	0h
CHCFG1	1h
CHCFG2	2h
CHCFG3	3h

12.5.1.2.2 Function

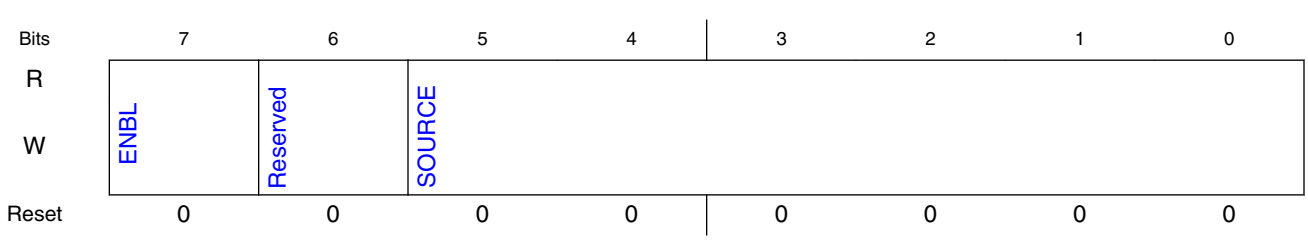
Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

NOTE

Setting multiple CHCFG registers with the same source value will result in unpredictable behavior. This is true, even if a channel is disabled (ENBL==0).

Before changing the source settings, a DMA channel must be disabled via CHCFGn[ENBL].

12.5.1.2.3 Diagram



12.5.1.2.4 Fields

Field	Function
7 ENBL	DMA Channel Enable Enables the DMA channel. 0b - DMA channel is disabled. This mode is primarily used during configuration of the DMAMUX. The DMA has separate channel enables/disables, which should be used to disable or reconfigure a DMA channel. 1b - DMA channel is enabled
6 —	Reserved This read/write field does not affect the functionality of the device and should not be used.
5-0 SOURCE	DMA Channel Source (Slot) Specifies which DMA source, if any, is routed to a particular DMA channel. See the chip-specific DMAMUX information for details about the peripherals and their slot numbers.

Chapter 13
Power Management Controller (PMC)

13.1 Introduction

13.1.1 Overview

Power Management Controller (PMC) contains the on-chip voltage regulators, as well as Power-On-Reset (POR) and Low Voltage Interrupt (LVI) power monitoring circuitry. On-chip regulators are stable and have sufficient capacity and dynamic response allowing device to operate correctly at all time and under all combinations of specified loads, frequencies, voltages, temperatures and fabrication process variations. POR generates a reset signal when power is applied to the device. It ensures that the device starts operating in a known state. LVI mechanism monitors VDD and generates the interrupts when VDD drops below the preset voltage thresholds. CPU response to LVI can place the device in safe state before supply voltage drops below the operating voltage.

13.1.2 Features

- A voltage regulator, called small regulator, generates 2.7 V and 1.2 V regulated supply for analog components.
- A large voltage regulator generates 1.2 V regulated supply for core digital logic.
- LVI_2p7 low voltage interrupt generated when VDD drops consistently below 2.7 V.
- LVI_2p2 low voltage interrupt generated when VDD drops consistently below 2.2 V.
- Power On Reset (POR) asserts when VDD drops below 2.0 V.
- POR deasserts only when VDD is consistently above 2.7 V.
- POR, LVI_2p7, and LVI_2p2 levels have 50-100 mV of internal hysteresis.
- Large regulator has standby mode which reduces its power consumption but limits the current it can supply to the part.
- Flag to indicate when the small regulator's 2.7 V supply is ready to be used.

The assumption is made that power supply voltages move relatively slowly with respect to the system clocks, allowing the chip some control over its future operation.

13.1.3 Modes of Operation

The PMC implements power supply regulation, power on reset, and low voltage detection functions. Power regulation includes a small regulator and a high power large regulator.

The large regulator generates the 1.2V digital supply which is used by IO cells, core logic, Flash and RAM. The small regulator generates the 2.7V and 1.2V analog supply which is used by various analog modules. The low voltage detect uses the supply VDD to generate a raw POR reset which releases at VDD=2.0V, an LVI_2p2 low voltage alarm which releases at VDD=2.2V and an LVI_2p7 low voltage alarm which releases at VDD=2.7V. The PMC uses the POR and low voltage detect signals to provide a flexible infrastructure for detecting VDD changes relative to the two low voltage detect levels and invoking the low voltage interrupt.

The SR27PDN, SR27STDBY, and LRSTDBY inputs are controlled from memory mapped register fields in the SIM. Standby mode reduces a regulator's drive capacity but also reduces its power consumption. LRSTDBY will control large regulator standby mode. SR27STDBY will control standby mode for the 2.7V supply from the small regulator. SR27PDN will control the powerdown of the 2.7V supply from the small regulator and takes precedence over SR27STDBY.

NOTE

The controls of SR27PDN, SR27STDBY and LRSTDBY are taken over by advanced power mode controls if enabled in SIM_PWRMODE register.

13.1.4 Block Diagram

The internal organization of the PMC is illustrated in Figure 13-1.

NOTE

The deglitch flops must be clocked by a continuously running clock so these interrupts can wake the part up if it is in stop mode.

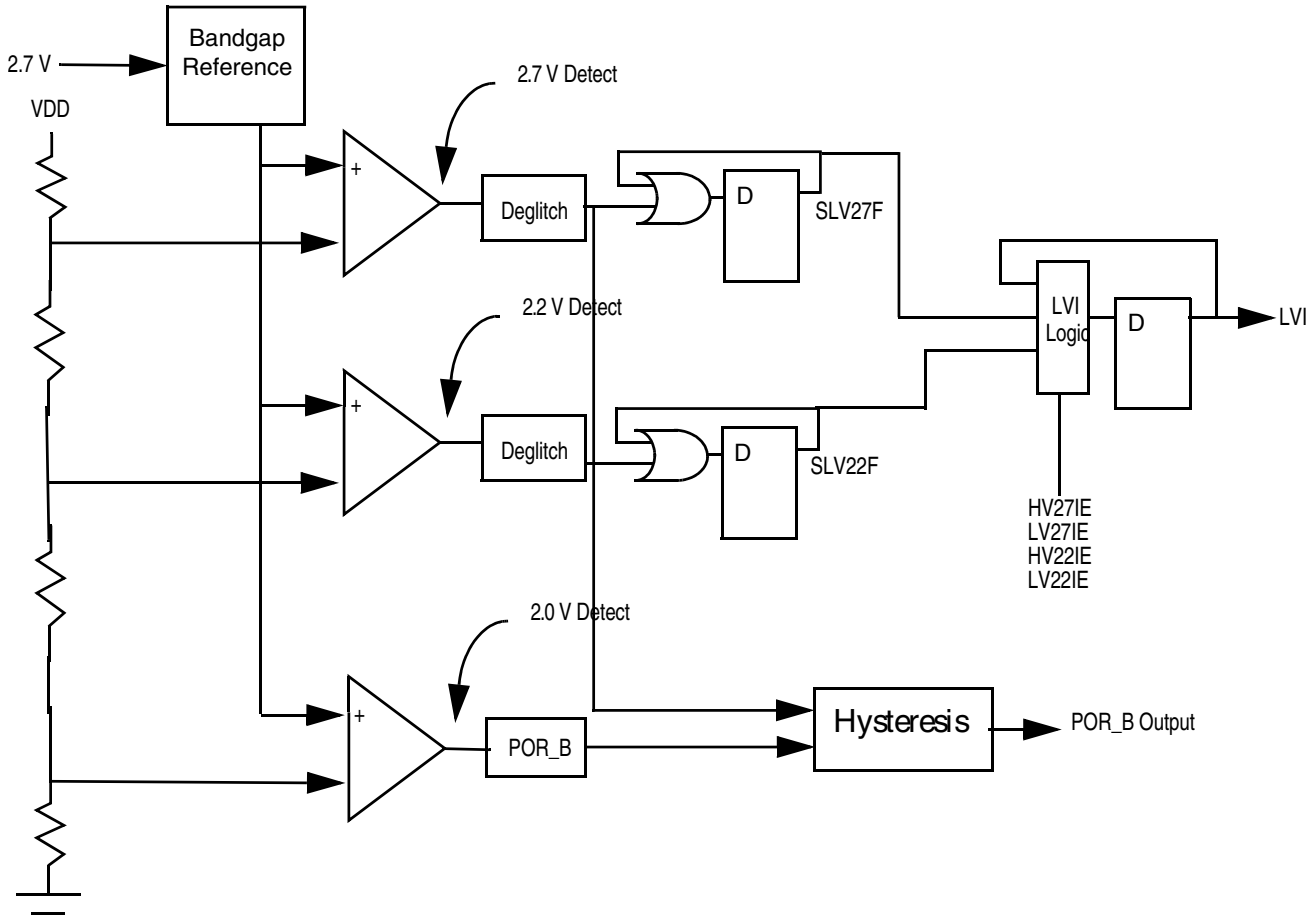


Figure 13-1. PMC Block Diagram

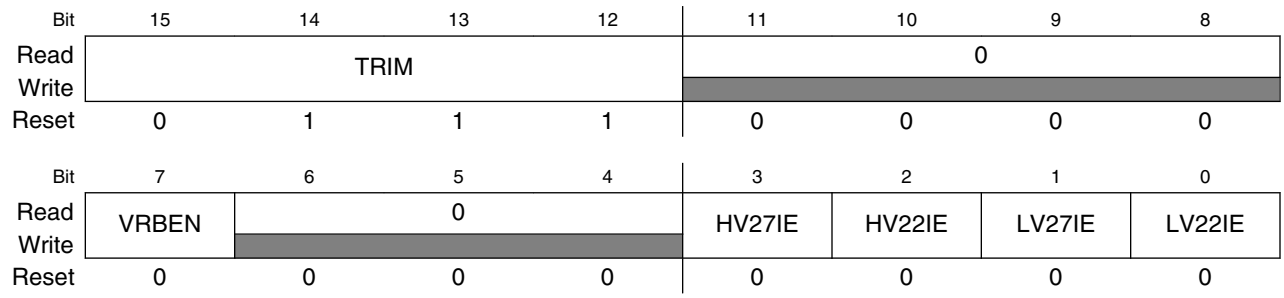
13.2 Memory Map and Register Descriptions

PMC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E2A0	Control Register (PMC_CTRL)	16	R/W	7000h	13.2.1/342
E2A1	Status Register (PMC_STS)	16	w1c	0020h	13.2.2/343

13.2.1 Control Register (PMC_CTRL)

Address: E2A0h base + 0h offset = E2A0h



PMC_CTRL field descriptions

Field	Description
15–12 TRIM	Bandgap Trim This field is used to trim the bandgap reference in the regulator. Its reset state is mid-range. **NOTE:** The bandgap trim is a factory trimmed value. User is prohibited to alter it. **NOTE:** The bandgap trim is automatically loaded from flash memory during reset.
11–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 VRBEN	Voltage Reference Buffer Enable This bit enables a buffer that drives the 1.2 V bandgap reference to the ADC. This bit should be enabled if the user wants to calibrate the ADC using the 1.2 V reference. The bit may be disabled to save power when ADC calibration is not being performed. 0 Disable voltage reference buffering. 1 Enable voltage reference buffering.
6–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 HV27IE	2.7 V High Voltage Interrupt Enable This bit allows the STS[LVI] bit to be set when the STS[LV27F] bit is clear. While LV27F is high (VDD is below 2.7 V), set this bit and an LVI interrupt is generated when LV27F becomes low (VDD becomes greater than 2.7 V). 0 Disable setting the high voltage interrupt. 1 Enable setting the high voltage interrupt.
2 HV22IE	2.2 V High Voltage Interrupt Enable This bit allows the STS[LVI] bit to be set when the STS[LV22F] bit is clear. While LV22F is high (VDD is below 2.2 V), set this bit and an LVI interrupt is generated when LV22F becomes low (VDD becomes greater than 2.2 V). 0 Disable setting the high voltage interrupt. 1 Enable setting the high voltage interrupt.

Table continues on the next page...

PMC_CTRL field descriptions (continued)

Field	Description
1 LV27IE	2.7 V Low Voltage Interrupt Enable This bit allows the STS[LVI] bit to be set when the STS[LV27F] bit is set. While LV27F is low (VDD is above 2.7 V), set this bit and an LVI interrupt is generated when LV27F becomes high (VDD becomes lower than 2.7 V). 0 Disable setting the low voltage interrupt. 1 Enable setting the low voltage interrupt.
0 LV22IE	2.2 V Low Voltage Interrupt Enable This bit allows the STS[LVI] bit to be set when the STS[LV22F] bit is set. While LV22F is low (VDD is above 2.2 V), set this bit and an LVI interrupt is generated when LV22F becomes high (VDD becomes lower than 2.2 V). 0 Disable setting the low voltage interrupt. 1 Enable setting the low voltage interrupt.

13.2.2 Status Register (PMC_STS)

Address: E2A0h base + 1h offset = E2A1h

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0		SR27	LVI	SLV27F	SLV22F	LV27F	LV22F
Write				w1c	w1c	w1c		
Reset	0	0	1	0	0	0	0	0

PMC_STS field descriptions

Field	Description
15–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SR27	Small Regulator 2.7 V Active Flag This read-only bit indicates that the small regulator 2.7 V supply, which supplies power to the crystal oscillator, relaxation oscillator, PLL, and duty cycle corrector, is powered up and ready to use. The small regulator 2.7 V supply is powered down using the SIM's PWR[SR27PDN] bits. The small regulator requires 100 µs to wake up from power down. Since the maximum clock frequency is 200 kHz while in VLPRUN mode (the only mode where the small regulator is powered down), this flag is set about 20 clock cycles after the SIM's PWR[SR27PDN] bits turn on the small regulator 2.7 V supply. Verify this flag is set before turning on any of the devices that are powered by the small regulator's 2.7 V supply.

Table continues on the next page...

PMC_STS field descriptions (continued)

Field	Description
	0 The small regulator 2.7 V supply is not ready to be used. 1 The small regulator 2.7 V supply is ready to be used.
4 LVI	Low Voltage Interrupt This bit is the low voltage interrupt. This bit is set by any of several conditions: • STS[LV22F] and CTRL[LV22IE], • STS[LV27F] and CTRL[LV27IE], • $\overline{\text{STS[LV22F]}}$ and CTRL[HV22IE], or • $\overline{\text{STS[LV27F]}}$ and CTRL[HV27IE] Once set, this bit remains set until a 1 is written to this bit position or a reset occurs. Writing a 0 has no effect. Following is an explanation of how to configure these controls to detect rising and/or falling transitions of VDD relative to 2.7 V or 2.2 V. When STS[LV27F] is low (VDD is above 2.7 V), both HV22IE and HV27IE should be cleared. Setting LV27IE and clearing LV22IE asserts LVI when VDD falls below 2.7 V. Setting LV22IE and clearing LV27IE asserts LVI if VDD falls below 2.2 V. Setting both LV27IE and LV22IE has the same effect as setting only LV27IE, which asserts LVI when VDD falls below 2.7 V. When STS[LV22F] and STS[LV27F] are both high (VDD is below 2.2 V), both LV22IE and LV27IE should be cleared. Setting HV22IE and clearing HV27IE asserts LVI when VDD rises above 2.2 V. Setting HV27IE and clearing HV22IE asserts LVI if VDD rises above 2.7 V. Setting both HV22IE and HV27IE has the same effect as setting only HV22IE, which asserts LVI when VDD rises above 2.2 V. When STS[LV27F] is high (VDD is below 2.7 V) and STS[LV22F] is low (VDD is above 2.2 V), both LV27IE and HV22IE should be cleared. Setting HV27IE and clearing LV22IE asserts LVI only if VDD rises above 2.7 V. Setting LV22IE and clearing HV22IE asserts LVI only if VDD falls below 2.2 V. Setting both HV27IE and LV22IE asserts LVI if VDD either falls below 2.2 V or rises above 2.7 V. 0 Low voltage interrupt cleared. 1 Low voltage interrupt asserted.
3 SLV27F	Sticky 2.7 V Low Voltage Flag This sticky bit indicates that the 3.3 V supply dropped below the 2.7 V level at some point. Once set, this bit remains set until a 1 is written to this bit position or a reset occurs. Writing a 0 has no effect. 0 3.3 V supply has not dropped below the 2.7 V threshold. 1 3.3 V supply has dropped below the 2.7 V threshold.
2 SLV22F	Sticky 2.2 V Low Voltage Flag This sticky bit indicates that the 3.3 V supply dropped below the 2.2 V level at some point. Once set, this bit remains set until a 1 is written to this bit position or a reset occurs. Writing a 0 has no effect. 0 3.3 V supply has not dropped below the 2.2 V threshold. 1 3.3 V supply has dropped below the 2.2 V threshold.
1 LV27F	2.7 V Low Voltage Flag This read-only bit indicates that the 3.3 V supply is currently below the 2.7 V level. This bit may reset itself if the supply voltage rises above the threshold. 0 3.3 V supply is not below the 2.7 V threshold. 1 3.3 V supply is below the 2.7 V threshold.

Table continues on the next page...

PMC_STS field descriptions (continued)

Field	Description
0 LV22F	2.2 V Low Voltage Flag This read-only bit indicates that the 3.3 V supply is currently below the 2.2 V level. This bit may reset itself if the supply voltage rises above the threshold. 0 3.3 V supply is not below the 2.2 V threshold. 1 3.3 V supply is below the 2.2 V threshold.

13.3 Functional Description

As shown in the block diagram, VDD is processed by the low voltage detect to generate an internal power-on reset and LVI_2p2 and LVI_2p7 low voltage detection signals. The PMC performs deglitch functions on these signals and uses them to generate noise-free versions of the raw POR and low voltage detects. These are available in both raw and sticky (registered) forms.

The Low Voltage Interrupt (LVI) logic in the PMC uses four interrupt enables and the two low voltage detects for VDD = 2.7 V and VDD = 2.2 V to generate a single low voltage interrupt. By properly configuring these four interrupt enables based upon the current VDD voltage range (as indicated by two low voltage detect signals), the LVI can be configured to assert on any possible falling or rising transition of VDD through either of the two fixed LVI levels.

The POR circuit is designed to assert the internal POR reset until VDD is above 2.0 V. The hysteresis function of the PMC, however, keeps the POR_B output asserted until LVI_2p7 detection indicates that VDD is above 2.7 V. The POR_B output will not reassert until internal VDD falls below 2.0 V.

The deglitch blocks are essentially strings of 4 flops in series. The outputs of all 4 must agree before the STS[LV27F] or STS[LV22F] bits change state. This is intended to prevent the LVI circuitry from responding to momentary glitches brought about as a result of normal operation.

Figure 13-2 illustrates operation of the POR_B versus low voltage detect circuits. Low voltage detection circuits indicate the voltage on VDD, the external 3.3 V supply, relative to 2.7 V and 2.2 V.

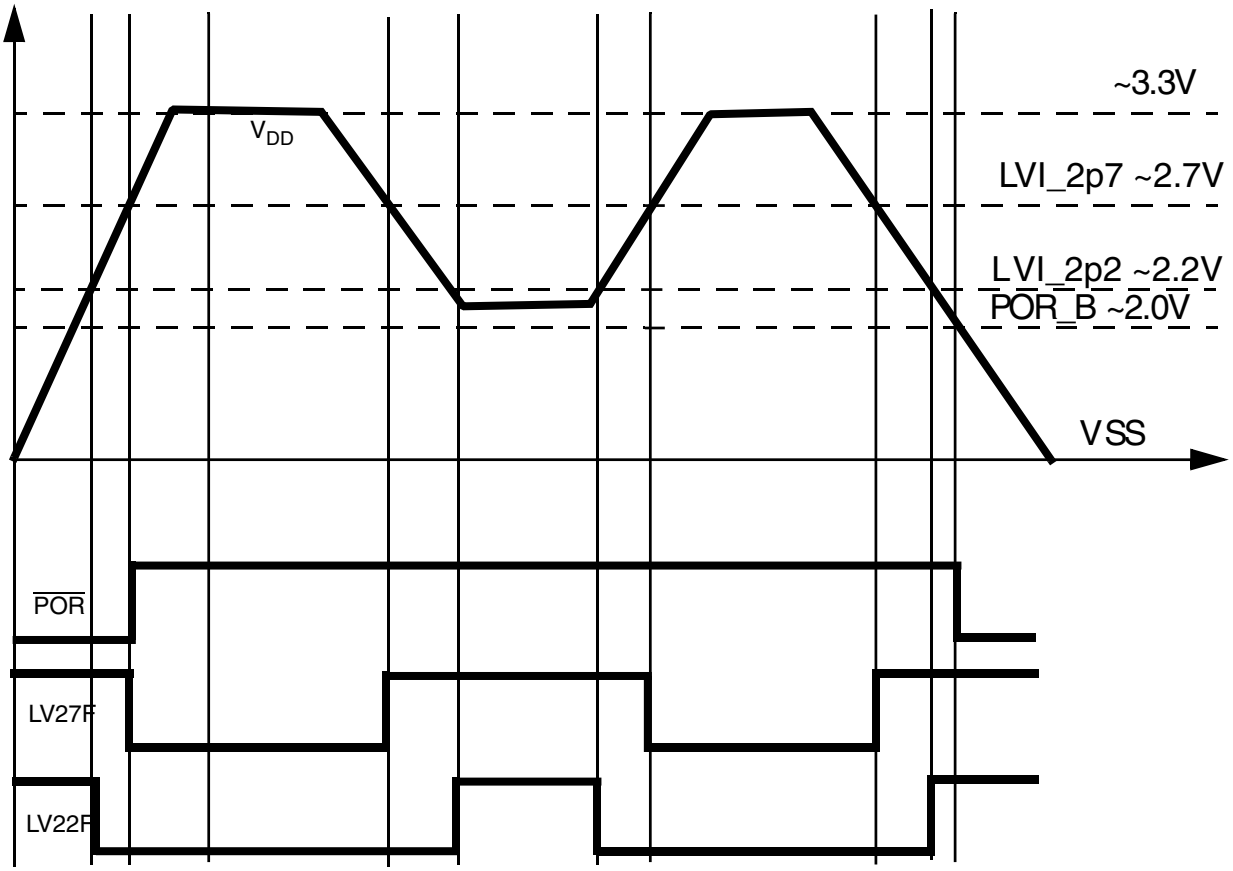


Figure 13-2. POR_B Versus Low-Voltage Detects

13.4 Resets

Table 13-1. Reset Summary

Reset	Source	Characteristics
POR_B	This module	When asserted, indicates that the supply voltage is too low for reliable operation.
RESET_B	SIM	Used to reset the power management controller registers. This signal is usually derived from POR_B and other chip reset sources.

13.5 Clocks

Clock	Source	Used by
IPBus Clock	SIM	Used by glitch filter during normal operation and by control registers at all times.
Oscillator Clock	Oscillator	Used by glitch filter during stop mode and during power on resets.

13.6 Interrupts

Table 13-2. Interrupt Summary

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
STS[LVI]	STS[SLV27F], STS[SLV22F]	CTRL[LV27IE], CTRL[LV22IE], CTRL[HV27IE], CTRL[HV22IE]	Low Voltage Interrupt	See Memory Map and Register Descriptions for details

Chapter 14
Event Generator (EVTG)

14.1 About this module

14.1.1 Introduction

The EVTG(Event Generator) module mainly includes two parts: Two AND/OR/INVERT (known simply as the AOI) modules and one configurable Flip-Flop. It supports the generation of a configurable number of EVENT signals. The two AOI combinational expressions share the four associated EVTG inputs: An, Bn, Cn, and Dn. The Flip-Flop can be configured to make the two expressions act as the Reset port, Set port or D port, CLK port or simply go through to EVTG output with FF(Flip-Flop) bypassed.

This module is designed to be integrated in conjuction with one or more inter-peripheral crossbar switch (XBAR_DSC) modules. A crossbar switch is typically used to select the 4 EVTG inputs from among available peripheral outputs and GPIO signals. The EVTG outputs are typically used as additional inputs to a second crossbar switch, adding to it the ability to connect to its outputs an arbitrary 4-input boolean function of its other inputs.

This module is a slave peripheral module connecting event input indicators from a variety of device modules and generating event output signals that can be routed to an inter-peripheral crossbar switch or other peripherals. Its programming model is accessed through the standard IPS (Sky Blue) slave interface. The module is designed to be very configurable in terms of the integrated AOI functionality and Flip-Flop variety.

14.1.2 Features

The EVTG includes the following features:

- Highly programmable module for creating combinational boolean events
  - Each EVTG has four inputs and two outputs

- Each AOI evaluates a combinational boolean expression as the sum of four products where each product term includes all four selected input sources available as true or complement values
- Each EVTG has two groups of AOI to generate two combinational expressions
- The two outputs can operate as hardware trigger signals or for other purpose.

- One flexible FF can be configured as RS, D-FF, T-FF, JK-FF, Latch.
- Programmable filter to remove AOI output glitch
- All logics are synchronous in bus clk domain
- Memory-mapped device connected to the slave peripheral (IPS) bus
  - Programming model organized per channel for simplified software

14.1.3 Block diagram

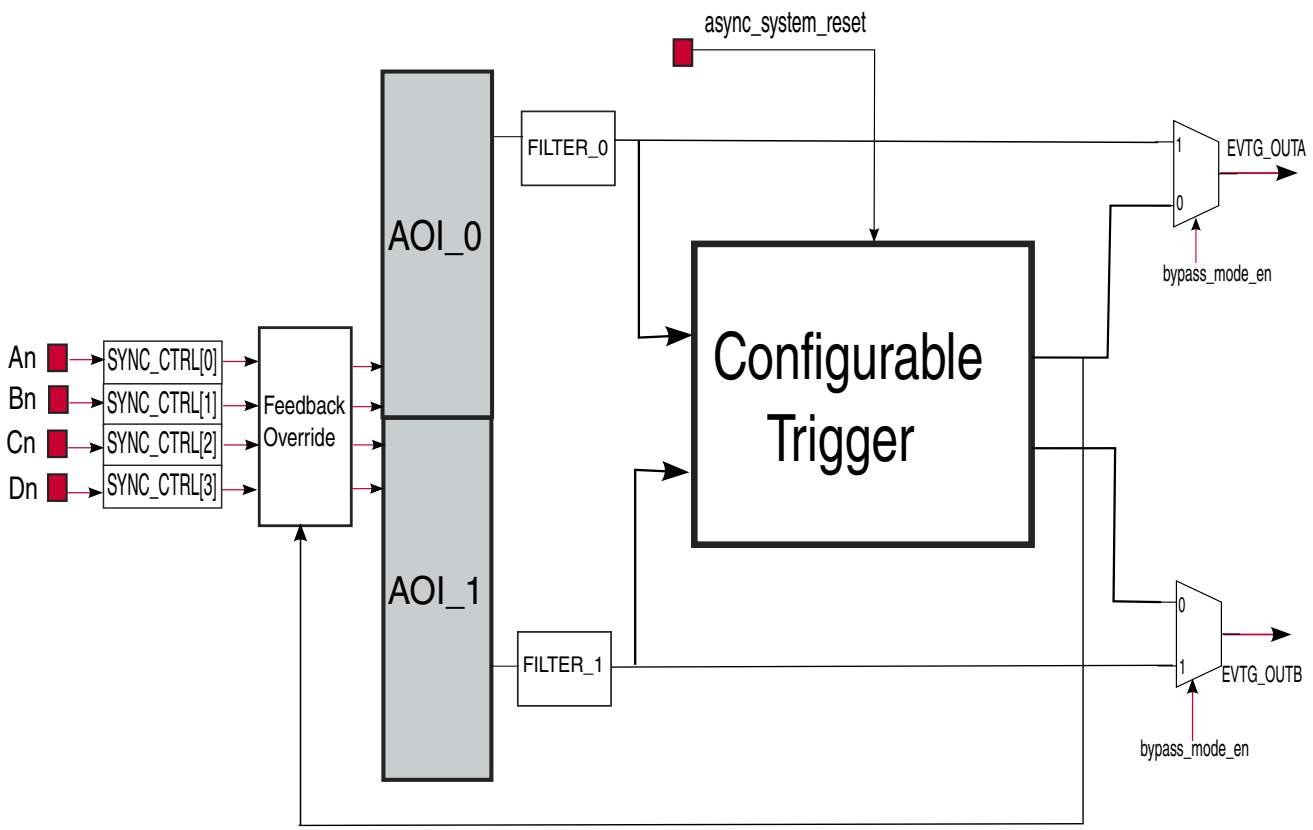


Figure 14-1. EVTG block diagram

14.2 Signals

EVTG module has no external I/O signals.

14.3 Memory Map and register definition

In accordance with DSC core, EVTG module is also word addressed, where a word is 16 bits. Functionality for accesses of other widths is undefined.

14.3.1 EVTG register descriptions

14.3.1.1 EVTG memory map

EVTG base address: E380h

Offset	Register	Width (In bits)	Access	Reset value
0h	AOI0 Boolean Function Term 0 and 1 Configuration Register (EVTG0_AOI0_BFT01)	16	RW	0000h
1h	AOI0 Boolean Function Term 2 and 3 Configuration Register (EVTG0_AOI0_BFT23)	16	RW	0000h
2h	AOI1 Boolean Function Term 0 and 1 Configuration Register (EVTG0_AOI1_BFT01)	16	RW	0000h
3h	AOI1 Boolean Function Term 2 and 3 Configuration Register (EVTG0_AOI1_BFT23)	16	RW	0000h
5h	Control/Status Register (EVTG0_CTRL)	16	RW	0000h
6h	AOI0 Output Filter Register (EVTG0_AOI0_FILT)	16	RW	0000h
7h	AOI1 Output Filter Register (EVTG0_AOI1_FILT)	16	RW	0000h
8h	AOI0 Boolean Function Term 0 and 1 Configuration Register (EVTG1_AOI0_BFT01)	16	RW	0000h
9h	AOI0 Boolean Function Term 2 and 3 Configuration Register (EVTG1_AOI0_BFT23)	16	RW	0000h
Ah	AOI1 Boolean Function Term 0 and 1 Configuration Register (EVTG1_AOI1_BFT01)	16	RW	0000h
Bh	AOI1 Boolean Function Term 2 and 3 Configuration Register (EVTG1_AOI1_BFT23)	16	RW	0000h
Dh	Control/Status Register (EVTG1_CTRL)	16	RW	0000h
Eh	AOI0 Output Filter Register (EVTG1_AOI0_FILT)	16	RW	0000h
Fh	AOI1 Output Filter Register (EVTG1_AOI1_FILT)	16	RW	0000h
10h	AOI0 Boolean Function Term 0 and 1 Configuration Register (EVTG2_AOI0_BFT01)	16	RW	0000h
11h	AOI0 Boolean Function Term 2 and 3 Configuration Register (EVTG2_AOI0_BFT23)	16	RW	0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
12h	AOI1 Boolean Function Term 0 and 1 Configuration Register (EVTG2_AOI1_BFT01)	16	RW	0000h
13h	AOI1 Boolean Function Term 2 and 3 Configuration Register (EVTG2_AOI1_BFT23)	16	RW	0000h
15h	Control/Status Register (EVTG2_CTRL)	16	RW	0000h
16h	AOI0 Output Filter Register (EVTG2_AOI0_FILT)	16	RW	0000h
17h	AOI1 Output Filter Register (EVTG2_AOI1_FILT)	16	RW	0000h
18h	AOI0 Boolean Function Term 0 and 1 Configuration Register (EVTG3_AOI0_BFT01)	16	RW	0000h
19h	AOI0 Boolean Function Term 2 and 3 Configuration Register (EVTG3_AOI0_BFT23)	16	RW	0000h
1Ah	AOI1 Boolean Function Term 0 and 1 Configuration Register (EVTG3_AOI1_BFT01)	16	RW	0000h
1Bh	AOI1 Boolean Function Term 2 and 3 Configuration Register (EVTG3_AOI1_BFT23)	16	RW	0000h
1Dh	Control/Status Register (EVTG3_CTRL)	16	RW	0000h
1Eh	AOI0 Output Filter Register (EVTG3_AOI0_FILT)	16	RW	0000h
1Fh	AOI1 Output Filter Register (EVTG3_AOI1_FILT)	16	RW	0000h

14.3.1.2 AOI0 Boolean Function Term 0 and 1 Configuration Register (EVTG0_AOI0_BFT01 - EVTG3_AOI0_BFT01)

14.3.1.2.1 Offset

Register	Offset
EVTG0_AOI0_BFT01	0h
EVTG1_AOI0_BFT01	8h
EVTG2_AOI0_BFT01	10h
EVTG3_AOI0_BFT01	18h

14.3.1.2.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PT0_AC		PT0_BC		PT0_CC		PT0_DC		PT1_AC		PT1_BC		PT1_CC		PT1_DC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

14.3.1.2.3 Fields

Field	Function
15-14 PT0_AC	Product term 0, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 0. 00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
13-12 PT0_BC	Product term 0, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 0. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
11-10 PT0_CC	Product term 0, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 0. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
9-8 PT0_DC	Product term 0, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 0. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one
7-6 PT1_AC	Product term 1, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 1. 00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
5-4 PT1_BC	Product term 1, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 1. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
3-2 PT1_CC	Product term 1, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 1. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
1-0	Product term 1, D input configuration

Field	Function
PT1_DC	This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 1. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one

14.3.1.3 AOI0 Boolean Function Term 2 and 3 Configuration Register (EVTG0_AOI0_BFT23 - EVTG3_AOI0_BFT23)

14.3.1.3.1 Offset

Register	Offset
EVTG0_AOI0_BFT23	1h
EVTG1_AOI0_BFT23	9h
EVTG2_AOI0_BFT23	11h
EVTG3_AOI0_BFT23	19h

14.3.1.3.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PT2_AC		PT2_BC		PT2_CC		PT2_DC		PT3_AC		PT3_BC		PT3_CC		PT3_DC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

14.3.1.3.3 Fields

Field	Function
15-14 PT2_AC	Product term 2, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 2. 00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
13-12 PT2_BC	Product term 2, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 2. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term

Table continues on the next page...

Field	Function
	10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
11-10 PT2_CC	Product term 2, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 2. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
9-8 PT2_DC	Product term 2, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 2. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one
7-6 PT3_AC	Product term 3, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 3. 00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
5-4 PT3_BC	Product term 3, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 3. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
3-2 PT3_CC	Product term 3, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 3. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
1-0 PT3_DC	Product term 3, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 3. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one

14.3.1.4 AOI1 Boolean Function Term 0 and 1 Configuration Register (EVTG0_AOI1_BFT01 - EVTG3_AOI1_BFT01)

14.3.1.4.1 Offset

Register	Offset
EVTG0_AOI1_BFT01	2h
EVTG1_AOI1_BFT01	Ah
EVTG2_AOI1_BFT01	12h
EVTG3_AOI1_BFT01	1Ah

14.3.1.4.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PT0_AC		PT0_BC		PT0_CC		PT0_DC		PT1_AC		PT1_BC		PT1_CC		PT1_DC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

14.3.1.4.3 Fields

Field	Function
15-14 PT0_AC	Product term 0, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 0. 00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
13-12 PT0_BC	Product term 0, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 0. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
11-10 PT0_CC	Product term 0, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 0. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
9-8 PT0_DC	Product term 0, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 0. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one

Table continues on the next page...

Field	Function
7-6 PT1_AC	Product term 1, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 1. 00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
5-4 PT1_BC	Product term 1, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 1. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
3-2 PT1_CC	Product term 1, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 1. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
1-0 PT1_DC	Product term 1, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 1. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one

14.3.1.5 AOI1 Boolean Function Term 2 and 3 Configuration Register (EVTG0_AOI1_BFT23 - EVTG3_AOI1_BFT23)

14.3.1.5.1 Offset

Register	Offset
EVTG0_AOI1_BFT23	3h
EVTG1_AOI1_BFT23	Bh
EVTG2_AOI1_BFT23	13h
EVTG3_AOI1_BFT23	1Bh

14.3.1.5.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PT2_AC		PT2_BC		PT2_CC		PT2_DC		PT3_AC		PT3_BC		PT3_CC		PT3_DC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

14.3.1.5.3 Fields

Field	Function
15-14 PT2_AC	Product term 2, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 2. 00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
13-12 PT2_BC	Product term 2, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 2. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
11-10 PT2_CC	Product term 2, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 2. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
9-8 PT2_DC	Product term 2, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 2. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one
7-6 PT3_AC	Product term 3, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 3. 00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
5-4 PT3_BC	Product term 3, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 3. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term

Table continues on the next page...

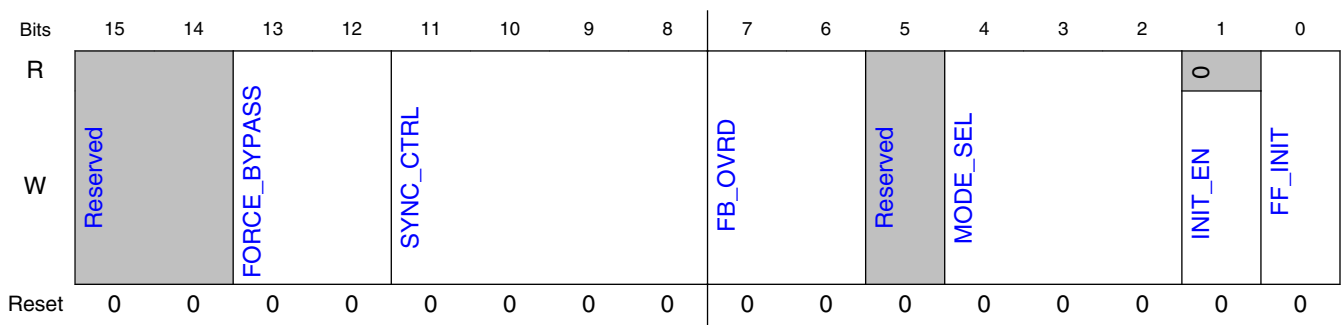
Field	Function
	11b - Force the B input in this product term to a logical one
3-2 PT3_CC	Product term 3, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 3. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
1-0 PT3_DC	Product term 3, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 3. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one

14.3.1.6 Control/Status Register (EVTG0_CTRL - EVTG3_CTRL)

14.3.1.6.1 Offset

Register	Offset
EVTG0_CTRL	5h
EVTG1_CTRL	Dh
EVTG2_CTRL	15h
EVTG3_CTRL	1Dh

14.3.1.6.2 Diagram



14.3.1.6.3 Fields

Field	Function
15-14 —	Reserved
13-12 FORCE_BYPASS	Force Bypass Control When MODE_SEL is set JK-FF mode, we should not enable FORCE_BYPASS, becasue the FB_OVRD will affect output 0xb - Will not force the bypass 1xb - Whatever "MODE_SEL" is, will force bypass Flip-Flop and route the AOI_1(Filter_1) value directly to EVTG_OUTB x0b - Will not force the bypass x1b - Whatever "MODE_SEL" is, will force bypass Flip-Flop and route the AOI_0(Filter_0) value directly to EVTG_OUTA
11-8 SYNC_CTRL	Four EVTG inputs synchronous with bus clk 0xxxb - EVTG input "Dn" will not be synced. 1xxxb - EVTG input "Dn" will be synced by two bus clk cycles. x0xxb - EVTG input "Cn" will not be synced. x1xxb - EVTG input "Cn" will be synced by two bus clk cycles. xx0xb - EVTG input "Bn" will not be synced. xx1xb - EVTG input "Bn" will be synced by two bus clk cycles. xxx0b - EVTG input "An" will not be synced. xxx1b - EVTG input "An" will be synced by two bus clk cycles.
7-6 FB_OVRD	EVTG output feedback override control When "MODE_SEL" is configured as JK-FF mode, need EVTG_OUTA feedback to EVTG input and replace one of the four inputs 00b - replace An 01b - replace Bn 10b - replace Cn 11b - replace Dn
5 —	Reserved
4-2 MODE_SEL	Flip-Flop mode configure 000b - Bypass mode(Default) 001b - RS trigger mode 010b - T-FF mode 011b - D-FF mode 100b - JK-FF mode 101b - Latch mode 110b - Reserved 111b - Reserved
1 INIT_EN	Flip-flop initial output enable control Force the value of FF_INIT to be presented on flip-flop positive output, write "1" to this bit will generate an enable pulse. Read this bit will return always "0". this bit should be set after FF_INIT is set 0b - Write 0 doesn't generate enable pulse 1b - Write 1 will generate enable pulse
0 FF_INIT	Configure flip-flop initial value FF_INIT does not take effect until "1" is written into INIT_EN this bit should be set before INIT_EN set

Field	Function
	0b - Configure the positive output of flip-flop as "0" 1b - Configure the positive output of flip-flop as "1"

14.3.1.7 AOI0 Output Filter Register (EVTG0_AOI0_FILT - EVTG3_AOI0_FILT)

14.3.1.7.1 Offset

Register	Offset
EVTG0_AOI0_FILT	6h
EVTG1_AOI0_FILT	Eh
EVTG2_AOI0_FILT	16h
EVTG3_AOI0_FILT	1Eh

14.3.1.7.2 Diagram

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

14.3.1.7.3 Fields

Field	Function
15-11 —	Reserved
10-8 FILT_CNT	Output Filter Sample Count These bits represent the number of consecutive samples that must agree prior to the output filter accepting an input transition. A value of 0x0 represents 3 samples. A value of 0x7 represents 10 samples. The value of FILT_CNT affects the input latency.
7-0 FILT_PER	Output Filter Sample Period These bits represent the sampling period (in IP bus clock cycles) of the input signals. Each input is sampled multiple times at the rate specified by this field. If FILT_PER is 0x00 (default), then the output filter is bypassed. The value of FILT_PER affects the input latency. When changing values for FILT_PER from one non-zero value to another non-zero value, write a value of zero first to clear the filter.

14.3.1.8 AOI1 Output Filter Register (EVTG0_AOI1_FILT - EVTG3_AOI1_FILT)

14.3.1.8.1 Offset

Register	Offset
EVTG0_AOI1_FILT	7h
EVTG1_AOI1_FILT	Fh
EVTG2_AOI1_FILT	17h
EVTG3_AOI1_FILT	1Fh

14.3.1.8.2 Diagram

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

14.3.1.8.3 Fields

Field	Function
15-11 —	Reserved
10-8 FILT_CNT	Output Filter Sample Count These bits represent the number of consecutive samples that must agree prior to the output filter accepting an output transition. A value of 0x0 represents 3 samples. A value of 0x7 represents 10 samples. The value of FILT_CNT affects the input latency.
7-0 FILT_PER	Output Filter Sample Period These bits represent the sampling period (in IP bus clock cycles) of the output signals. Each output is sampled multiple times at the rate specified by this field. If FILT_PER is 0x00 (default), then the output filter is bypassed. The value of FILT_PER affects the output latency. When changing values for FILT_PER from one non-zero value to another non-zero value, write a value of zero first to clear the filter.

14.4 Functional description

The following sections describe functional details of the EVTG module.

14.4.1 Configuration Examples for AOI Combinational function

This section presents examples of the programming model configuration for simple boolean expressions.

The AOI module provides a universal boolean function generator using a four-term sum of products expression with each product term containing true or complement values of the four selected event inputs (A, B, C, D). Specifically, the EVTG output is defined by the following “4 x 4” boolean expression:

EVTGn_AOIm (m=A,B)

= (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 0

| (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 1

| (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 2

| (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 3

where each selected input term in each product term can be configured to produce a logical 0 or 1 or pass the true or complement of the selected event input. Each product term uses eight bits of configuration information, two bits for each of the four selected event inputs. The actual boolean expression implemented in each channel is:

EVTGn_AOIm (m=A,B)

= (PT0_AC[0] & A | PT0_AC[1] & ~A)// product term 0

& (PT0_BC[0] & B | PTO_BC[1] & ~B)

& (PT0_CC[0] & C | PTO_CC[1] & ~C)

& (PT0_DC[0] & D | PTO_DC[1] & ~D)

| (PT1_AC[0] & A | PT1_AC[1] & ~A)// product term 1

& (PT1_BC[0] & B | PT1_BC[1] & ~B)

& (PT1_CC[0] & C | PT1_CC[1] & ~C)

& (PT1_DC[0] & D | PT1_DC[1] & ~D)

| (PT2_AC[0] & A | PT2_AC[1] & ~A)// product term 2

& (PT2_BC[0] & B | PT2_BC[1] & ~B)

& (PT2_CC[0] & C | PT2_CC[1] & ~C)

& (PT2_DC[0] & D | PT2_DC[1] & ~D)

| (PT3_AC[0] & A | PT3_AC[1] & ~A)// product term 3

& (PT3_BC[0] & B | PT3_BC[1] & ~B)

& (PT3_CC[0] & C | PT3_CC[1] & ~C)

& (PT3_DC[0] & D | PT3_DC[1] & ~D)

Consider the settings of the 32-bit registers for several simple boolean expressions as shown in below Table.

Table 14-1. EVTGn_AOIm_BFT(m=0,1) Values for Simple Boolean Expressions

Event Output Expression	PT0	PT1	PT2	PT3	{EVTGn_AOIm_BFT01, EVTGn_AOIm_BFT23}
A&B	A&B	0	0	0	01011111_00000000_00000000_00000000
A&B&C	A&B&C	0	0	0	01010111_00000000_00000000_00000000
(A&B&C) | D	A&B&C	D	0	0	01010111_11111101_00000000_00000000
A|B|C|D	A	B	C	D	01111111_11011111_11110111_11111101
(A & ~B) | (~A & B)	A & ~B	~A & B	0	0	01101111_10011111_00000000_00000000

As can be seen in these examples, the resulting logic provides a simple yet powerful boolean function evaluation for defining an AOI output.

14.4.2 Input Sync and Filter Logic Description

EVTG support sync to evtg inputs and filter function to AOI output.

Both features are default disabled. We can enable input sync function by configuring EVTGn_CTRL[SYNC_CTRL], or enable AOI filter function by configuring EVTGn_AOI0/1_FILT[FILT_PER] a non-zero value.

The application scenario for both are different. Input sync feature makes evtg input sync for two bus_clk cycles. Any glitch whose width is less than a bus clk period will be removed. But for Filter, though this module works in bus_clk and have the same function as the first one, it's designed for filter. Since the Filter_Delay is "(FILT_CNT + 3) x FILT_PER + 2". Any signal whose width is less than this value will be filtered.

Therefore, we recommend enabling just one of both, and which one should be enabled depends on the actual user case. Whether some signals with expected width bigger than bus_clk still need be removed by FILTER.

Of course, it's also acceptable to enable both functions. For some user cases in bypass mode and RS mode, both functions have to be disabled. Except for this scenario, it's recommended that at least one of input sync and filter has to be enabled.

14.4.3 Flip-Flop mode configuration

EVTG module inside implements different kinds of Flip-Flop for the generation of desired EVTG output. The Flip-Flop can be configured as Bypass mode, RS trigger mode, T-FF mode, D-FF mode, JK-FF mode, Latch mode.

14.4.3.1 Bypass Mode

When register bits EVTGn_CTRL[MODE_SEL] are configured as 0x0, Bypass mode will be selected. So default is Bypass mode.

In this mode, filp-flop will be passed, The two AOI expressions "AOI_0" and "AOI_1" will be directly assigned to EVTG outputs(EVTG_OUTA and EVTG_OUTB) .

In this mode, user can choose to enable or disable input sync logic and filter function. Following diagram shows the disabled case(input sync and filter are removed from the diagram).

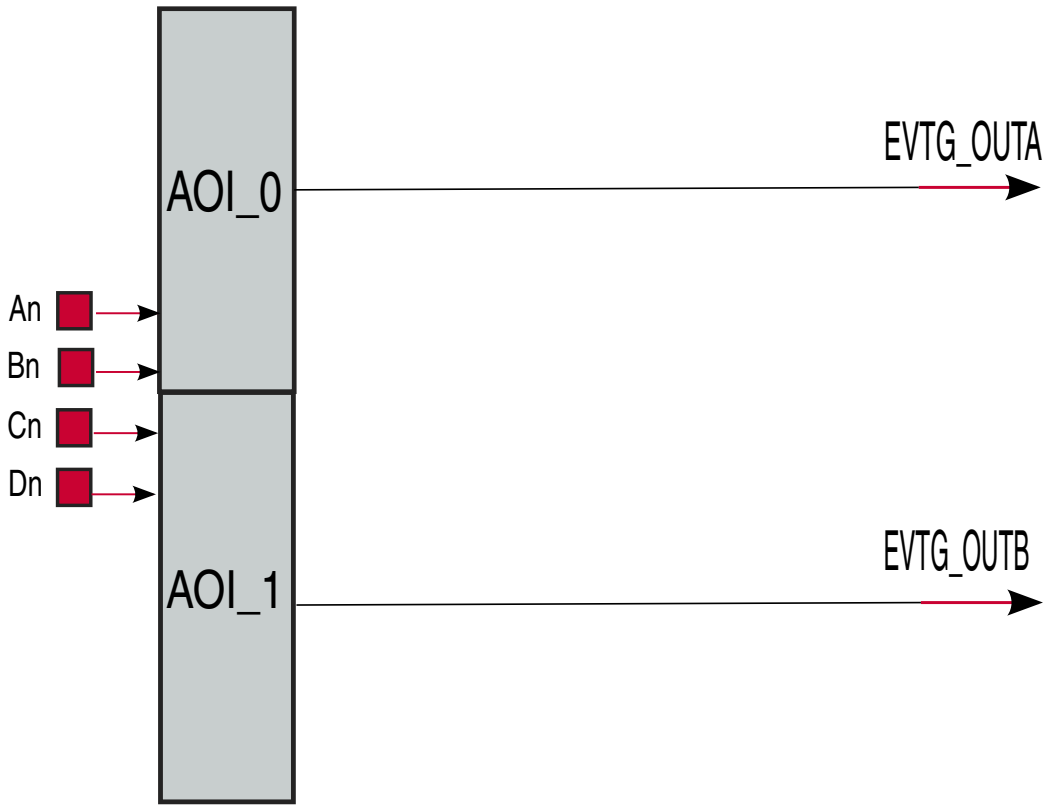


Figure 14-2. Bypass mode diagram

14.4.3.2 RS Trigger Mode

When register bits EVTGn_CTRL[MODE_SEL] are configured as 0x1, RS trigger mode will be selected.

In this mode, AOI_0 expression is Reset port, and AOI_1 is Set port. Both are active high. When "R"(Reset) is high, whatever "S"(Set) is, EVTG_OUTA will be "0". When "R" is low and "S" is high, EVTG_OUTA will be "1". If both "R" and "S" are low, EVTG output will be kept. See the following figure.

EVTG_OUTB is always the complement of EVTG_OUTA.

In this mode, user can choose to enable or disable input sync logic and filter function. Below diagram shows the disabled case(input sync and filter are removed from the following diagram).

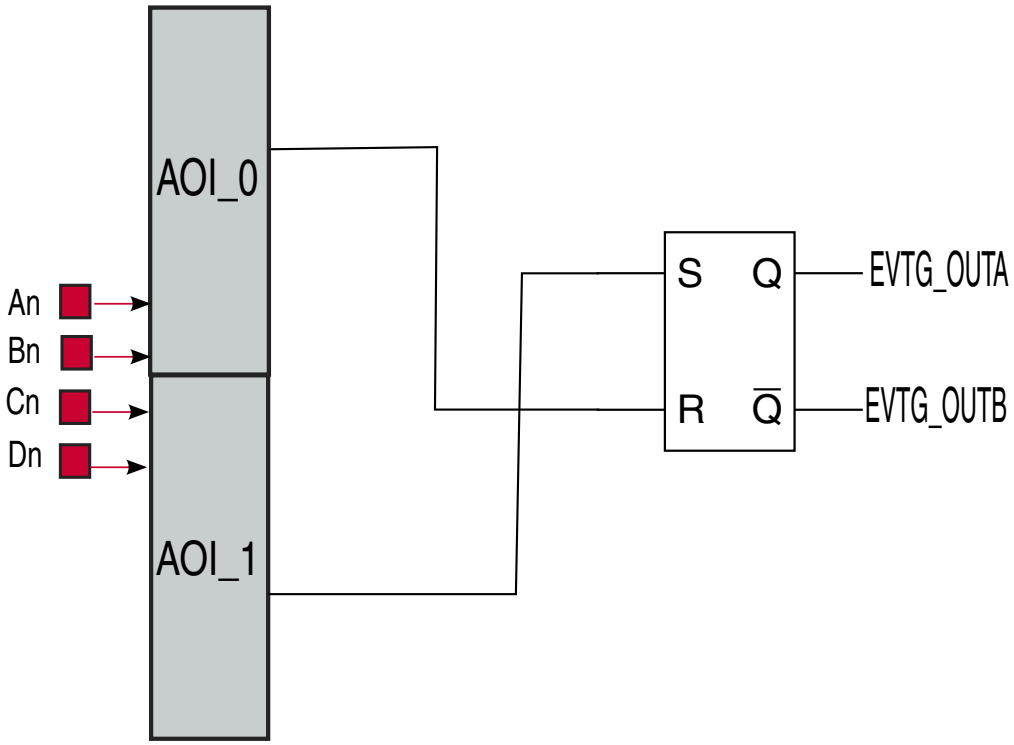


Figure 14-3. RS trigger mode diagram

14.4.3.3 T-FF Mode

When register bits EVTGn_CTRL[MODE_SEL] are configured as 0x2, T-FF mode will be selected.

In this mode, AOI_0 expression is T port of T-FF, AOI_1 is CLK port. When T assert, the Q port (EVTG_OUTA) will turnover at the rising edge of "CLK". When T dis-assert, Q(EVTG_OUTA) will be kept. See the following figure.

EVTG_OUTB is always the complement of EVTG_OUTA.

In this mode, input sync or filter has to be enabled to remove the possible glitch.

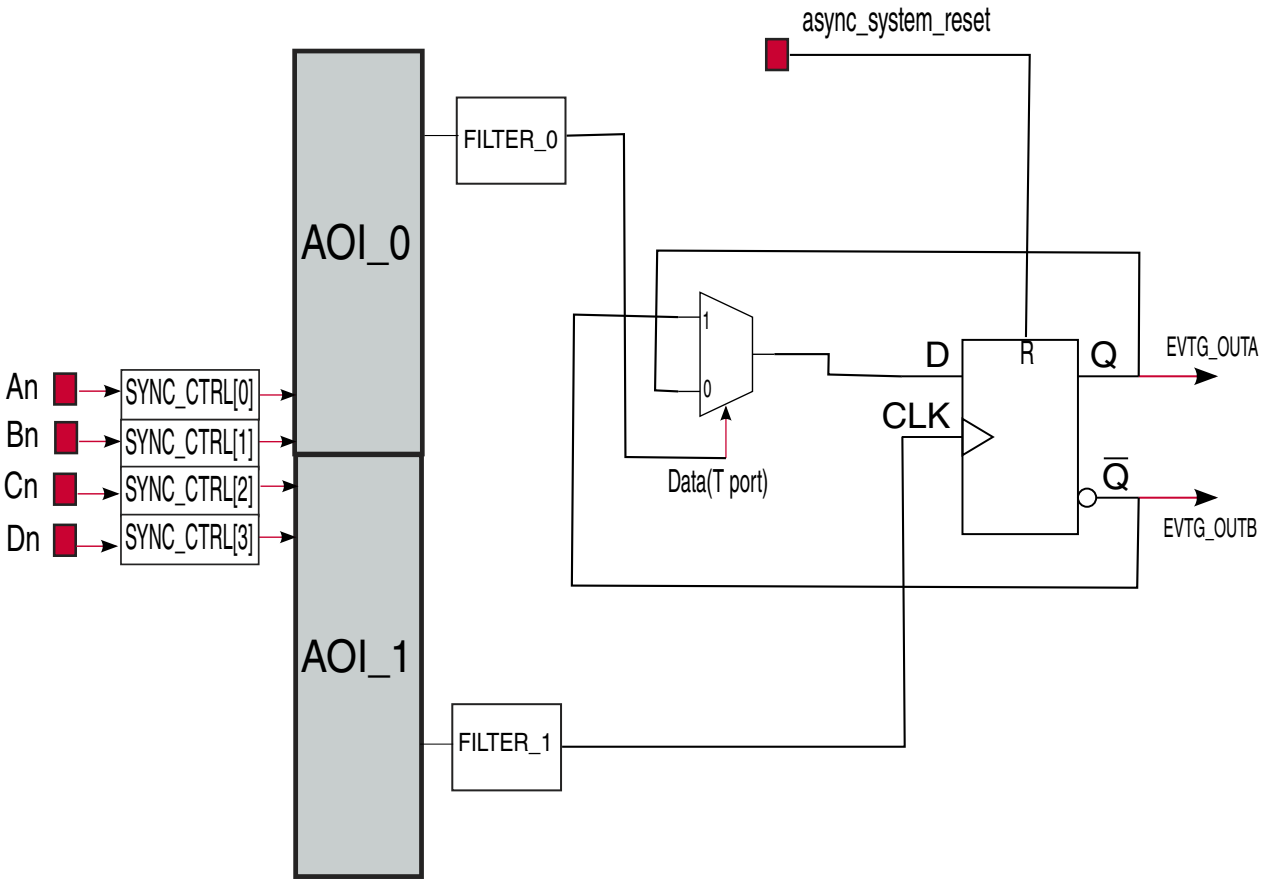


Figure 14-4. T-FF mode diagram

14.4.3.4 D-FF Mode

When register bits EVTGn_CTRL[MODE_SEL] are configured as 0x3, D-FF mode will be selected.

In this mode, AOI_0 expression is D port of D-FF, AOI_1 expression is CLK port. At the rising edge of "CLK", D will be captured to Q (EVTG_OUTA). See the following figure.

EVTG_OUTB is always the complement of EVTG_OUTA.

In this mode, input sync or filter has to be enabled to remove the possible glitch.

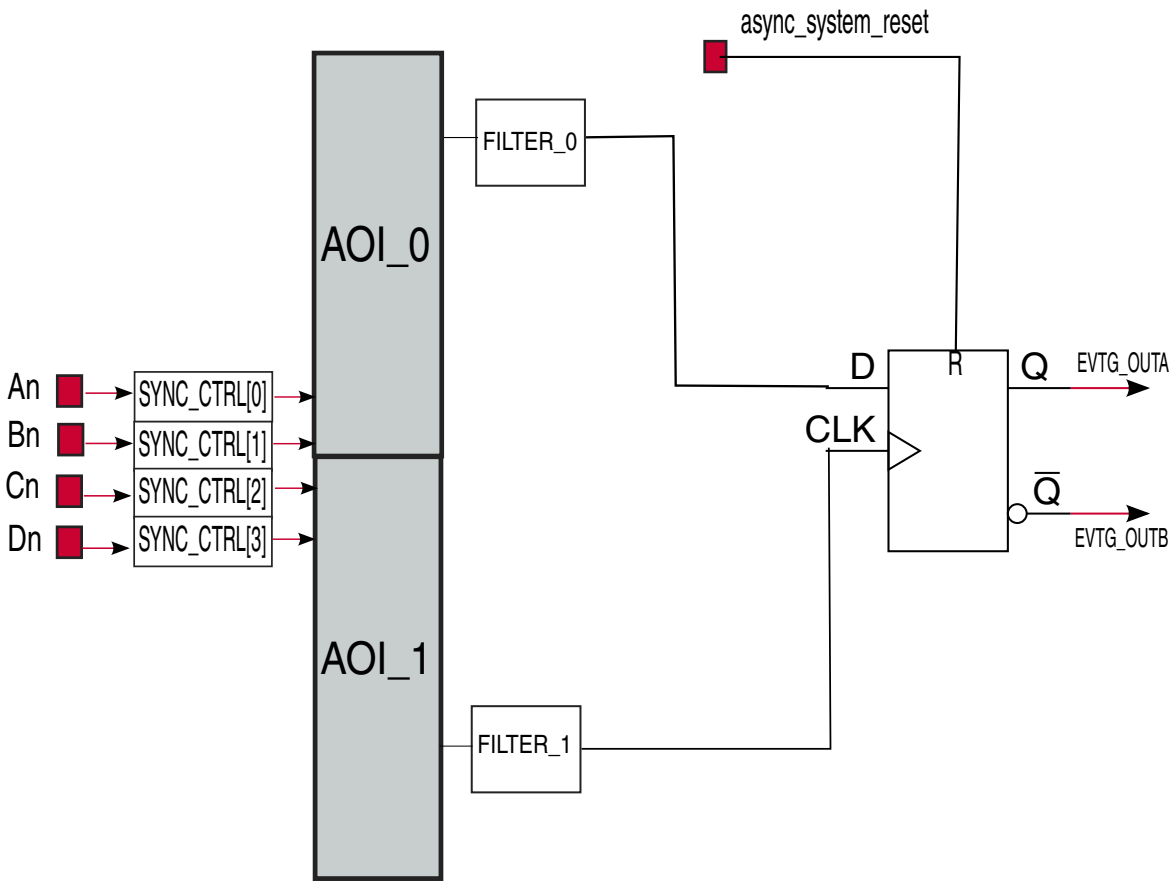


Figure 14-5. D-FF mode diagram

14.4.3.5 JK-FF Mode

In general, JK Flip-Flop have four input ports: J, K , Q and CLK(Q is output of Flip-Flop) . And the logical expression is J&~Q | ~K&Q;

When register bits EVTGn_CTRL[MODE_SEL] are configured as 0x4, JK-FF mode will be selected.

Here we implement the logic expression by AOI so that we can reuse the D-FF to implement JK-FF. Suppose we set EVTG input "An" as "J" port, "Cn" as "K" port, "Dn" as "CLK" port, and "Q" port of FF feed back and override "Bn", as shown in the following figure. According to the JK logic expression, the AOI_0 expression will be "An&~Bn | Bn&~Cn", AOI_1 expression will be "Dn". So the corresponding register configure should be:

EVTGn_AOI0_BFT01 = 01101111_11011011;

EVTGn_AOI0_BFT23 = 00000000_00000000;

EVTGn_AOI1_BFT01 = 11111101_00000000;

EVTGn_AOI1_BFT23 = 00000000_00000000;

EVTGn_CTRL = 00001111_01010000; (EVTGn_CTRL[FB_OVRD] = 01)

EVTGn_CTRL[FB_OVRD] represents which EVTG input is replaced by FF output(EVTG_OUTA), in this case, Bn is replaced, this is why we set EVTGn_CTRL[FB_OVRD] as "01".

In this mode, input sync or filter has to be enabled to remove the possible glitch.

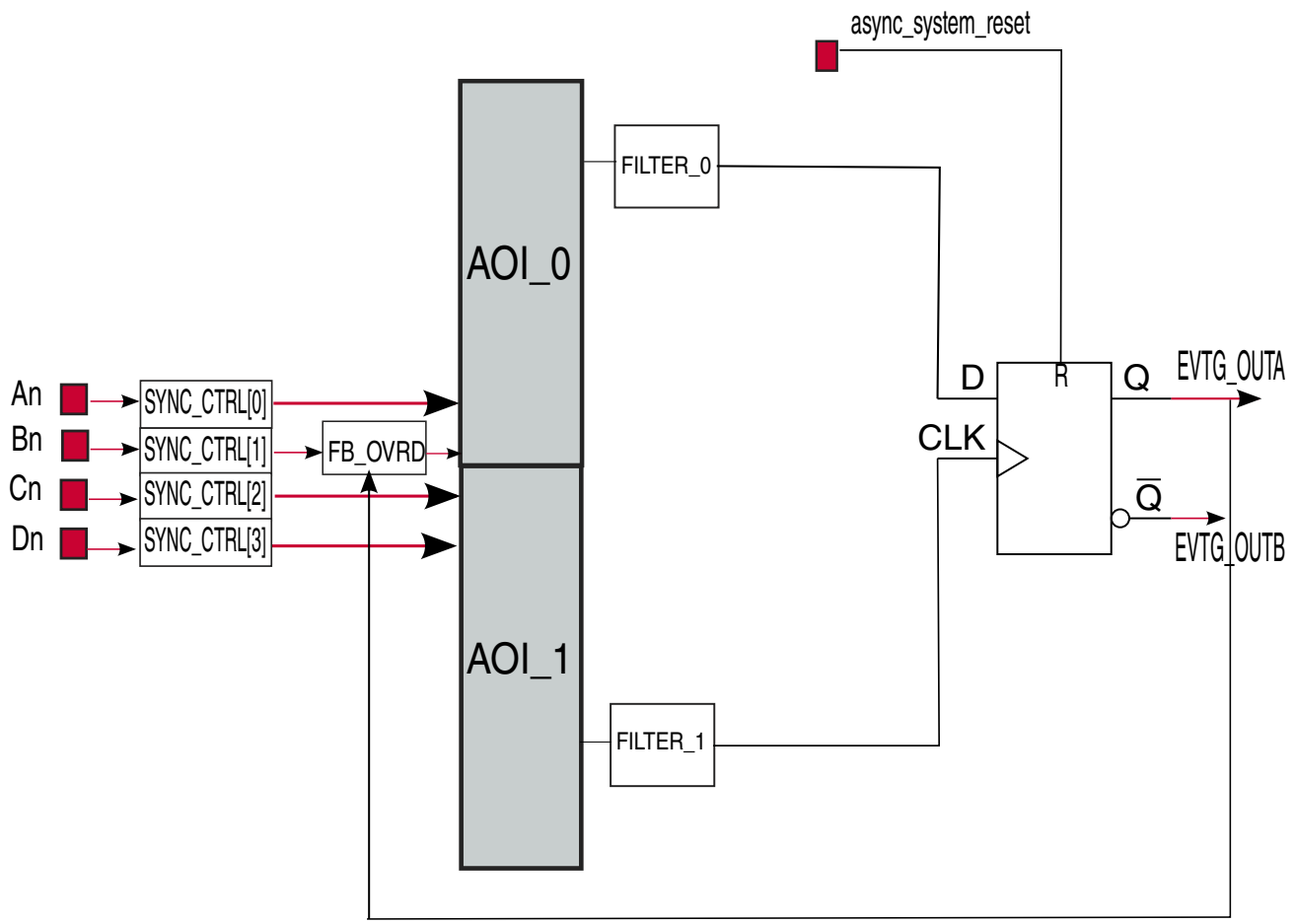


Figure 14-6. JK-FF mode diagram

14.4.3.6 Latch Mode

When register bits EVTGn_CTRL[MODE_SEL] are configured as 0x5, Latch mode will be selected.

In this mode, AOI_0 expression is D port, AOI_1 is CLK port. Different from D-FF mode, in Latch mode, D port will be passed only when CLK is high, and output will be kept when CLK is low.

EVTG_OUTB is always the complement of EVTG_OUTA.

In this mode, input sync or filter has to be enabled to remove the possible glitch.

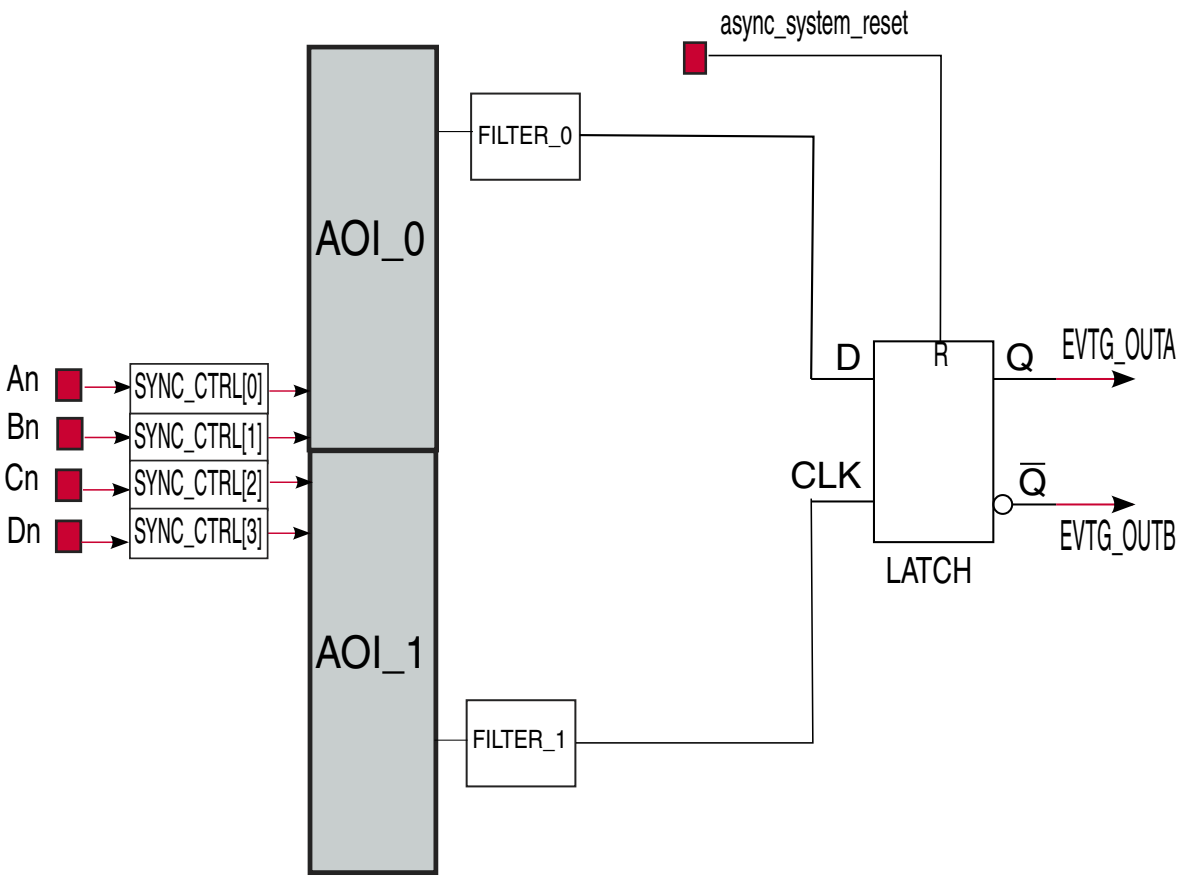


Figure 14-7. Latch mode diagram

14.4.4 EVTG Timing Between Inputs and Outputs

- For BYPASS mode and RS mode with input sync and Filter disabled, there will be no any delay between inputs and outputs.
- For the modes with input sync and Filter function enabled, the delay between input and output will be:

Sync_Delay : 2

Flip-Flop_Delay : 1

Filter_Delay : (FILT_CNT + 3) x FILT_PER + 2

Whole_Delay = ((FILT_CNT + 3) x FILT_PER + 2) + 3

In this case, all signals will be synchronous in bus clk

14.5 Application information

This section describes applications supported by the EVTG module.

Chapter 15
Inter-Peripheral Crossbar Switch (XBAR)

15.1 Chip-specific information for this module

NOTE

XBAR functions as path connection. When initializing or reconfiguring XBAR, it could generate glitch during the input source switching.

15.1.1 Number of inputs and outputs

The dedicated XBAR chapters refer to each instance's number of inputs as NUM_IN or N, and number of outputs as NUM_OUT or M. The following table identifies the values.

Table 15-1. XBAR inputs and outputs

XBAR instance	Number of inputs (NUM_IN or N)	Number of outputs (NUM_OUT or M)
XBARA	54	66

The XBAR inter-connection is shown in the following figure.

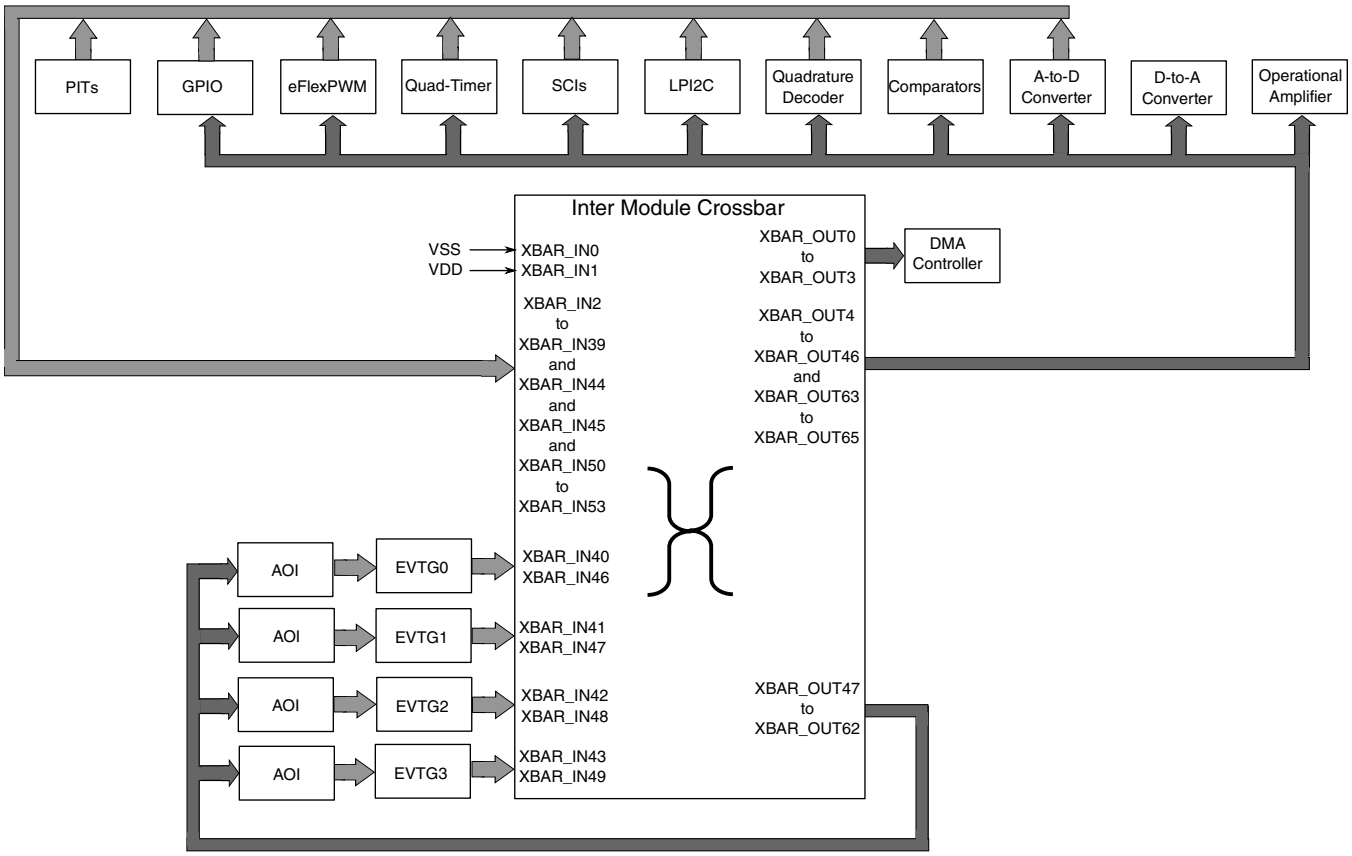


Figure 15-1. XBAR inter-connection

15.1.2 XBARA Inputs

The following table shows the signals that can be inputs to the XBAR.

Table 15-2. XBARA Inputs

From Signal	Signal Description	XBARA Input
VSS	VSS	XBAR_IN0
VDD	VDD	XBAR_IN1
XB_IN2	Package Pin	XBAR_IN2
XB_IN3	Package Pin	XBAR_IN3
XB_IN4	Package Pin	XBAR_IN4
XB_IN5	Package Pin	XBAR_IN5
XB_IN6	Package Pin	XBAR_IN6
XB_IN7	Package Pin	XBAR_IN7
XB_IN8	Package Pin	XBAR_IN8
XB_IN9	Package Pin	XBAR_IN9
LPI2C0_M_TRIG / LPI2C0_S_TRIG	LPI2C0 Master or Slave Output Trigger	XBAR_IN10
$\overline{\text{EWM\_out}}$	EWM reset output	XBAR_IN11

Table continues on the next page...

Table 15-2. XBARA Inputs (continued)

From Signal	Signal Description	XBARA Input
CMPA_O	Comparator A Output	XBAR_IN12
CMPB_O	Comparator B Output	XBAR_IN13
CMPC_O	Comparator C Output	XBAR_IN14
CMPD_O	Comparator D Output	XBAR_IN15
AN8_LIMIT	ADC conversion result 8 high/low limit [1]	XBAR_IN16
AN9_LIMIT	ADC conversion result 9 high/low limit [1]	XBAR_IN17
AN10_LIMIT	ADC conversion result 10 high/low limit [1]	XBAR_IN18
AN11_LIMIT	ADC conversion result 11 high/low limit [1]	XBAR_IN19
PWMA0_A	PWM SM0 output A	XBAR_IN20
PWMA0_B	PWM SM0 output B	XBAR_IN21
PWMA1_A	PWM SM1 output A	XBAR_IN22
PWMA1_B	PWM SM1 output B	XBAR_IN23
PWMA2_A	PWM SM2 output A	XBAR_IN24
PWMA2_B	PWM SM2 output B	XBAR_IN25
PWMA3_A	PWM SM3 output A	XBAR_IN26
PWMA3_B	PWM SM3 output B	XBAR_IN27
PWMA0_OUT_TRIG0	PWM SM0 output trigger0	XBAR_IN28
PWMA0_OUT_TRIG1	PWM SM0 output trigger1	XBAR_IN29
PWMA1_OUT_TRIG0	PWM SM1 output trigger0	XBAR_IN30
PWMA1_OUT_TRIG1	PWM SM1 output trigger1	XBAR_IN31
PWMA2_OUT_TRIG0	PWM SM2 output trigger0	XBAR_IN32
PWMA2_OUT_TRIG1	PWM SM2 output trigger1	XBAR_IN33
PWMA3_OUT_TRIG0	PWM SM3 output trigger0	XBAR_IN34
PWMA3_OUT_TRIG1	PWM SM3 output trigger1	XBAR_IN35
AN0_LIMIT/TA0_OUT [2]	ADC conversion result 0 high/low limit [1] or Quad-Timer A0 output	XBAR_IN36
AN1_LIMIT/TA1_OUT [2]	ADC conversion result 1 high/low limit [1] or Quad-Timer A1 output	XBAR_IN37
AN2_LIMIT/TA2_OUT [2]	ADC conversion result 2 high/low limit [1] or Quad-Timer A2 output	XBAR_IN38
AN3_LIMIT/TA3_OUT [2]	ADC conversion result 3 high/low limit [1] or Quad-Timer A3 output	XBAR_IN39
EVTG0_OUTA	EVTG0 output A	XBAR_IN40
EVTG1_OUTA	EVTG1 output A	XBAR_IN41
EVTG2_OUTA	EVTG2 output A	XBAR_IN42
EVTG3_OUTA	EVTG3 output A	XBAR_IN43
PIT0_SYNC_OUT	PIT0 Sync Output	XBAR_IN44
PIT1_SYNC_OUT	PIT1 Sync Output	XBAR_IN45
EVTG0_OUTB	EVTG0 output B	XBAR_IN46
EVTG1_OUTB	EVTG1 output B	XBAR_IN47
EVTG2_OUTB	EVTG2 output B	XBAR_IN48

Table continues on the next page...

Table 15-2. XBARA Inputs (continued)

From Signal	Signal Description	XBARA Input
EVTG3_OUTB	EVTG3 output B	XBAR_IN49
Reserved	-	XBAR_IN50
PWMA_ALL_TRIG	All PWMA[n]_out_trig[x] OR'ed[3]	XBAR_IN51
SCI0_RX_FULL	SCI0 Receive Data register full flag	XBAR_IN52
LPI2C1_M_TRIG / LPI2C1_S_TRIG	LPI2C1 Master or Slave Output Trigger	XBAR_IN53

1. The signal state is decided by ADC conversion result in the result register RSLT[n].
   - If the ADC conversion result in RSLT[n] is greater than the value programmed into the High Limit register HILIM[n], this signal clears.
   - If the ADC conversion result in RSLT[n] is less than the value programmed into the Low Limit register LOLIM[n], this signal sets.
   - No change if the ADC conversion result in RSLT[n] is between the high and the low limits.
2. The SIM_ADC_TMR_SEL register selects AN[n]_LIMIT or TA[n].
3. PWMA0_out_trig0 | PWMA0_out_trig1 | PWMA1_out_trig0 | PWMA1_out_trig1 | PWMA2_out_trig0 | PWMA2_out_trig1 | PWMA3_out_trig0 | PWMA3_out_trig1

15.1.3 XBARA Outputs

Table 15-3. XBARA Outputs

XBARA Output	To Signal	Signal Description
XBAR_OUT0	DMA_REQ0	XBAR DMA Request 0
XBAR_OUT1	DMA_REQ1	XBAR DMA Request 1
XBAR_OUT2	DMA_REQ2	XBAR DMA Request 2
XBAR_OUT3	DMA_REQ3	XBAR DMA Request 3
XBAR_OUT4	XB_OUT4	Package Pin
XBAR_OUT5	XB_OUT5	Package Pin
XBAR_OUT6	XB_OUT6	Package Pin
XBAR_OUT7	XB_OUT7	Package Pin
XBAR_OUT8	XB_OUT8	Package Pin
XBAR_OUT9	XB_OUT9	Package Pin
XBAR_OUT10	XB_OUT10	Package Pin
XBAR_OUT11	XB_OUT11	Package Pin
XBAR_OUT12	ADC_SYNC0	ADC SYNC0 input
XBAR_OUT13	ADC_SYNC1	ADC SYNC1 input
XBAR_OUT14	OPAMPA (CFGSELB1 & CFGSELC1) and OPAMPB (CFGSELB1 & CFGSELC1)	OPAMPA and OPAMPB
XBAR_OUT15	DACA_12B_SYNC	12bit DAC SYNC_IN
XBAR_OUT16	CMPA	Comparator A Window/Sample
XBAR_OUT17	CMPB	Comparator B Window/Sample
XBAR_OUT18	CMPC	Comparator C Window/Sample

Table continues on the next page...

Table 15-3. XBARA Outputs (continued)

XBARA Output	To Signal	Signal Description
XBAR_OUT19	CMPD	Comparator D Window/Sample
XBAR_OUT20	PWMA0_EXTA	Alternate PWM control signal on PWMA0_EXTA input
XBAR_OUT21	PWMA1_EXTA	Alternate PWM control signal on PWMA1_ EXTA input
XBAR_OUT22	PWMA2_EXTA	Alternate PWM control signal on PWMA2_EXTA input
XBAR_OUT23	PWMA3_EXTA	Alternate PWM control signal on PWMA3_EXTA input
XBAR_OUT24	PWMA0_EXT_SYNC	PWMA SM0 external synchronization signal
XBAR_OUT25	PWMA1_EXT_SYNC	PWMA SM1 external synchronization signal
XBAR_OUT26	PWMA2_EXT_SYNC	PWMA SM2 external synchronization signal
XBAR_OUT27	PWMA3_EXT_SYNC	PWMA SM3 external synchronization signal
XBAR_OUT28	PWMA_EXT_CLK	PWMA External Clock
XBAR_OUT29	PWMA_FAULT0	PWMA Fault0 input
XBAR_OUT30	PWMA_FAULT1	PWMA Fault1 input
XBAR_OUT31	PWMA_FAULT2	PWMA Fault2 input
XBAR_OUT32	PWMA_FAULT3	PWMA Fault3 input
XBAR_OUT33	PWMA_FORCE	PWMA external Force Input
XBAR_OUT34	OPAMPA (CFGSELB0 & CFGSELC0)	OPAMPA
XBAR_OUT35	OPAMPB (CFGSELB0 & CFGSELC0)	OPAMPB
XBAR_OUT36	SCI0_RXD	SCI0 receive data input
XBAR_OUT37	SCI1_RXD	SCI1 receive data input
XBAR_OUT38	TA0_IN	Quad-Timer A Input 0
XBAR_OUT39	TA1_IN	Quad-Timer A Input 1
XBAR_OUT40	TA2_IN	Quad-Timer A Input 2
XBAR_OUT41	TA3_IN	Quad-Timer A Input 3
XBAR_OUT42 to XBAR_OUT46	Reserved	-
XBAR_OUT47	EVTG0_A	EVTG0 Input A
XBAR_OUT48	EVTG0_B	EVTG0 Input B
XBAR_OUT49	EVTG0_C	EVTG0 Input C
XBAR_OUT50	EVTG0_D	EVTG0 Input D
XBAR_OUT51	EVTG1_A	EVTG1 Input A
XBAR_OUT52	EVTG1_B	EVTG1 Input B
XBAR_OUT53	EVTG1_C	EVTG1 Input C
XBAR_OUT54	EVTG1_D	EVTG1 Input D
XBAR_OUT55	EVTG2_A	EVTG2 Input A
XBAR_OUT56	EVTG2_B	EVTG2 Input B

Table continues on the next page...

Table 15-3. XBARA Outputs (continued)

XBARA Output	To Signal	Signal Description
XBAR_OUT57	EVTG2_C	EVTG2 Input C
XBAR_OUT58	EVTG2_D	EVTG2 Input D
XBAR_OUT59	EVTG3_A	EVTG3 Input A
XBAR_OUT60	EVTG3_B	EVTG3 Input B
XBAR_OUT61	EVTG3_C	EVTG3 Input C
XBAR_OUT62	EVTG3_D	EVTG3 Input D
XBAR_OUT63	EWM_in	External Watchdog Monitor
XBAR_OUT64	LPI2C0_IN_TRIG / HREQ	LPI2C0 input trigger / Host Request
XBAR_OUT65	LPI2C1_IN_TRIG / HREQ	LPI2C1 input trigger / Host Request

15.2 Overview

This module implements an array of M N-input combinational muxes. All muxes share the same N inputs in the same order, but each mux has its own independent select field.

The intended application of this module is to provide a flexible crossbar switch function that allows any input (typically from external GPIO or internal module outputs) to be connected to any output (typically to external GPIO or internal module inputs) under user control. This is used to allow user configuration of data paths between internal modules and between internal modules and GPIO.

A subset of the muxes can be configured to support edge detection and either interrupt or DMA request generation based on detected signal edges on the mux output. This allows signal transitions on the signals feeding the crossbar to trigger interrupts or initiate data transfers via DMA into or out of other system modules.

15.2.1 Block Diagram

The block diagram for XBAR is shown in Figure 15-2.

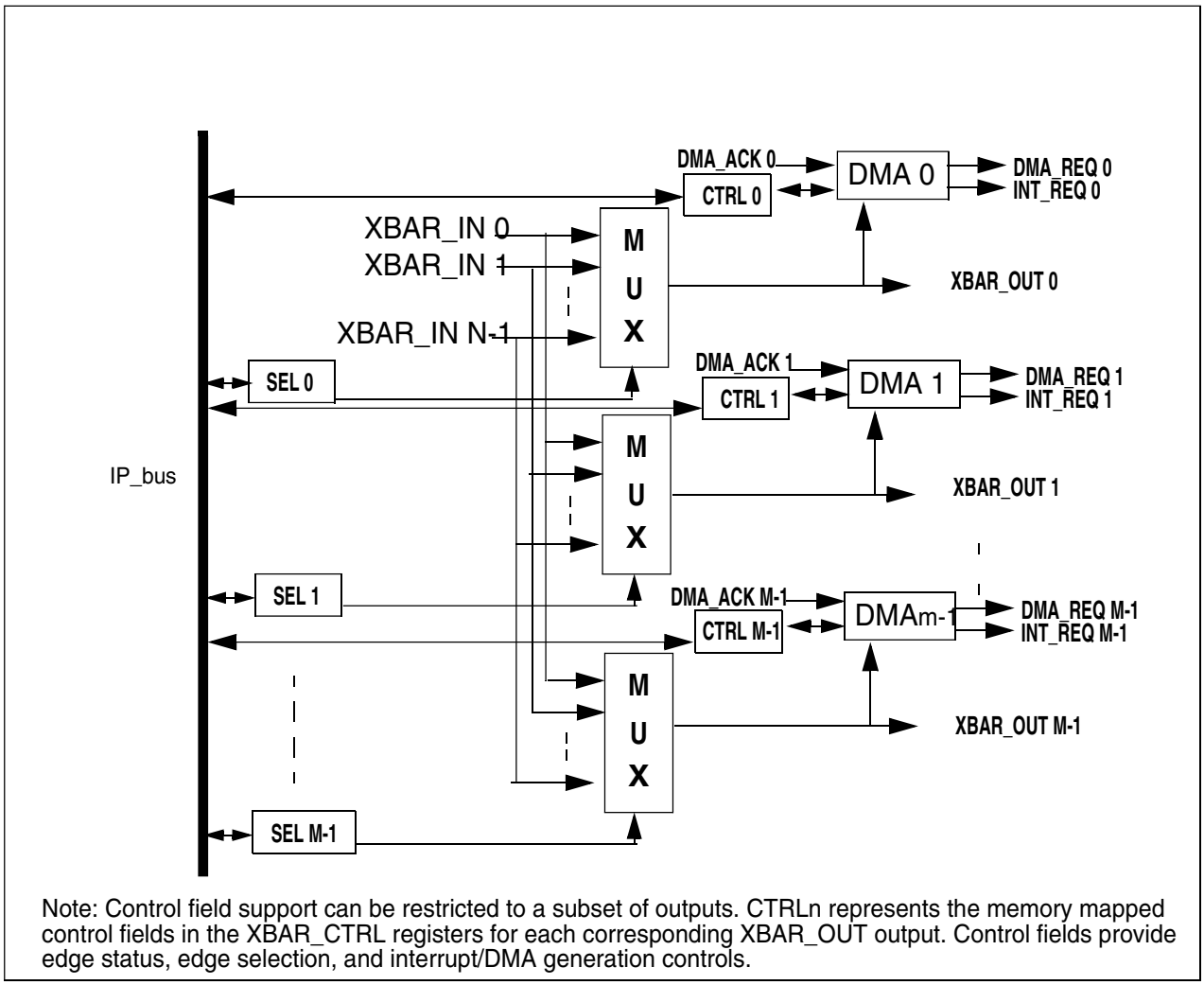


Figure 15-2. XBAR Block Diagram

15.2.2 Features

The XBAR module design includes these distinctive features:

- M identical N-input muxes with individual select fields.
- Edge detection with associated interrupt or DMA request generation for a subset of mux outputs.
- Memory mapped registers with IPBus interface for select and control fields.
- Register write protection input signal.

15.2.3 Modes of Operation

The XBAR module design operates in only a single mode of operation: Functional Mode. The various operation modes are detailed in Functional Mode.

15.3 Functional Description

15.3.1 General

The XBAR module has only one mode of operation, functional mode.

15.3.2 Functional Mode

The value of each mux output is XBAR_OUT[n] = XBAR_IN[SELn]. The SELn select values are configured in the XBAR_SEL registers. All muxes share the same inputs in the same order.

The following is an example to show how to specify the specific input for the specific output:

To select XBAR_IN03 to connect with XBAR_OUT07, according to the index of XBAR_OUT07, select the SEL07 (same number with the selected XBAR_OUT index) fields in the MSBs of the Select Register 3 XBAR_SEL3, and assign the index (0x03, Hexadecimal) of the selected XBAR_IN in it.

```
XBAR_SEL3 &= 0x0011; /*Clear the SEL7 fields
XBAR_SEL3 |= 0x0300; /*Assign the 0x03 to the SEL7 fields
```

A subset of XBAR_OUT[*] outputs has dedicated control fields in a Crossbar Control (XBAR_CTRL) register. Control fields provide the ability to perform edge detection on the corresponding XBAR_OUT output. Edge detection in turn can optionally be used to trigger an interrupt or DMA request. The intention is that, by detecting specified edges on signals propagating through the Crossbar, interrupts or DMA requests can be triggered to perform data transfers to or from other system components.

Control fields include an edge status field (STS), an detected edge type field (EDGE), and interrupt and DMA enable fields (IEN and DEN). STSn is set to 1 when an edge consistent with EDGEn occurs on XBAR_OUT[n]. STSn is cleared by writing 1 to it. Writing 0 as no effect. See Interrupts and DMA Requests for details on the use of STSn for DMA and interrupt request generation.

15.3.3 Clocks

All sequential functionality is controlled by the Bus Clock.

15.3.4 Resets

The XBAR module can be reset by only a hard reset, which forces all registers to their reset state.

15.3.5 Interrupts and DMA Requests

For each XBAR_OUT[*] output with XBAR_CTRL register support, DMA or interrupt functionality can be enabled by setting the corresponding XBAR_CTRL register bit DENn or IENn to 1. DENn and IENn should not be set to 1 at the same time for the same output XBAR_OUT[n].

Setting DENn to 1 enables DMA functionality for XBAR_OUT[n]. When DMA functionality is enabled, the output DMA_REQ[n] reflects the value of STSn. Thus the DMA request asserts when the edge specified by EDGEn is detected on XBAR_OUT[n]. Also, a rising edge on DMA_ACK[n] sets STSn to zero and thus clears the DMA request. When DEN is 0, DMA_REQ[n] is held low and DMA_ACK[n] is ignored.

Setting IENn to 1 enables interrupt functionality for XBAR_OUT[n]. When interrupt functionality is enabled, the output INT_REQ[n] reflects the value of STSn. Thus the interrupt request asserts when the edge specified by EDGEn is detected on XBAR_OUT[n]. The interrupt request is cleared by writing a 1 to STSn. When IENn is 0, INT_REQ[n] is held low.

15.4 External Signals

The following table summarizes the module's external signals.

Table 15-4. Control Signal Properties

Name	I/O Type	Function	Reset State	Notes
XBAR_OUT [0:NUMOUT-1]	O	Mux Outputs with configurable width	XBAR_IN[0]	
XBAR_IN [0:NUMIN-1]	I	Mux Inputs with configurable width	*	
DMA_REQ	O	DMA request	0	
INT_REQ	O	Interrupt request	0	
DMA_ACK	I	DMA acknowledge	0	

At reset, each output XBAR_OUT[*] contains the reset value of the signal driving XBAR_IN[0].

15.4.1 XBAR_OUT[0:NUM_OUT-1] - MUX Outputs

This is a one-dimensional array of the mux outputs. The value on each output XBAR_OUT[n] is determined by the setting of the corresponding memory mapped register SELn such that XBAR_OUT[n] = XBAR_IN[SELn].

15.4.2 XBAR_IN[0:NUM_IN-1] - MUX Inputs

This is a one-dimensional array consisting of the inputs shared by each mux. All muxes share the same inputs in the same order.

15.4.3 DMA_REQ[n] - DMA Request Output(s)

DMA_REQ[n] is a DMA request to the DMA controller.

15.4.4 DMA_ACK[n] - DMA Acknowledge Input(s)

DMA_ACK[n] is a DMA acknowledge input from the DMA controller.

15.4.5 INT_REQ[n] - Interrupt Request Output(s)

INT_REQ[n] is an interrupt request output to the interrupt controller.

15.5 Memory Map and Register Descriptions

The XBAR module has select registers and control registers.

In the XBAR select registers, the SELn fields select which of the shared inputs (XBAR_IN[*]) is muxed to each mux output (XBAR_OUT[*]). There is one SELn field per mux and therefore one per XBAR_OUT output. Crossbar output XBAR_OUT[n]

presents the value of XBAR_IN[SELn]. Each select register contains two SELn fields. In the first select register, the LSBs contain the select field for mux 0, and the MSBs contain the select field for mux 1. The pattern repeats in subsequent select registers.

The actual signals connected to XBAR_IN and XBAR_OUT are application specific and are described in the Chip-specific information.

The XBAR control registers configure edge detection, interrupt, and DMA features for a subset of the XBAR_OUT[*] outputs.

XBARA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E340	Crossbar A Select Register 0 (XBARA_SEL0)	16	R/W	0000h	15.5.1/384
E341	Crossbar A Select Register 1 (XBARA_SEL1)	16	R/W	0000h	15.5.2/384
E342	Crossbar A Select Register 2 (XBARA_SEL2)	16	R/W	0000h	15.5.3/385
E343	Crossbar A Select Register 3 (XBARA_SEL3)	16	R/W	0000h	15.5.4/385
E344	Crossbar A Select Register 4 (XBARA_SEL4)	16	R/W	0000h	15.5.5/386
E345	Crossbar A Select Register 5 (XBARA_SEL5)	16	R/W	0000h	15.5.6/386
E346	Crossbar A Select Register 6 (XBARA_SEL6)	16	R/W	0000h	15.5.7/387
E347	Crossbar A Select Register 7 (XBARA_SEL7)	16	R/W	0000h	15.5.8/387
E348	Crossbar A Select Register 8 (XBARA_SEL8)	16	R/W	0000h	15.5.9/388
E349	Crossbar A Select Register 9 (XBARA_SEL9)	16	R/W	0000h	15.5.10/388
E34A	Crossbar A Select Register 10 (XBARA_SEL10)	16	R/W	0000h	15.5.11/389
E34B	Crossbar A Select Register 11 (XBARA_SEL11)	16	R/W	0000h	15.5.12/389
E34C	Crossbar A Select Register 12 (XBARA_SEL12)	16	R/W	0000h	15.5.13/390
E34D	Crossbar A Select Register 13 (XBARA_SEL13)	16	R/W	0000h	15.5.14/390
E34E	Crossbar A Select Register 14 (XBARA_SEL14)	16	R/W	0000h	15.5.15/391
E34F	Crossbar A Select Register 15 (XBARA_SEL15)	16	R/W	0000h	15.5.16/391
E350	Crossbar A Select Register 16 (XBARA_SEL16)	16	R/W	0000h	15.5.17/392
E351	Crossbar A Select Register 17 (XBARA_SEL17)	16	R/W	0000h	15.5.18/392
E352	Crossbar A Select Register 18 (XBARA_SEL18)	16	R/W	0000h	15.5.19/393
E353	Crossbar A Select Register 19 (XBARA_SEL19)	16	R/W	0000h	15.5.20/393
E354	Crossbar A Select Register 20 (XBARA_SEL20)	16	R/W	0000h	15.5.21/394
E355	Crossbar A Select Register 21 (XBARA_SEL21)	16	R/W	0000h	15.5.22/394
E356	Crossbar A Select Register 22 (XBARA_SEL22)	16	R/W	0000h	15.5.23/395
E357	Crossbar A Select Register 23 (XBARA_SEL23)	16	R/W	0000h	15.5.24/395
E358	Crossbar A Select Register 24 (XBARA_SEL24)	16	R/W	0000h	15.5.25/396
E359	Crossbar A Select Register 25 (XBARA_SEL25)	16	R/W	0000h	15.5.26/396
E35A	Crossbar A Select Register 26 (XBARA_SEL26)	16	R/W	0000h	15.5.27/397
E35B	Crossbar A Select Register 27 (XBARA_SEL27)	16	R/W	0000h	15.5.28/397
E35C	Crossbar A Select Register 28 (XBARA_SEL28)	16	R/W	0000h	15.5.29/398

Table continues on the next page...

XBARA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E35D	Crossbar A Select Register 29 (XBARA_SEL29)	16	R/W	0000h	15.5.30/398
E35E	Crossbar A Select Register 30 (XBARA_SEL30)	16	R/W	0000h	15.5.31/399
E35F	Crossbar A Select Register 31 (XBARA_SEL31)	16	R/W	0000h	15.5.32/399
E360	Crossbar A Select Register 31 (XBARA_SEL32)	16	R/W	0000h	15.5.33/400
E361	Crossbar A Control Register 0 (XBARA_CTRL0)	16	R/W	0000h	15.5.34/400
E362	Crossbar A Control Register 1 (XBARA_CTRL1)	16	R/W	0000h	15.5.35/402

15.5.1 Crossbar A Select Register 0 (XBARA_SEL0)

Address: E340h base + 0h offset = E340h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL1						0		SEL0					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL0 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL1	Input (XBARA_INn) to be muxed to XBARA_OUT1 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL0	Input (XBARA_INn) to be muxed to XBARA_OUT0 (refer to Functional Description section for input/output assignment)

15.5.2 Crossbar A Select Register 1 (XBARA_SEL1)

Address: E340h base + 1h offset = E341h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL3						0		SEL2					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL1 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

XBARA_SEL1 field descriptions (continued)

Field	Description
13–8 SEL3	Input (XBARA_INn) to be muxed to XBARA_OUT3 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL2	Input (XBARA_INn) to be muxed to XBARA_OUT2 (refer to Functional Description section for input/output assignment)

15.5.3 Crossbar A Select Register 2 (XBARA_SEL2)

Address: E340h base + 2h offset = E342h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL5						0		SEL4					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL2 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL5	Input (XBARA_INn) to be muxed to XBARA_OUT5 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL4	Input (XBARA_INn) to be muxed to XBARA_OUT4 (refer to Functional Description section for input/output assignment)

15.5.4 Crossbar A Select Register 3 (XBARA_SEL3)

Address: E340h base + 3h offset = E343h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL7						0		SEL6					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL3 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL7	Input (XBARA_INn) to be muxed to XBARA_OUT7 (refer to Functional Description section for input/output assignment)

Table continues on the next page...

XBARA_SEL3 field descriptions (continued)

Field	Description
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL6	Input (XBARA_INn) to be muxed to XBARA_OUT6 (refer to Functional Description section for input/output assignment)

15.5.5 Crossbar A Select Register 4 (XBARA_SEL4)

Address: E340h base + 4h offset = E344h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL9						0		SEL8					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL4 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL9	Input (XBARA_INn) to be muxed to XBARA_OUT9 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL8	Input (XBARA_INn) to be muxed to XBARA_OUT8 (refer to Functional Description section for input/output assignment)

15.5.6 Crossbar A Select Register 5 (XBARA_SEL5)

Address: E340h base + 5h offset = E345h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL11						0		SEL10					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL5 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL11	Input (XBARA_INn) to be muxed to XBARA_OUT11 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

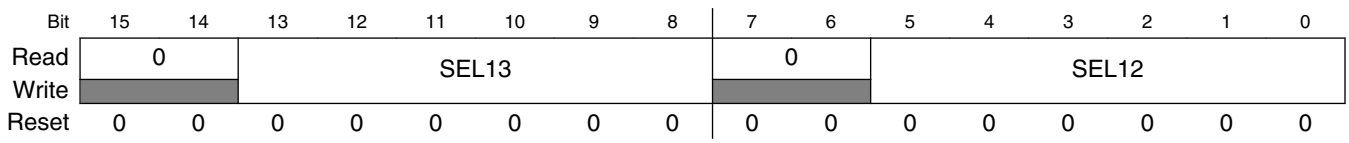
Table continues on the next page...

XBARA_SEL5 field descriptions (continued)

Field	Description
SEL10	Input (XBARA_INn) to be muxed to XBARA_OUT10 (refer to Functional Description section for input/ output assignment)

15.5.7 Crossbar A Select Register 6 (XBARA_SEL6)

Address: E340h base + 6h offset = E346h

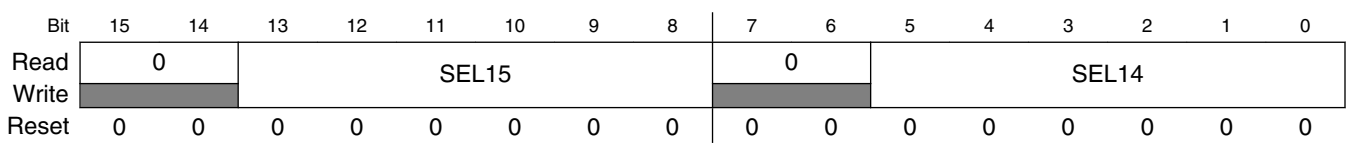


XBARA_SEL6 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL13	Input (XBARA_INn) to be muxed to XBARA_OUT13 (refer to Functional Description section for input/ output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL12	Input (XBARA_INn) to be muxed to XBARA_OUT12 (refer to Functional Description section for input/ output assignment)

15.5.8 Crossbar A Select Register 7 (XBARA_SEL7)

Address: E340h base + 7h offset = E347h

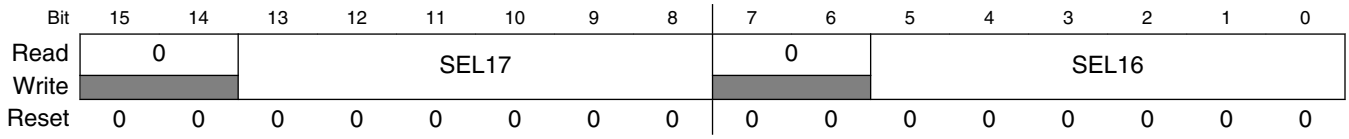


XBARA_SEL7 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL15	Input (XBARA_INn) to be muxed to XBARA_OUT15 (refer to Functional Description section for input/ output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL14	Input (XBARA_INn) to be muxed to XBARA_OUT14 (refer to Functional Description section for input/ output assignment)

15.5.9 Crossbar A Select Register 8 (XBARA_SEL8)

Address: E340h base + 8h offset = E348h

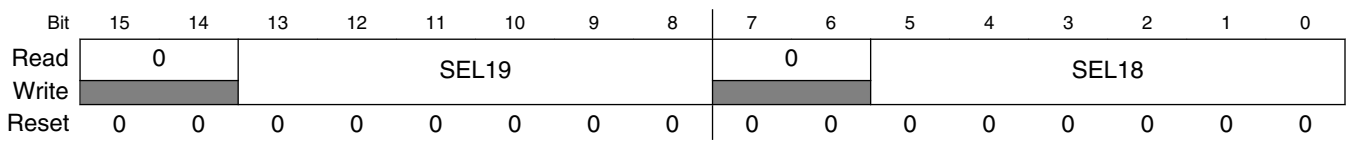


XBARA_SEL8 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL17	Input (XBARA_INn) to be muxed to XBARA_OUT17 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL16	Input (XBARA_INn) to be muxed to XBARA_OUT16 (refer to Functional Description section for input/output assignment)

15.5.10 Crossbar A Select Register 9 (XBARA_SEL9)

Address: E340h base + 9h offset = E349h



XBARA_SEL9 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL19	Input (XBARA_INn) to be muxed to XBARA_OUT19 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL18	Input (XBARA_INn) to be muxed to XBARA_OUT18 (refer to Functional Description section for input/output assignment)

15.5.11 Crossbar A Select Register 10 (XBARA_SEL10)

Address: E340h base + Ah offset = E34Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL21						0		SEL20					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL10 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL21	Input (XBARA_INn) to be muxed to XBARA_OUT21 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL20	Input (XBARA_INn) to be muxed to XBARA_OUT20 (refer to Functional Description section for input/output assignment)

15.5.12 Crossbar A Select Register 11 (XBARA_SEL11)

Address: E340h base + Bh offset = E34Bh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL23						0		SEL22					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL11 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL23	Input (XBARA_INn) to be muxed to XBARA_OUT23 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL22	Input (XBARA_INn) to be muxed to XBARA_OUT22 (refer to Functional Description section for input/output assignment)

15.5.13 Crossbar A Select Register 12 (XBARA_SEL12)

Address: E340h base + Ch offset = E34Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL25						0		SEL24					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL12 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL25	Input (XBARA_INn) to be muxed to XBARA_OUT25 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL24	Input (XBARA_INn) to be muxed to XBARA_OUT24 (refer to Functional Description section for input/output assignment)

15.5.14 Crossbar A Select Register 13 (XBARA_SEL13)

Address: E340h base + Dh offset = E34Dh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL27						0		SEL26					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL13 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL27	Input (XBARA_INn) to be muxed to XBARA_OUT27 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL26	Input (XBARA_INn) to be muxed to XBARA_OUT26 (refer to Functional Description section for input/output assignment)

15.5.15 Crossbar A Select Register 14 (XBARA_SEL14)

Address: E340h base + Eh offset = E34Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL29						0		SEL28					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL14 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL29	Input (XBARA_INn) to be muxed to XBARA_OUT29 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL28	Input (XBARA_INn) to be muxed to XBARA_OUT28 (refer to Functional Description section for input/output assignment)

15.5.16 Crossbar A Select Register 15 (XBARA_SEL15)

Address: E340h base + Fh offset = E34Fh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL31						0		SEL30					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL15 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL31	Input (XBARA_INn) to be muxed to XBARA_OUT31 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL30	Input (XBARA_INn) to be muxed to XBARA_OUT30 (refer to Functional Description section for input/output assignment)

15.5.17 Crossbar A Select Register 16 (XBARA_SEL16)

Address: E340h base + 10h offset = E350h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL33						0		SEL32					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL16 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL33	Input (XBARA_INn) to be muxed to XBARA_OUT33 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL32	Input (XBARA_INn) to be muxed to XBARA_OUT32 (refer to Functional Description section for input/output assignment)

15.5.18 Crossbar A Select Register 17 (XBARA_SEL17)

Address: E340h base + 11h offset = E351h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL35						0		SEL34					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL17 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL35	Input (XBARA_INn) to be muxed to XBARA_OUT35 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL34	Input (XBARA_INn) to be muxed to XBARA_OUT34 (refer to Functional Description section for input/output assignment)

15.5.19 Crossbar A Select Register 18 (XBARA_SEL18)

Address: E340h base + 12h offset = E352h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL37						0		SEL36					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL18 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL37	Input (XBARA_INn) to be muxed to XBARA_OUT37 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL36	Input (XBARA_INn) to be muxed to XBARA_OUT36 (refer to Functional Description section for input/output assignment)

15.5.20 Crossbar A Select Register 19 (XBARA_SEL19)

Address: E340h base + 13h offset = E353h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL39						0		SEL38					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL19 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL39	Input (XBARA_INn) to be muxed to XBARA_OUT39 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL38	Input (XBARA_INn) to be muxed to XBARA_OUT38 (refer to Functional Description section for input/output assignment)

15.5.21 Crossbar A Select Register 20 (XBARA_SEL20)

Address: E340h base + 14h offset = E354h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL41						0		SEL40					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL20 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL41	Input (XBARA_INn) to be muxed to XBARA_OUT41 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL40	Input (XBARA_INn) to be muxed to XBARA_OUT40 (refer to Functional Description section for input/output assignment)

15.5.22 Crossbar A Select Register 21 (XBARA_SEL21)

Address: E340h base + 15h offset = E355h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL43						0		SEL42					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL21 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL43	Input (XBARA_INn) to be muxed to XBARA_OUT43 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL42	Input (XBARA_INn) to be muxed to XBARA_OUT42 (refer to Functional Description section for input/output assignment)

15.5.23 Crossbar A Select Register 22 (XBARA_SEL22)

Address: E340h base + 16h offset = E356h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL45						0		SEL44					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL22 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL45	Input (XBARA_INn) to be muxed to XBARA_OUT45 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL44	Input (XBARA_INn) to be muxed to XBARA_OUT44 (refer to Functional Description section for input/output assignment)

15.5.24 Crossbar A Select Register 23 (XBARA_SEL23)

Address: E340h base + 17h offset = E357h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL47						0		SEL46					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL23 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL47	Input (XBARA_INn) to be muxed to XBARA_OUT47 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL46	Input (XBARA_INn) to be muxed to XBARA_OUT46 (refer to Functional Description section for input/output assignment)

15.5.25 Crossbar A Select Register 24 (XBARA_SEL24)

Address: E340h base + 18h offset = E358h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL49						0		SEL48					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL24 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL49	Input (XBARA_INn) to be muxed to XBARA_OUT49 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL48	Input (XBARA_INn) to be muxed to XBARA_OUT48 (refer to Functional Description section for input/output assignment)

15.5.26 Crossbar A Select Register 25 (XBARA_SEL25)

Address: E340h base + 19h offset = E359h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL51						0		SEL50					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL25 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL51	Input (XBARA_INn) to be muxed to XBARA_OUT51 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL50	Input (XBARA_INn) to be muxed to XBARA_OUT50 (refer to Functional Description section for input/output assignment)

15.5.27 Crossbar A Select Register 26 (XBARA_SEL26)

Address: E340h base + 1Ah offset = E35Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL53						0		SEL52					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL26 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL53	Input (XBARA_INn) to be muxed to XBARA_OUT53 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL52	Input (XBARA_INn) to be muxed to XBARA_OUT52 (refer to Functional Description section for input/output assignment)

15.5.28 Crossbar A Select Register 27 (XBARA_SEL27)

Address: E340h base + 1Bh offset = E35Bh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL55						0		SEL54					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL27 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL55	Input (XBARA_INn) to be muxed to XBARA_OUT55 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL54	Input (XBARA_INn) to be muxed to XBARA_OUT54 (refer to Functional Description section for input/output assignment)

15.5.29 Crossbar A Select Register 28 (XBARA_SEL28)

Address: E340h base + 1Ch offset = E35Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL57						0		SEL56					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL28 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL57	Input (XBARA_INn) to be muxed to XBARA_OUT57 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL56	Input (XBARA_INn) to be muxed to XBARA_OUT56 (refer to Functional Description section for input/output assignment)

15.5.30 Crossbar A Select Register 29 (XBARA_SEL29)

Address: E340h base + 1Dh offset = E35Dh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL59						0		SEL58					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL29 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL59	Input (XBARA_INn) to be muxed to XBARA_OUT59 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL58	Input (XBARA_INn) to be muxed to XBARA_OUT58 (refer to Functional Description section for input/output assignment)

15.5.31 Crossbar A Select Register 30 (XBARA_SEL30)

Address: E340h base + 1Eh offset = E35Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL61						0		SEL60					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL30 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL61	Input (XBARA_INn) to be muxed to XBARA_OUT61 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL60	Input (XBARA_INn) to be muxed to XBARA_OUT60 (refer to Functional Description section for input/output assignment)

15.5.32 Crossbar A Select Register 31 (XBARA_SEL31)

Address: E340h base + 1Fh offset = E35Fh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL63						0		SEL62					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL31 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL63	Input (XBARA_INn) to be muxed to XBARA_OUT63 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL62	Input (XBARA_INn) to be muxed to XBARA_OUT62 (refer to Functional Description section for input/output assignment)

15.5.33 Crossbar A Select Register 31 (XBARA_SEL32)

Address: E340h base + 20h offset = E360h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL65						0		SEL64					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA_SEL32 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL65	Input (XBARA_INn) to be muxed to XBARA_OUT65 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL64	Input (XBARA_INn) to be muxed to XBARA_OUT64 (refer to Functional Description section for input/output assignment)

15.5.34 Crossbar A Control Register 0 (XBARA_CTRL0)

Use this register to configure edge detection, interrupt, and DMA features for the XBAR_OUT0 and XBAR_OUT1 outputs.

The XBAR_CTRL registers are organized similarly to the XBAR_SEL registers, with control fields for two XBAR_OUT outputs in each register. In control register 0, the LSBs contain the control fields for XBAR_OUT0, and the MSBs contain the control fields for XBAR_OUT1.

Address: E340h base + 21h offset = E361h

Bit	15	14	13	12	11	10	9	8
Read	0			STS1	EDGE1		IEN1	DEN1
Write				w1c				
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0			STS0	EDGE0		IEN0	DEN0
Write				w1c				
Reset	0	0	0	0	0	0	0	0

XBARA_CTRL0 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 STS1	Edge detection status for XBAR_OUT1 This bit reflects the results of edge detection for XBAR_OUT1. This field is set to 1 when an edge consistent with the current setting of EDGE1 is detected on XBAR_OUT1. This field is cleared by writing 1 to it or by a DMA_ACK1 reception when DEN1 is set. Writing 0 to the field has no effect. When interrupt or DMA functionality is enabled for XBAR_OUT1, this field is 1 when the interrupt or DMA request is asserted and 0 when the interrupt or DMA request has been cleared. 0 Active edge not yet detected on XBAR_OUT1 1 Active edge detected on XBAR_OUT1
11–10 EDGE1	Active edge for edge detection on XBAR_OUT1 This field selects which edges on XBAR_OUT1 cause STS1 to assert. 00 STS1 never asserts 01 STS1 asserts on rising edges of XBAR_OUT1 10 STS1 asserts on falling edges of XBAR_OUT1 11 STS1 asserts on rising and falling edges of XBAR_OUT1
9 IEN1	Interrupt Enable for XBAR_OUT1 This bit enables the interrupt function on the corresponding XBAR_OUT1 output. When the interrupt is enabled, the output INT_REQ1 reflects the value STS1. When the interrupt is disabled, INT_REQ1 remains low. The interrupt request is cleared by writing a 1 to STS1. **Restriction:** IEN1 and DEN1 should not both be set to 1. 0 Interrupt disabled 1 Interrupt enabled
8 DEN1	DMA Enable for XBAR_OUT1 This bit enables the DMA function on the corresponding XBAR_OUT1 output. When enabled, DMA_REQ1 presents the value STS1. When disabled, the DMA_REQ1 output remains low. **Restriction:** IEN1 and DEN1 should not both be set to 1. 0 DMA disabled 1 DMA enabled
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 STS0	Edge detection status for XBAR_OUT0 This bit reflects the results of edge detection for XBAR_OUT0. This field is set to 1 when an edge consistent with the current setting of EDGE0 is detected on XBAR_OUT0. This field is cleared by writing 1 to it or by a DMA_ACK0 reception when DEN0 is set. Writing 0 to the field has no effect. When interrupt or DMA functionality is enabled for XBAR_OUT0, this field is 1 when the interrupt or DMA request is asserted and 0 when the interrupt or DMA request has been cleared.

Table continues on the next page...

XBARA_CTRL0 field descriptions (continued)

Field	Description
	0 Active edge not yet detected on XBAR_OUT0 1 Active edge detected on XBAR_OUT0
3–2 EDGE0	Active edge for edge detection on XBAR_OUT0 This field selects which edges on XBAR_OUT0 cause STS0 to assert. 00 STS0 never asserts 01 STS0 asserts on rising edges of XBAR_OUT0 10 STS0 asserts on falling edges of XBAR_OUT0 11 STS0 asserts on rising and falling edges of XBAR_OUT0
1 IEN0	Interrupt Enable for XBAR_OUT0 This bit enables the interrupt function on the corresponding XBAR_OUT0 output. When the interrupt is enabled, the output INT_REQ0 reflects the value STS0. When the interrupt is disabled, INT_REQ0 remains low. The interrupt request is cleared by writing a 1 to STS0. **Restriction:** IEN0 and DEN0 should not both be set to 1. 0 Interrupt disabled 1 Interrupt enabled
0 DEN0	DMA Enable for XBAR_OUT0 This bit enables the DMA function on the corresponding XBAR_OUT0 output. When enabled, DMA_REQ0 presents the value STS0. When disabled, the DMA_REQ0 output remains low. **Restriction:** IEN0 and DEN0 should not both be set to 1. 0 DMA disabled 1 DMA enabled

15.5.35 Crossbar A Control Register 1 (XBARA_CTRL1)

Use this register to configure edge detection, interrupt, and DMA features for the XBAR_OUT2 and XBAR_OUT3 outputs.

The XBAR_CTRL registers are organized similarly to the XBAR_SEL registers, with control fields for two XBAR_OUT outputs in each register. In control register 1, the LSBs contain the control fields for XBAR_OUT2, and the MSBs contain the control fields for XBAR_OUT3.

Address: E340h base + 22h offset = E362h

Bit	15	14	13	12	11	10	9	8
Read	0			STS3	EDGE3		IEN3	DEN3
Write				w1c				
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0			STS2	EDGE2		IEN2	DEN2
Write				w1c				
Reset	0	0	0	0	0	0	0	0

XBARA_CTRL1 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 STS3	Edge detection status for XBAR_OUT3 This bit reflects the results of edge detection for XBAR_OUT3. This field is set to 1 when an edge consistent with the current setting of EDGE3 is detected on XBAR_OUT3. This field is cleared by writing 1 to it or by a DMA_ACK3 reception when DEN3 is set. Writing 0 to the field has no effect. When interrupt or DMA functionality is enabled for XBAR_OUT3, this field is 1 when the interrupt or DMA request is asserted and 0 when the interrupt or DMA request has been cleared. 0 Active edge not yet detected on XBAR_OUT3 1 Active edge detected on XBAR_OUT3
11–10 EDGE3	Active edge for edge detection on XBAR_OUT3 This field selects which edges on XBAR_OUT3 cause STS3 to assert. 00 STS3 never asserts 01 STS3 asserts on rising edges of XBAR_OUT3 10 STS3 asserts on falling edges of XBAR_OUT3 11 STS3 asserts on rising and falling edges of XBAR_OUT3
9 IEN3	Interrupt Enable for XBAR_OUT3 This bit enables the interrupt function on the corresponding XBAR_OUT3 output. When the interrupt is enabled, the output INT_REQ3 reflects the value STS3. When the interrupt is disabled, INT_REQ3 remains low. The interrupt request is cleared by writing a 1 to STS3. **Restriction:** IEN3 and DEN3 should not both be set to 1. 0 Interrupt disabled 1 Interrupt enabled
8 DEN3	DMA Enable for XBAR_OUT3 This bit enables the DMA function on the corresponding XBAR_OUT3 output. When enabled, DMA_REQ3 presents the value STS3. When disabled, the DMA_REQ3 output remains low. **Restriction:** IEN3 and DEN3 should not both be set to 1. 0 DMA disabled 1 DMA enabled
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 STS2	Edge detection status for XBAR_OUT2 This bit reflects the results of edge detection for XBAR_OUT2.

Table continues on the next page...

XBARA_CTRL1 field descriptions (continued)

Field	Description
	This field is set to 1 when an edge consistent with the current setting of EDGE2 is detected on XBAR_OUT2. This field is cleared by writing 1 to it or by a DMA_ACK2 reception when DEN2 is set. Writing 0 to the field has no effect. When interrupt or DMA functionality is enabled for XBAR_OUT2, this field is 1 when the interrupt or DMA request is asserted and 0 when the interrupt or DMA request has been cleared. 0 Active edge not yet detected on XBAR_OUT2 1 Active edge detected on XBAR_OUT2
3–2 EDGE2	Active edge for edge detection on XBAR_OUT2 This field selects which edges on XBAR_OUT2 cause STS2 to assert. 00 STS2 never asserts 01 STS2 asserts on rising edges of XBAR_OUT2 10 STS2 asserts on falling edges of XBAR_OUT2 11 STS2 asserts on rising and falling edges of XBAR_OUT2
1 IEN2	Interrupt Enable for XBAR_OUT2 This bit enables the interrupt function on the corresponding XBAR_OUT2 output. When the interrupt is enabled, the output INT_REQ2 reflects the value STS2. When the interrupt is disabled, INT_REQ2 remains low. The interrupt request is cleared by writing a 1 to STS2. **Restriction:** IEN2 and DEN2 should not both be set to 1. 0 Interrupt disabled 1 Interrupt enabled
0 DEN2	DMA Enable for XBAR_OUT2 This bit enables the DMA function on the corresponding XBAR_OUT2 output. When enabled, DMA_REQ2 presents the value STS2. When disabled, the DMA_REQ2 output remains low. **Restriction:** IEN2 and DEN2 should not both be set to 1. 0 DMA disabled 1 DMA enabled

Chapter 16
On-Chip Clock Synthesis (OCCS)

16.1 Overview

The on-chip clock synthesis (OCCS) module provides various clock sources and rates to different peripherals, which use the multiple clocks to perform the operation and monitor functions. The main clock is a 2X system clock frequency (mstr_2x_clk) to the system integration module (SIM), which then generates the various derivative system and peripheral clocks for the chip.

All of the user selectable clocks are derived from an 8 MHz(standby 2MHz) internal RC oscillator (IRC8M),a 200 kHz internal RC oscillator (ROSC200k), an external clock input, a 4-16 MHz external crystal oscillator (XOSC), and a PLL to run up to a 100 MHz system bus frequency.

NOTE

The 8MHz IRC oscillator has a reduced power standby mode at which it operates at 2MHz. For clarity, this will be referred to as IRC8M throughout this chapter.

16.1.1 Block Diagram

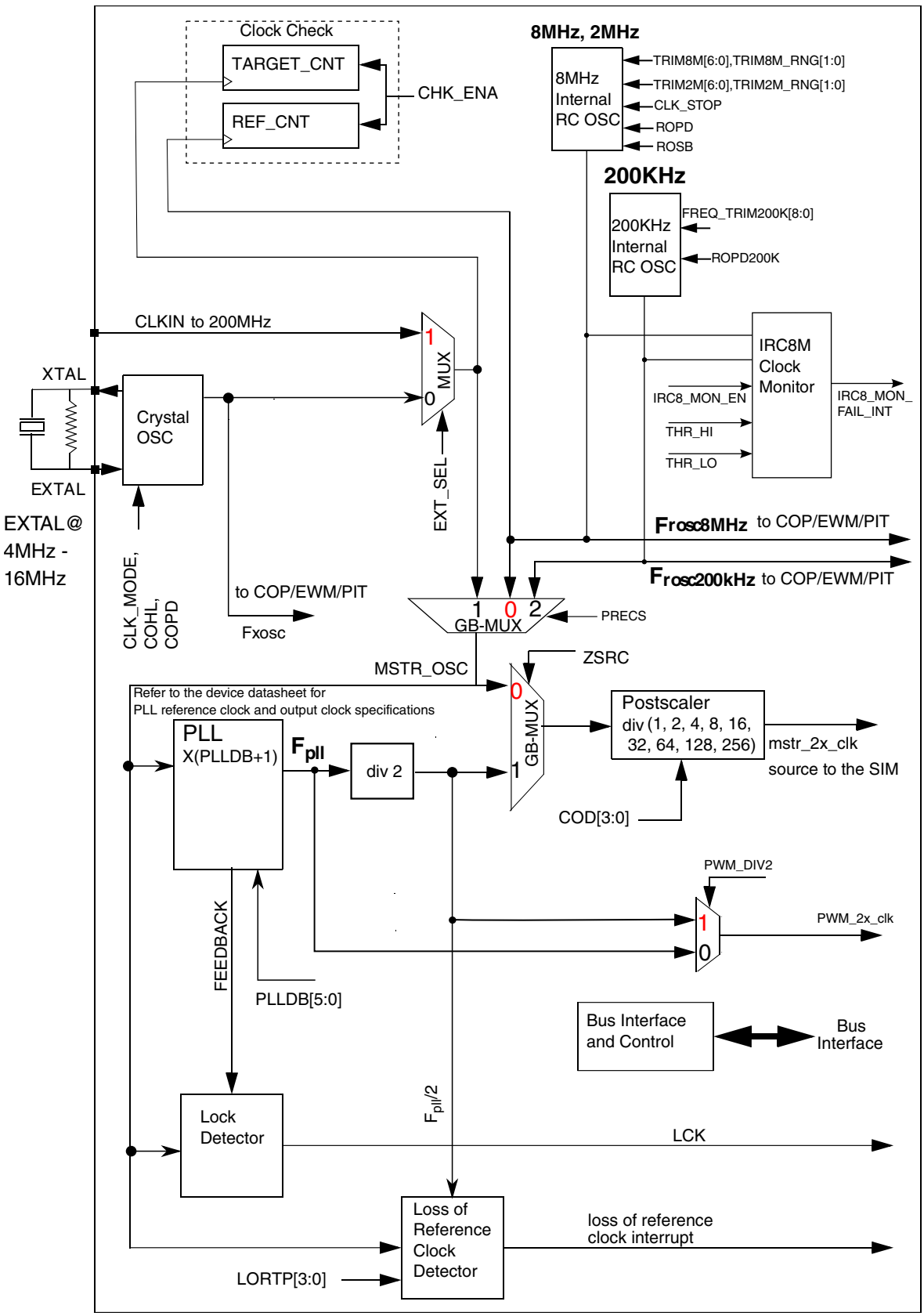


Figure 16-1. OCCS Block Diagram with Crystal Oscillator

16.1.2 Features

OCCS interfaces to the on-chip clock sources, PLL. The device has a wide variety of options for clock generation. The OCCS reference clock sources include:

- Internal 8MHz Oscillator (IRC8M) with frequency and temperature trim control, which can be used as the main system. This module nominally generates an 8MHz clock signal. It is also capable of operating at 2MHz in standby mode.
- Internal 200kHz RC oscillator with frequency trim control.
- Crystal Oscillator (XOSC), which can also be used as the main system. This module is designed for use with a crystal or resonator in the 4 to 16 MHz range. When used with the on-chip PLL, the maximum crystal/resonator frequency is 16 MHz and the minimum is 8 MHz.
- Off-chip external clock source

Other clock generation features include:

- PLL supporting programmable integer feedback divide (multiply factor). Refer to the device datasheet for detailed PLL input clock and output clock specifications
- PLL divide by 2 act as a digital duty cycle corrector.
- Glitch free selection of input clock source.
- Glitch free selection of output clock to be any input clock source or PLL.
- Output clock postscaler with divide power of 2 divide factors from 1 to 256.
- All input clock sources available as OCCS outputs.

NOTE

The postscaled output clock (mstr_2x_clk) is output to the SIM module which divides it by 2 to produce the system bus clocks.

Additional features in OCCS are as follows:

- Ability to power down the 8 MHz internal RC oscillator
- Ability to put the 8MHz internal RC oscillator into a reduced power 2MHz standby mode
- Ability to power down external crystal oscillator
- Ability to power down the 200 kHz internal RC oscillator
- 4-bit postscaler provides operates on either PLL output or, in the case where the PLL is not in use, one of the oscillators or external clock source.
- Ability to power down the internal PLL
- Provides 2X master clock frequency and 2X High Speed Peripheral clock signals
- Optional automatic switch to backup clock on loss of clock reference
- PLL Lock or loss-of-lock detection
- Memory mapped registers for configuring the OCCS and internal clock sources

Key features of the crystal oscillator module are:

- Supports 4 MHz - 16 MHz crystals and resonators
- High gain option
- Voltage and frequency filtering to guarantee clock frequency and stability

16.2 Functional Description

A block diagram of the OCCS module is shown in Figure 16-1.

The chip provides several alternative clock sources. This signal is referred to as MSTR_OSC in the block diagram of the OCCS. Possible clock source choices are:

- Internal 8 MHz RC oscillator with 2MHz standby mode
- Internal 200 kHz oscillator
- External ceramic resonator or crystal oscillator attached to XTAL/EXTAL pad signals
- External clock source on CLKIN

The active clock source (MSTR_OSC) is selected by a glitch-free mux controlled by the setting of the PRECS field. The Internal 8 MHz clock is selected at reset. MSTR_OSC is used as the input to the PLL which can be used to derive higher frequencies. When MSTR_OSC is within the spec of the PLL (Refer to the device datasheet for detailed specifications), the PLL can be used to perform integer multiplication of MSTR_OSC to provide a high-speed clock source for the part. The PLL output is fed to a divide by 2 circuit which acts as a duty cycle corrector.

The primary output clock or clock reference to the SIM module is named mstr_2x_clk (also referred to as MSTR_2X by the SIM). The clock reference to the SIM is selected by another glitch-free mux controlled by the setting of the ZSRC field. This mux is configured to select MSTR_OSC at reset, however, the clock reference can be changed using ZSRC to the PLL output divided by 2. A post-scaler is provided which can divide the clock reference by from 1 to 256 before being output to the SIM. This post-scaler can be reconfigured any time without inducing glitches on the reference clock output to the SIM.

When ZSRC is selecting MSTR_OSC as the current clock reference, the post-scaler can be used to provide very low operating frequencies for the part. When ZSRC is selecting a PLL based clock source, the post-scaler can be configured to provide a wide spectrum of intermediate frequencies all the way up to 200MHz. The SIM uses this clock reference directly for high speed applications and divides it by 2 and gates it to generate all the system and peripheral clocks which operate at a maximum 100MHz.

The OCCS Status Register (STAT) shows the status of the DSP core clock source. Because the synchronizing circuit changes modes to avoid any glitches, the STAT ZCLOCK source (ZSRCS) will show overlapping modes as an intermediate step. After PLL lock is detected the DSP core clock can be switched to the PLL by writing to the ZSRC bits in the CTRL register.

A proper sequence must be followed when reconfiguring the OCCS to insure correct operation of the part. Before changing PRECS to change to a new clock source, the new clock source should be powered on, configured, and stable. PRECS should not be changed while ZSRC is selecting a PLL based clock reference. This amounts to changing the PLL input while its output is in use and would cause the clock reference to become unstable. The user should also allow for the fact that transitions of the glitch-free muxes controlled by PRECS and ZSRC require two clock periods of the old clock to disable it, and two clock periods of the new clock to enable it.

Frequencies going out of the OCCS are controlled by the postscaler, and/or the divide-by ratio within the PLL. For proper operation of the PLL, the user must keep the VCO, within the PLL, in its operational range(Refer to the device datasheet for detailed specifications), the output of the VCO is depicted as F_{pll} in Figure 16-1. The input frequency multiplied by the divide-by ratio is the frequency at which the VCO is running.

It is recommended when powering down, or powering up, the PLL be deselected as the clocking source. Only after lock is achieved should the PLL be selected as a valid clocking source.

Table 16-1 shows how to configure from reset defaults to any available clock configuration.

NOTE

In the following table, the clock source refers to the PRECS selection and the clock output refers to the ZSRC selection. Configuring the clock output for direct clocking means that the PLL is not in use and ZSRC is selecting MSTR_OSC.

Table 16-1. Clock Choices Without Crystal Oscillator

Clock Source	Clock Output	Configuration Steps
8 MHz RC Oscillator	Direct	Default. 1. Change the COD, if desired.
200 kHz RC Oscillator	Direct	1. Select the 200 kHz oscillator (PRECS=10). **NOTE:** Power up 200 kHz RC oscillator first. 2. Wait 6 NOPs for resynchronization. 3. The 8MHz RC oscillator can optionally be powered down (ROPD=1). 4. Change the COD, if desired.
8 MHz RC Oscillator	Any PLL	1. Set PLLDB for desired multiply and enable the PLL (PLLPD=0)

Table continues on the next page...

Table 16-1. Clock Choices Without Crystal Oscillator (continued)

Clock Source	Clock Output	Configuration Steps
		2. Wait for PLL lock (LCK1=1 and LCK0=1) 3. Change ZSRC to select a PLL based source 4. Change the COD, if desired.
External Clock Source	Direct	1. The clock source (CLKIN) should be enabled in the GPIO and SIM. 2. Select CLKIN as the source clock (PRECS=01, EXT_SEL=1). 3. Wait 6 NOP's for the synchronizing circuit to change clocks. 4. The 8MHz RC oscillator can optionally be powered down (ROPD=1). 5. Change the COD, if desired.
External Clock Source	Any PLL	1. The clock source (CLKIN) should be enabled in the GPIO and SIM. 2. Select CLKIN as the source clock (PRECS=01, EXT_SEL=1). 3. Wait 6 NOP's for the synchronizing circuit to change clocks. 4. The 8MHz RC oscillator can optionally be powered down (ROPD=1). 5. Set PLLDB for desired multiply and enable the PLL (PLLPD=0). 6. Wait for PLL lock (LCK1=1 and LCK0=1) 7. Change ZSRC to select a PLL based output clock. 8. Change the COD, if desired.

NOTE

All of the crystal oscillator options in Table 16-2 require enabling the oscillator by setting OSCTL2[COPD] to 0.

Table 16-2. Clock Choices with Crystal Oscillator

Clock Source	Clock Output	Configuration Steps
Crystal Oscillator or Resonator in FSP mode	Direct	1. The pin functions XTAL and EXTAL should be enabled in the GPIO and SIM. 2. The external clock source should be changed to the crystal oscillator (EXT_SEL=0). 3. The crystal oscillator should be configured to FSP mode (CLK_MODE=0, COHL=0). 4. Power up the crystal oscillator (COPD=0). 5. Wait for the oscillator to stabilize (OSC_OK=1). 6. The crystal oscillator clock source should be selected (PRECS=01). 7. Wait 6 NOP's for the synchronizing circuit to change clocks. 8. The 8MHz RC oscillator can optionally be powered down (ROPD=1). 9. Change the COD, if desired.
Crystal Oscillator or Resonator in FSP mode	Any PLL	1. The pin functions XTAL and EXTAL should be enabled in the GPIO and SIM. 2. The external clock source should be set to the crystal oscillator (EXT_SEL=0). 3. The crystal oscillator should be configured to FSP mode (CLK_MODE=0, COHL=0). 4. Power up the crystal oscillator (COPD=0). 5. Wait for the crystal oscillator to stabilize (OSC_OK=1). 6. The crystal oscillator clock source should be selected (PRECS=01). 7. Wait 6 NOP's for the synchronizing circuit to change clocks. 8. The 8MHz RC oscillator can optionally be powered down (ROPD=1). 9. Set PLLDB for desired multiply and enable the PLL (PLLPD=0) 10. Wait for PLL lock (LCK1=1 and LCK0=1). 11. Change ZSRC to select a PLL based output clock. 12. Change COD, if desired.

Table continues on the next page...

Table 16-2. Clock Choices with Crystal Oscillator (continued)

Clock Source	Clock Output	Configuration Steps
Crystal Oscillator or Resonator in LCP mode	Direct	1. The pin functions XTAL and EXTAL should be enabled in the GPIO and SIM. 2. The external clock source should be changed to the crystal oscillator (EXT_SEL=0). 3. The crystal oscillator should be configured to LCP mode (CLK_MODE=0, COHL=1). 4. Power up the crystal oscillator (COPD=0). 5. Wait for the oscillator to stabilize (OSC_OK=1). 6. The crystal oscillator clock source should be selected (PRECS=01). 7. Wait 6 NOP's for the synchronizing circuit to change clocks. 8. The 8MHz RC oscillator can optionally be powered down (ROPD=1). 9. Change COD, if desired.
Crystal Oscillator or Resonator in LCP mode	Any PLL	1. The pin functions XTAL and EXTAL should be enabled in the GPIO and SIM. 2. The external clock source should be set to the crystal oscillator (EXT_SEL=0). 3. The crystal oscillator should be configured to LCP mode (CLK_MODE=0, COHL=1). 4. Power up the crystal oscillator (COPD=0). 5. Wait for the crystal oscillator to stabilize (OSC_OK=1). 6. The crystal oscillator clock source should be selected (PRECS=01). 7. Wait 6 NOP's for the synchronizing circuit to change clocks. 8. The 8MHz RC oscillator can optionally be powered down (ROPD=1). 9. Set PLLDB for desired multiply and enable the PLL (PLLPD=0). 10. Wait for PLL lock (LCK1=1 and LCK0=1). 11. Change ZSRC to select a PLL based output clock. 12. Change COD, if desired

16.2.1 Modes of Operation

Either an internal oscillator, a crystal oscillator, or an external frequency source can be used to provide a reference clock (mstr_2x_clk) to SIM.

The 2X system clock source output from OCCS has a maximum supported frequency of 200MHz and can be described by one of the following equations:

```
2X system frequency = (reference clock frequency) / (postscaler)

2X system frequency = (reference clock frequency X PLL divide factor) / (2 × postscaler)
```

where:

- postscaler = 1, 2, 4, 8, 16, 32, 64, 128 or 256
- PLL output divider PLL divide factor = integer from 1-64
- Reference clock frequency for use with PLL limited to the range specified in the device datasheet

SIM is responsible for further dividing these frequencies by two, which insures a 50% duty cycle in the system clocks.

16.2.1.1 Clock Sources

All of the clocks in this device are derived from one of the following clock sources.

16.2.1.1.1 Internal Clock Sources

The two internal RC oscillators are optimized for accuracy and programmability while providing power saving configurations to accommodate different operating conditions. The internal RC oscillators have trim controls whose initialization values are determined in the factory to compensate for variability with temperature and voltage and fabrication process. They are very fast in reaching a stable frequency.

Both oscillators are characterized in the factory and recommended trim values delivered in a reserved area in the parts flash memory.

16.2.1.1.1.1 Internal 8Mhz RC Oscillator (IRC8M)

The 8 MHz internal RC oscillator provides a 8 MHz clock at the center of its tuning range. It also supports a 2MHz standby state and a power-down state. Frequency can be adjusted up or down using a primary trim adjustment. This is used to shift the frequency vs. temperature curve up and down to center it on 8 MHz. This is useful for reducing the maximum deviation from nominal frequency over temperature.

This is a default system clock source at power-up and reset. It is used to run the boot ROM and can be used as the system clock source for the application. Because of frequency variation over temperature, it may not meet the timing requirements for some communication peripherals, such as CAN. It also uses to monitor external clock when it is available. During the reset sequence, this oscillator will be enabled and trimmed with factory settings by default. Application code can then switch to another clock source and power down this oscillator if desired.

NOTE

For 8M internal RC oscillator, its factory trim value is loaded to OCCS_OSCTL3[TRIM8M_RNG] and OCCS_OSCTL3[FREQ_TRIM8M] during reset. For 2M clock case, its factory trim value is loaded to OCCS_OSCTL4[TRIM2M_RNG] and OCCS_OSCTL4[FREQ_TRIM2M] during reset.

16.2.1.1.1.2 Internal 200Khz RC Oscillator (IRC200K)

The 200 kHz internal RC oscillator provides a 200 kHz clock at the center frequency of its tuning range. Frequency can be adjusted up or down using a primary trim value. This is used to shift the frequency vs. temperature curve up or down to center it on 200 kHz. During the reset sequence, this oscillator will be disabled by default.

This clock mainly is used as alternative clock source for watchdog and low power timers. It also uses for monitoring IRC8M. If IRC8Mhz clock monitor is enabled, an error flag asserts when a IRC8M clock failure is detected. This clock may also be selected as the system clock source in low power mode.

NOTE

This clock must be enabled when IRC8Mhz clock monitor function enables.

NOTE

For 200kHz internal RC oscillator, its factory trim value is loaded to OCCS_OSCTL2[FREQ_TRIM200K] during reset.

16.2.1.1.2 External Oscillator

The internal crystal oscillator circuit is designed to interface with an external crystal or resonator with a frequency in the range of 4-16 MHz. The oscillator supports two primary modes of operation (via the control bit OCCS_OSCTL1[COHL]): Loop Controlled Pierce mode (LCP) with a frequency range from 4-16MHz and Full Swing Pierce mode (FSP) with a frequency range from 4-16 MHz.

The external crystal or resonator can be used as the main system and CAN bit clock source. When used to supply a source to the internal PLL, the crystal/resonator must meet the PLL reference clock specification (See the device datasheet for details). Follow the crystal supplier's recommendations when selecting a crystal, since crystal parameters determine the component values required to provide maximum stability and reliable start-up. The load capacitance values used in the oscillator circuit design should include all stray layout capacitances. The crystal and associated components should be mounted as close as possible to the EXTAL and XTAL pins to minimize output distortion and start-up stabilization time.

16.2.1.1.3 External Clock Sources

External clock can directly feed into the device through a pin.

16.2.1.1.3.1 External Clock Source - CLKIN

In this mode, an external clock is applied using the external CLKIN pin function. This is propagated into the OCCS and OCCS is configured to select CLKIN as the clock source. The recommended method of connecting an external clock is given in Figure 16-2 and Figure 16-3. On this device, selected GPIO can be programmed to provide the external clock input CLKIN as one of its alternate pin functions. In this mode the AC/DC parametrics (V_{ih}, V_{il} ...) are determined by the GPIO cell.

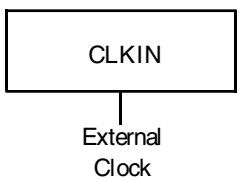


Figure 16-2. Connecting an External Clock Signal using GPIO

16.2.2 RC Oscillators

16.2.2.1 Trimming Frequency on the Internal 8 MHz RC Oscillator

The 8MHz IRC is an internal reference clock designed to deliver an 8MHz output frequency. Under typical operating conditions the circuit provides 8MHz at the center of its tuning range. The tuning range is controlled by 2-bit register TRIM8M_RNG[1:0] and a 7-bit register FREQ_TRIM8M[6:0].

A selectable, trimmable, standby mode of approximately 2MHz is also available. The tuning range is controlled by 2-bit register TRIM2M_RNG[1:0] and a 7-bit register FREQ_TRIM2M[6:0].

16.2.2.2 Trimming Frequency on the Internal 200 kHz RC Oscillator

Like the frequency of the 8 MHz RC oscillator, the frequency of the 200 kHz RC oscillator varies. A single 9-bit trim control is provided in the OSCTL2 register to trim the 200 kHz RC oscillator. Trim values for individual parts are determined at the factory, delivered in the flash memory of the part. Users can adjust the trim similarly to how they adjust it for the 8 MHz RC oscillator.

16.2.3 External Reference

If higher clock precision is required the chip can be operated from an external clock source by CLKIN.

16.2.4 Crystal Oscillator

The crystal oscillator is designed to operate with either an external crystal or resonator. It can also be configured in an external clock bypass mode so that it's extal input is propagated directly to its clock output. This mode is not supported since the use of the CLKIN external pin function avoids frequency limitations on the signal. The oscillator can be operated in either a lower power/lower frequency loop controlled pierce (LCP mode) or a higher power/ higher frequency full swing pierce mode (FSP mode) of operation.

COPD==1, CLK_MODE==x, COHL==x is powerdown mode

COPD==0, CLK_MODE==1, COHL==0 is external clock mode (ext square wave)

NOTE

External clock mode is not supported in this device.

COPD==0, CLK_MODE==0, COHL==0 is FSP mode

COPD==0, CLK_MODE==0, COHL==1 is LCP mode

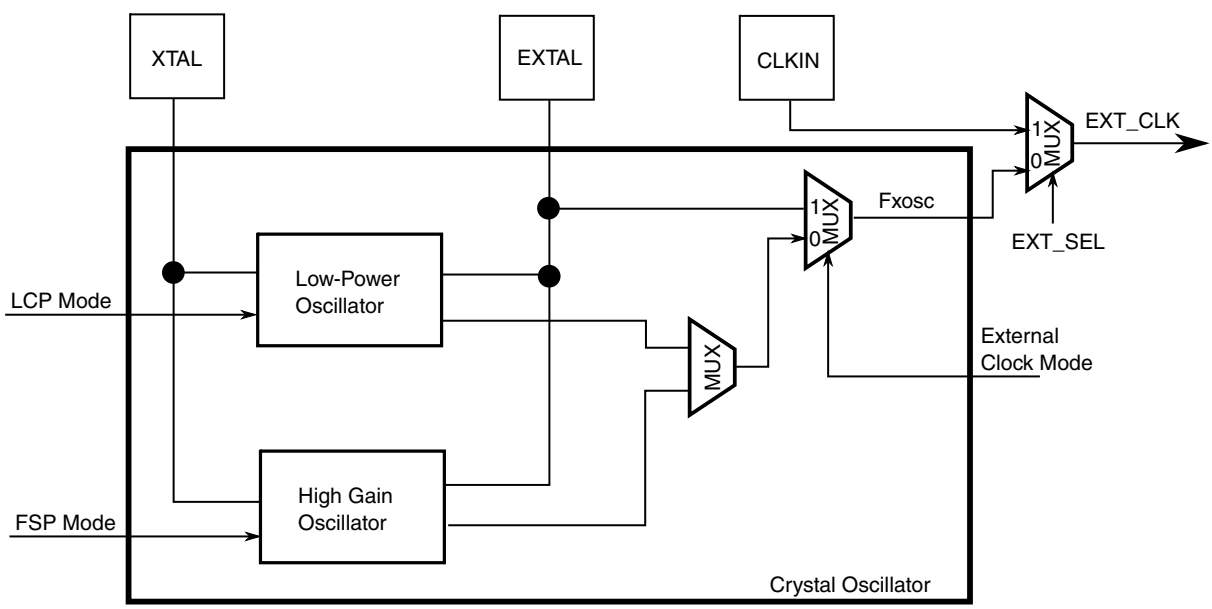


Figure 16-3. Crystal Oscillator Clock Structure

16.2.4.1 Switching Clock Sources

To robustly switch between the Internal RC Oscillator clocks, External oscillator Clock and CLKIN, the changeover switch assumes the clocks are completely asynchronous, so a synchronizing circuit is required to make the transition. When the select input (PRECS) is changed, the switch will continue to operate off the original clock for between 1 and 2 cycles as the select input is transitioned through one side of the synchronizer. Next, the output will be held low for between 1 and 2 cycles of the new clock as the select input transitions through the other side. Then the output starts switching at the new clock's frequency. This transition guarantees that no glitches will be seen on the output even though the select input may change asynchronously to the clocks. The unpredictability of the transition period is a necessary result of the asynchronicity. The switch automatically selects the 8MHz clock during Reset.

Switching sources requires both clock sources to be enabled and stable. A simple flow requires:

- If switching to the crystal oscillator, make sure that XTAL and EXTAL have been enabled via GPIO and SIM and the oscillator is properly configured (COPD, COHL, CLK_MODE).
- If switching to a RC Oscillator, make sure that it is powered up and configured.
- Wait for a few cycles for the clock to become Active.
- Switch clocks
- Execute 6 NOPs instructions
- Disable previous clock source (i.e. power down RC Osc if Crystal is selected).

The key point to remember in this flow is that the clock source should not be switched unless the new desired clock is on and stable.

When a new DSP core clock is selected, the clock generation module will synchronize the request and select the new clock. The OCCS Status Register (STAT) shows the status of the DSP core clock source. Since the synchronizing circuit changes clock sources as to avoid any glitches, the ZSRCS bits in STAT will show overlapping modes as an intermediate step.

16.2.5 Phase Locked Loop

16.2.5.1 PLL Recommended Range of Operation

The voltage controlled oscillator (VCO) within the PLL has a characterized operating range(Refer to the device datasheet for detailed specifications). The output of the PLL, F_{pll}, is fed to the input of the postscaler.

16.2.5.2 PLL Lock Time Specification

In many applications, the lock time of the PLL is the most critical PLL design parameter. Proper use of the PLL ensures the highest stability and lowest lock time.

16.2.5.2.1 Lock Time Definition

Typical control systems refer to the lock time as the reaction time within specified tolerances of the system to a step input. In a PLL, the step input occurs when the PLL is turned on or when it suffers a noise hit. The tolerance is usually specified as a percent of the step input or when the output settles to the desired value plus or minus a percent of the frequency change. Therefore, the reaction time is constant in this definition, regardless of the size of the step input.

When the PLL is coming from a powered down state, PLL_PDN high, to a powered up condition, it will incrementally seek its target output frequency until eventually both the gross and fine lock indicators indicate a locked condition. Refer to device's Data Sheet for lock time specifications. Other systems refer to lock time as the time the system takes to reduce the error between the actual output and the desired output to within specified tolerances. Therefore, the lock time varies according to the original error in the output. Minor errors may be shorter or longer in many cases.

16.2.5.2.2 Parametric Influences on Reaction Time

Lock time is designed to be as short as possible while still providing the highest possible stability. The reaction time is not constant, however. Many factors directly and indirectly affect the lock time.

The most critical parameter affecting the reaction time of the PLL is the reference frequency, MSTR_OSC, illustrated in Figure 16-1. This frequency is the input to the phase detector and controls how often the PLL makes corrections. For stability, it is desirable for the corrections to be small and frequent. Therefore, a higher reference frequency provides optimal performance; 8MHz is minimum.

16.2.5.3 PLL Frequency Lock Detector Block

This digital block monitors the VCO output clock and sets the LCK[1:0] bits in the OCCS Status Register (STAT) based on its frequency accuracy. The lock detector is enabled with the LCKON bit of the PLL control register (CTRL), as well as the PLL_PDN bit of CTRL. Once enabled, the detector starts two counters whose outputs are periodically compared. The input clocks to these counters are the VCO output clock divided by the value in DIVBY[PLLDB], called FEEDBACK, and the PLL input clock, shown as MSTR_OSC in Figure 16-1. The period of the pulses being compared cover one whole period of each clock.

FEEDBACK and MSTR_OSC clocks are compared after 16, 32, and 64 cycles. If, after 32 cycles, the clocks match, the LCK0 bit is set to one. If, after 64 cycles of MSTR_OSC, there is the same number of MSTR_OSC clocks as FEEDBACK clocks, the LCK1 bit is also set. The LCK bit stay set until:

- Clocks fail to match
- On reset caused by LCKON, PLL_PDN
- Chip-level reset

When the circuit sets the LCK1, the two counters are reset and start the count again. The lock detector is designed so if LCK1 is reset to zero because clocks did not match, LCK0 can stay high. This provides the processor the accuracy of the two clocks with respect to each other.

16.2.5.4 Loss of Reference Clock Detector

The loss of reference clock detector is designed to generate an interrupt when the reference clock to the PLL is interrupted. An LOR interrupt should occur after a minimum time of (LORTP+1) x 10 x (reference clock period) / ((PLLDB+1) / 2). This is the minimum time because the PLL output frequency will start to decrease as the PLL tries to track the input reference source. Figure 16-4 illustrates the general operation of the LOR detector, which relies on the fact that the phase locked loop can continue running for a time after its reference clock has been disturbed. This provides time for detection of the problem and an orderly system shutdown

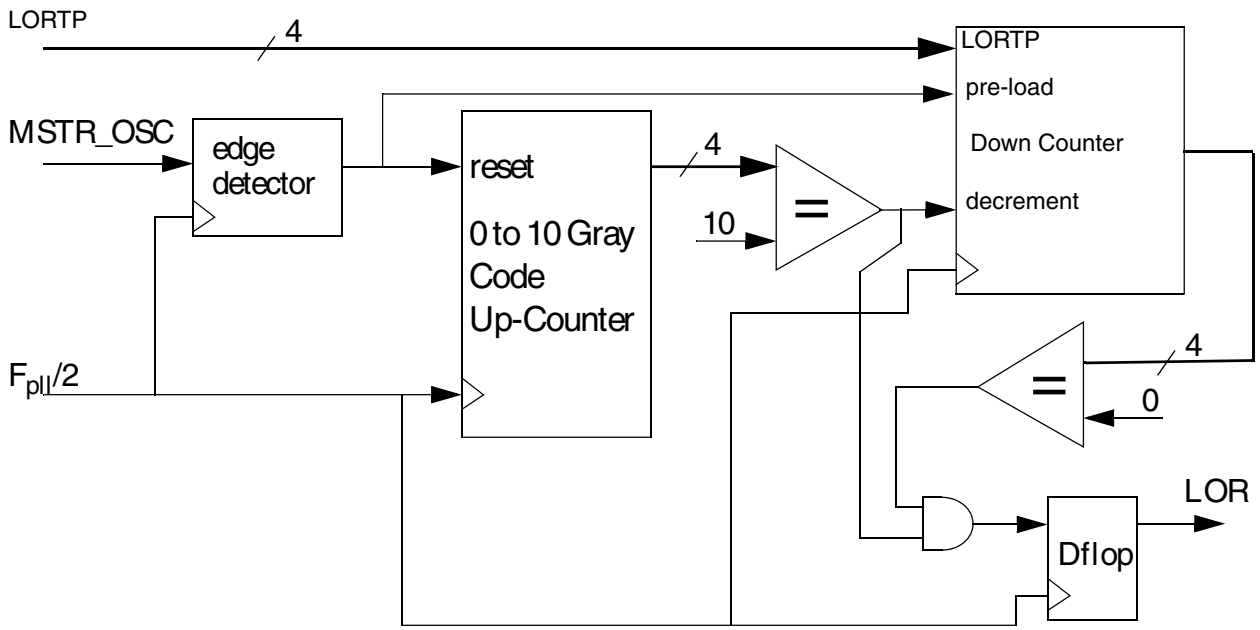


Figure 16-4. Simplified Block Diagram of the Loss Of Reference Clock Detector

16.2.6 Clock Verification

The clock verification logic detects clock source failure or frequency out-of-range, using a known reference clock source.

16.2.6.1 External Clock Checking

The external clock (XOSC) monitor verifies the operation and relative frequency of the external clock source (crystal oscillator or clock input CLK) as compared to the IRC8MHz clock. Verification can be performed any time after both external clock monitor function (OSCTRL2 [MON_ENABLE]) and IRC8M reference clock (OSCTRL1[ROPD]) are enabled. The monitor functions as below:

1. Verification starts by writing a logic one to CLKCHKR[CHK_ENA] which also clears reference clock counter CLKCHKR[REF_CNT] and target clock counter CLKCHKT[TARGET_CNT].
2. The reference clock counter CLKCHKR[REF_CNT] starts to count the IRC8MHz cycle until 128 (0x0080) clocks, then stops.
3. The target clock counter CLKCHKT[TARGET_CNT] starts to count the XOSC cycle until the reference clock counter CLKCHKR[REF_CNT] reaches 128.
4. Compare logic compares the value of CLKCHKR[REF_CNT] and CLKCHKT[TARGET_CNT]. If XOSC clock frequency drops below 680 kHz (typical), STAT[MON_FAILURE] asserts.

NOTE

Recommend to use the trimmed IRC8M clock for XOSC clock monitor.

16.2.6.2 Internal 8MHz RC Oscillator (IRC8M) monitor

The IRC8M monitor verifies whether the relative frequency of the IRC8M clock source operates within a range of nominal frequency as compared to the IRC200k clock. This frequency range is programmable by setting the proper values in IRC8M_MON_THR_HI and IRC8M_MON_THR_LO registers. The default range is ±10% of the IRC8M nominal frequency. When IRC8M monitor function is enabled, every ten IRC200K clock cycles, the value of IRC8M clock counter is compared to values of IRC8M_MON_THR_HI and IRC8M_MON_THR_LO registers. If IRC8M frequency is out of the defined range, STAT[IRC8M_MON_F] asserts.

In fact, this monitor functions to verify whether both IRC8M and IRC200K operate in desired frequency range.

NOTE

Recommend to use both trimmed clocks for monitor.

Both IRC8M and IRC200K must be powered up in order to perform the clock monitoring function.

16.2.7 Clocking

Table 16-3 summarizes the various clock signals in the OCCS module. All clocks are ultimately derived from the oscillator output.

Table 16-3. Clock Summary

Clock	Source	Characteristics
mstr_2x_clk	This module	Primary source for most on-chip clocks. The SIM divides this signal by two to generate the master system frequency.
FEEDBACK	PLL	FEEDBACK pin of the PLL.
$F_{rosc8MHz}$	RC oscillator	Nominally this is 8MHz (2MHz in standby).
$F_{rosc200kHz}$	RC oscillator	Nominally this is 200KHz.
F_{xosc}	Crystal oscillator	Nominally this is 4-16MHz.
F_{pll}	PLL	Output of the PLL.

16.2.8 Resets

The OCCS module is reset by a POR, an external reset, or a COP or software reset.

16.2.9 Interrupts

The interrupts listed in Table 16-4 may be OR'ed into a single processor core interrupt for some chip integrations. Inspect the interrupt table in the chip spec to determine what permutation of these interrupts are actually used.

Table 16-4. Interrupt Summary

Interrupt	Source	Description	Reference
LOLI1	STAT	Lock 1 Interrupt	CTRL register
LOLI0	STAT	Lock 2 Interrupt	CTRL register
LOCI	STAT	Loss of Reference Clock Interrupt	CTRL register
IRC8_MON_FAIL_INT	STAT	IRC8M clock check failed interrupt	CTRL register

If the LOCI interrupt is enabled and the PLL is enabled then the LOCI is permitted to wakeup the system from some STOP modes.

16.3 External signals

16.3.1 External Clock Reference

The 8 MHz internal RC oscillator is enabled in its normal operating mode at 8 MHz at reset. The user has the option of switching to an external clock reference if desired. A number of GPIOs can be programmed (via SIM) to function as the external clock input CLKIN.

16.3.2 Oscillator IO (XTAL, EXTAL)

After reset, the user has the option of switching to the external oscillator. The oscillator inputs can be used to connect an external crystal, ceramic resonator. The EXTAL input pin function is available on the same pad, GPIOC0, as the CLKIN external clock input. Design considerations for the external clock mode of operation are discussed in Modes of Operation.

16.3.3 CLKO - Output Pins

These pins can be programmed to externalize any of the internal clock signals. CLKO functionality exists in SIM. A number of OCCS clocks are available for use on a CLKO pin. The chip has a number of CLKO outputs, so those outputed internal clocks can be compared to each other externally.

CAUTION

There is no defined phase relationship between the signals present on CLKO and their internal counterparts. CLKO is therefore useful for observing internal frequencies, but cannot be used to sequence data onto or off of the chip.

16.4 Memory Map and Register Descriptions

16.4.1 OCCS register descriptions

The address of a register is the sum of a base address and an address offset. The base address is defined at the system level and the address offset is defined at the module level.

16.4.1.1 OCCS memory map

OCCS base address: E2B0h

Offset	Register	Width (In bits)	Access	Reset value
0h	PLL Control Register (CTRL)	16	RW	0010h
1h	PLL Divide-By Register (DIVBY)	16	RW	2071h
2h	OCCS Status Register (STAT)	16	W1C	0210h
4h	Oscillator Control Register 1 (OSCTL1)	16	RW	0400h
5h	Oscillator Control Register 2 (OSCTL2)	16	RW	C100h
6h	External Clock Check Reference (CLKCHKR)	16	RW	0000h
7h	External Clock Check Target (CLKCHKT)	16	RO	0000h
8h	Protection Register (PROT)	16	RW	0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
9h	Oscillator Control Register 3 (OSCTL3)	16	RW	0240h
Ah	Oscillator Control Register 4 (OSCTL4)	16	RW	0240h
Ch	IRC8M High Threshold (IRC8M_MON_THR_HI)	16	RW	01B8h
Dh	IRC8M Low Threshold (IRC8M_MON_THR_LO)	16	RW	0168h

16.4.1.2 PLL Control Register (CTRL)

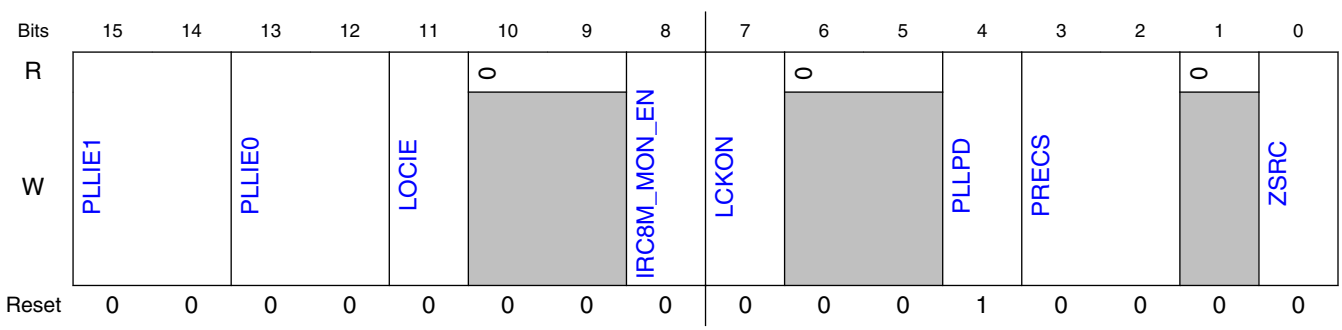
16.4.1.2.1 Offset

Register	Offset
CTRL	0h

16.4.1.2.2 Function

PLL control bits.

16.4.1.2.3 Diagram



16.4.1.2.4 Fields

Field	Function
15-14	PLL Interrupt Enable 1
PLLIE1	An optional interrupt can be generated when the PLL lock status bit (LCK1) in the OCCS Status Register (STAT) changes. 00b - Disable interrupt. 01b - Enable interrupt on any rising edge of LCK1. 10b - Enable interrupt on falling edge of LCK1.

Table continues on the next page...

Field	Function
	11b - Enable interrupt on any edge change of LCK1.
13-12 PLLIE0	PLL Interrupt Enable 0 An optional interrupt can be generated if the PLL lock status bit (LCK0) in the OCCS Status Register (STAT) changes. 00b - Disable interrupt. 01b - Enable interrupt on any rising edge of LCK0. 10b - Enable interrupt on falling edge of LCK0. 11b - Enable interrupt on any edge change of LCK0.
11 LOCIE	Loss of Reference Clock Interrupt Enable The loss of reference clock circuit monitors the output of the on-chip oscillator circuit used as PLL input reference clock. In the event of loss of reference clock, an optional interrupt can be generated. An optional interrupt can be generated if the oscillator circuit output clock is lost. 0b - Interrupt disabled. 1b - Interrupt enabled.
10-9 —	RESERVED
8 IRC8M_MON_EN	IRC8M Monitor Enable Enables IRC8M clock monitor and IRC8M check failed interrupt. **NOTE:** When IRC8M monitor is enabled, IRC200K must be powered up by clearing OSCTL2[ROPD200K]. 0b - IRC8M monitor disabled 1b - IRC8M monitor enabled
7 LCKON	Lock Detector On 0b - Lock detector disabled 1b - Lock detector enabled
6-5 —	RESERVED
4 PLLPD	PLL Power Down The PLL can be turned off by setting the PLLPD bit. There is a delay of four IP bus clocks between changing the bit and signaling the PLL. When the PLL is powered down, the gear shifting logic automatically switches to ZSRC = 0 to prevent a loss of the reference clock to the core. 0b - PLL enabled 1b - PLL powered down
3-2 PRECS	Prescaler Clock Select This bit selects between, on one hand, the external clock source or oscillator and, on the other hand, the internal RC oscillator. **NOTE:** Before switching to a new clock source, you must enable the new source. The RC oscillators are configured entirely within the OCCS. The external reference, CLKIN or external oscillator, requires configuration of the GPIO and SIM to configure the related external pads for the appropriate function as well as configuration of the OCCS itself. 00b - Internal 8 MHz RC oscillator is selected (reset value) 01b - External reference is selected 10b - 200 kHz RC oscillator is selected 11b - Reserved
1 —	RESERVED

Table continues on the next page...

Field	Function
0 ZSRC	CLOCK Source This field determines the mstr_2x_clk source to the SIM, which generates divided-down versions of this signal for use by memories and the IP Bus. If PLLPD is set, ZSRC is automatically set to 0 to prevent a loss of the reference clock to the core. **NOTE:** Before switching to a new clock source, you must enable the new source. The PLL should be on, configured, and locked before switching to it. For extra assurance in cases where the PLL may be stressed, confirm that the PLL remains locked for a period of time before switching to it. 0b - MSTR_OSC 1b - PLL output divided by 2

16.4.1.3 PLL Divide-By Register (DIVBY)

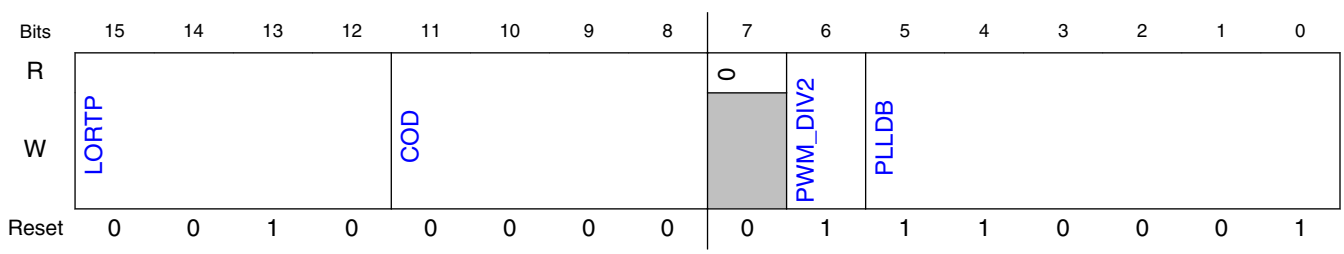
16.4.1.3.1 Offset

Register	Offset
DIVBY	1h

16.4.1.3.2 Function

Clock divider.

16.4.1.3.3 Diagram



16.4.1.3.4 Fields

Field	Function
15-12 LORTP	Loss of Reference Clock Trip Point These bits control the amount of time required for the loss of reference clock interrupt to be generated. This failure detection time is `((LORTP + 1) x 10) x (reference clock period) / (PLL Multiplier / 2)`. The PLL Multiplier is set by the PLLDB register. The recommendation is to keep the value of LORTP >= 0010b

Table continues on the next page...

Field	Function
11-8 COD	Clock Output Divide or Postscaler The PLL output clock can be divided down by a 4-bit postscaler. The input of the postscaler is a selectable clock source for the DSP core as determined by the ZSRC bit in the control register. The output of the postscaler is guaranteed to be glitch free, even when the COD field has been changed. 0000b - Divide clock output by 1. 0001b - Divide clock output by 2. 0010b - Divide clock output by 4. 0011b - Divide clock output by 8. 0100b - Divide clock output by 16. 0101b - Divide clock output by 32. 0110b - Divide clock output by 64. 0111b - Divide clock output by 128. 1xxxb - Divide clock output by 256.
7 —	RESERVED
6 PWM_DIV2	PWM_DIV2 This bit decides the 200MHz clock source to PWM nano edge 0b - Raw PLL output selected as PWM_2X clock if PLL output is 200MHz 1b - PLL DIV2 Clock Selected as PWM 2X clock if PLL output is 400 MHz. This is recommended setting.
5-0 PLLDB	PLL Divide By The output frequency of the PLL is controlled, in part, by the PLLDB[5:0] field. The value written to this field, plus one, is used by the PLL to directly multiply the input frequency and present it at its output. For example, if the input frequency is 8 MHz and the PLLDB[5:0] field is set to 49 (the default), then the PLL output frequency is 400 MHz. PLLDB settings should be limited such that the output frequency of the PLL does not exceed the maximum frequency specified in the device datasheet. The frequency of the primary output clock to the SIM (mstr_2x_clk) depends on the setting of the ZSRC field, which selects the primary output clock source, and the COD field, which configures the postscaler. Before the divide-by value is changed, the core clock must first be switched to the MSTR_OSC clock. **NOTE:** Upon writing to the PLL Divide By register, the loss of reference detector circuit is reset.

16.4.1.4 OCCS Status Register (STAT)

16.4.1.4.1 Offset

Register	Offset
STAT	2h

16.4.1.4.2 Function

A PLL interrupt is generated if any of the LOLI or LOCI bits are set and the corresponding interrupt enable is set in the control register.

16.4.1.4.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LOLI1	LOLI0	LOCI	IRC8M_MON_FAIL_INT	0		IRC8M_OK	MON_FAILURE	OSC_OK	LCK1	LCK0	PLLPDN	0		ZSRCS	
W	W1C	W1C	W1C	W1C												
Reset	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0

16.4.1.4.4 Fields

Field	Function
15 LOLI1	PLL Lock or Loss of Lock Interrupt 1 LOLI1 shows the status of the lock detector state from the LCK1 circuit. This bit is cleared by writing a one to it. This bit will not be set (by the hardware) if the corresponding CTRL[PLLIE1] bit is cleared (set to zero). 0b - No lock or loss of lock event has occurred. 1b - PLL lock status based on PLLIE1.
14 LOLI0	PLL Lock or Loss of Lock Interrupt 0 LOLI0 shows the status of the lock detector state from LCK0 circuit. This bit is cleared by writing a one to it. This bit will not be set (by the hardware) if the corresponding CTRL[PLLIE0] bit is cleared (set to zero). 0b - No lock or loss of lock event has occurred. 1b - PLL lock status based on PLLIE0.
13 LOCI	Loss of Reference Clock Interrupt LOCI shows the status of the reference clock detection circuit. This bit is cleared by writing a one to it. 0b - Oscillator clock normal. 1b - Loss of oscillator clock detected.
12 IRC8M_MON_FAIL_INT	IRC8M Monitor Failed Interrupt Monitors ICR8M clock frequency. When the frequency is out of range (default ±10%), then it shows the failed interrupt flag. This bit is cleared by writing a logic one to it. 0b - IRC8M clock is normal. 1b - IRC8M clock is out of range.
11-10 —	RESERVED
9	Enable indicator from internal RC 8MHz clock

Table continues on the next page...

Field	Function
IRC8M_OK	0b - Internal RC 8MHz(standby 2MHz) clock is stopped(when register bit CLK_STOP is asserted) or disabled. 1b - Internal RC 8MHz(standby 2MHz) clock is enabled.
8 MON_FAILURE	XOSC Clock Monitor Failure Indicator XOSC Clock Monitor Failure Indicator. If MON_ENABLE is enabled, this flag indicates XOSC clock frequency drops below 680 kHz(Typical). If MON_ENABLE is disabled, no failure is indicated. 0b - No clock failure, or XOSC clock monitor is disabled. 1b - XOSC clock frequency drops below 680 kHz(Typical) when clock monitor is enabled.
7 OSC_OK	OSC_OK Indicator from XOSC 0b - Oscillator clock is still not stable, or XOSC is disabled. 1b - Oscillator clock is stable after crystal oscillator startup.
6 LCK1	PLL Lock 1 Status 0b - PLL is unlocked. 1b - PLL is locked (fine).
5 LCK0	PLL Lock 0 Status 0b - PLL is unlocked. 1b - PLL is locked (coarse).
4 PLLPDN	PLL Power Down PLL power down status is delayed by four IPbus clocks from the PLLPD bit in the control register. 0b - PLL not powered down. 1b - PLL powered down.
3-2 —	RESERVED
1-0 ZSRCS	CLOCK Source Status ZSRCS indicates the current mstr_2x_clk clock source. Because the synchronizing circuit switches the system clock source, ZSRCS takes more than one IP bus clock to indicate the new selection. 00b - MSTR_OSC 01b - PLL output divided by 2 1xb - Synchronization in progress

16.4.1.5 Oscillator Control Register 1 (OSCTL1)

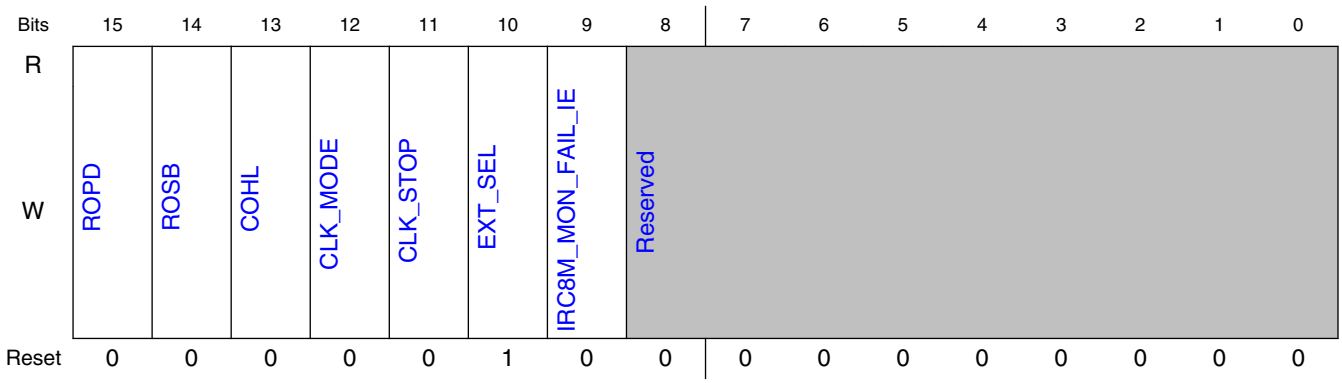
16.4.1.5.1 Offset

Register	Offset
OSCTL1	4h

16.4.1.5.2 Function

This register controls aspects of both the internal RC oscillator and the crystal/resonator oscillator.

16.4.1.5.3 Diagram



16.4.1.5.4 Fields

Field	Function
15 ROPD	8 MHz RC Oscillator Power Down This bit powers down the 8 MHz RC oscillator. To prevent a loss of clock to the core or the PLL, this bit should never be asserted while this clock source is selected by the PRECS field in the control register. 0b - RC oscillator enabled. 1b - RC oscillator powered down.
14 ROSB	8 MHz RC Oscillator Standby This bit controls the power usage and gross frequency of the 8 MHz RC oscillator. It is reset to the more accurate but higher power state. 0b - Normal mode. The RC oscillator output frequency is 8 MHz. 1b - Standby mode. The RC oscillator output frequency is reduced to 2MHz . The PLL should be disabled in this mode and MSTR_OSC should be selected as the output clock.
13 COHL	Crystal Oscillator High/Low Power Level This bit controls the power usage of the crystal oscillator. It is reset to the high power state. The low power mode should be selected to operate the crystal oscillator in Loop Controlled Pierce (LCP) mode. The high power mode should be selected to operate the crystal oscillator in Full Swing Pierce (FSP) mode or in the oscillator's external clock bypass mode. 0b - High power mode. 1b - Low power mode.
12 CLK_MODE	Crystal Oscillator Clock Mode When crystal oscillator is powered on (COPD=0), this bit controls the operating mode of the crystal oscillator. When CLK_MODE is set to 1, COHL should be set to 0. External Clock Bypass Mode is not supported. **NOTE:** If the crystal oscillator is turned off and then turned on again, the clock should not be switched back to the oscillator until after the crystal has had time to stabilize. See the crystal data sheet to determine this time duration. 0b - Crystal oscillator is in FSP mode (COHL=0) or LCP mode (COHL=1), when COPD=0. 1b - External clock bypass mode. This enables the crystal oscillator's external clock bypass mode and allows an external clock source on the EXTAL input of the oscillator to propagate directly to the oscillator's clock output.

Table continues on the next page...

Field	Function
11 CLK_STOP	Internal 8MHz RC Clock Stop When internal 8MHz RC clock is powered on (ROPD=0), this bit controls the output of the 8MHz clock. 0b - Internal 8MHz(standby 2MHz) outputs the clock. 1b - Internal 8MHz(standby 2MHz) stops the clock output. When CLK_STOP is writen to 0, the clock outputs immediately without startup time.
10 EXT_SEL	External Clock In Select This bit selects the source of the external clock input. PRECS should be 0 before changing the value of EXT_SEL to avoid glitches on the system clock. 0b - Use the output of the crystal oscillator as the external clock input. 1b - Use CLKIN as the external clock input.
9 IRC8M_MON_FAIL_IE	IRC8M Monitor Failed Interrupt Enable Enables the IRC8M Monitor Failed Interrupt function. 0b - Disables. 1b - Enables.
8-0 —	Reserved

16.4.1.6 Oscillator Control Register 2 (OSCTL2)

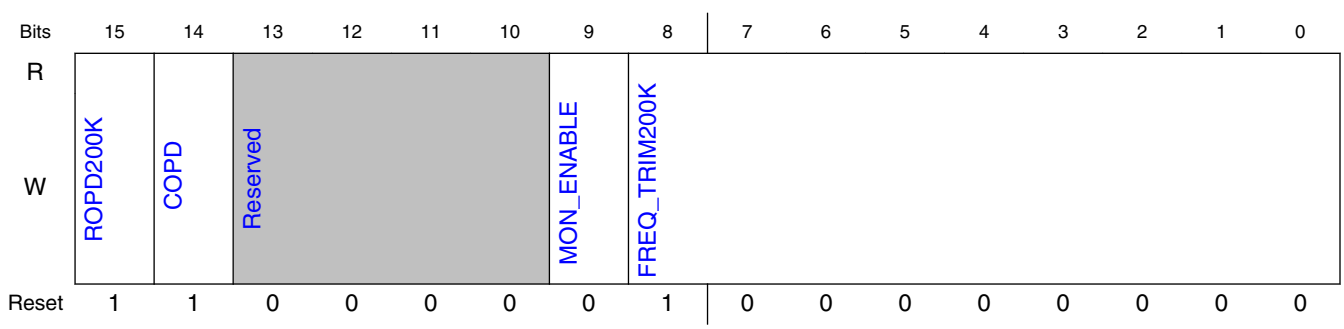
16.4.1.6.1 Offset

Register	Offset
OSCTL2	5h

16.4.1.6.2 Function

This register controls aspects of both the internal RC oscillator and the crystal/resonator oscillator.

16.4.1.6.3 Diagram



16.4.1.6.4 Fields

Field	Function
15 ROPD200K	200 kHz RC Oscillator Power Down This bit powers down the 200 kHz internal RC oscillator. To prevent a loss of clock to the core or the PLL, this bit should never be asserted while this clock source is selected by the PRECS field in the control register. 0b - IRC200K is powered on. 1b - IRC200K is powered down.
14 COPD	Crystal Oscillator Power Down This bit powers down the external crystal oscillator. To prevent a loss of clock to the core or the PLL, this bit should never be asserted while this clock source is selected by the PRECS field in the control register. 0b - Crystal oscillator is powered on. 1b - Crystal oscillator is powered down.
13-10 —	Reserved
9 MON_ENABLE	XOSC Clock Monitor Enable Control This bit enables the clock monitor functionality of the XOSC. The clock monitor consists of a time-out circuit which gets reset with every XOSC clock edge. If the XOSC clock frequency is lower than the clock monitor failure assert frequency f_{CMFA} (typical value is 680 kHz), the flag gets asserted high. 0b - XOSC Clock Monitor is disabled. 1b - XOSC Clock Monitor is enabled.
8-0 FREQ_TRIM200K	200 kHz Internal RC Oscillator Frequency Trim These bits correct part-specific variation in oscillator frequency. This control adjusts the part's average frequency over temperature up and down for the purpose of aligning it to 200 kHz. By testing the frequency of the internal clock and incrementing or decrementing this factor accordingly, the accuracy of the internal clock can be improved. A reset sets these bits to $100, centering the range of possible adjustment.

16.4.1.7 External Clock Check Reference (CLKCHKR)

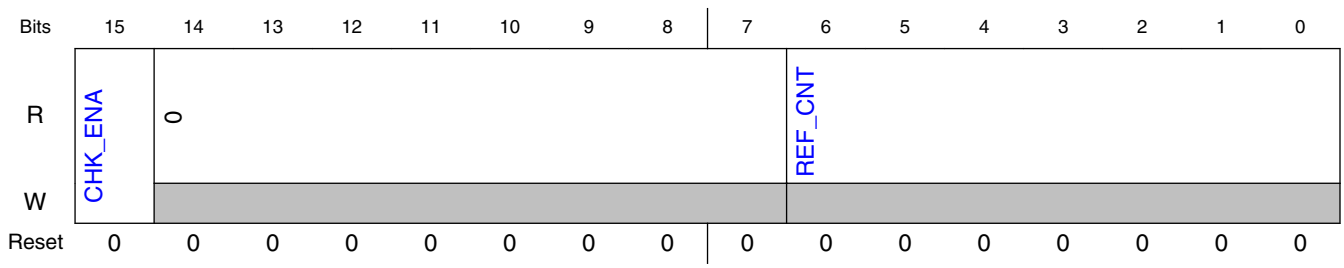
16.4.1.7.1 Offset

Register	Offset
CLKCHKR	6h

16.4.1.7.2 Function

Along with the External Clock Check Target register, this register verifies the operation and relative frequency of the external clock source (target) as compared to the IRC8M clock . Verification can be performed any time the target and reference clocks are enabled; however, the greatest benefit is the ability to verify the external clock source prior to selecting it with the PRECS field.

16.4.1.7.3 Diagram



16.4.1.7.4 Fields

Field	Function
15 CHK_ENA	Check Enable This bit starts and stops the clock checking function. Allow enough time after the CLK_ENA bit is cleared to allow for two IRC8M clock periods before attempting to start another verification cycle. 0b - Writing a low while the clock checking operation is in progress stops the check in its current state. Reading a low after a check has been started indicates that the check operation is complete and the final values are valid in the REF_CNT and TARGET_CNT fields. 1b - Writing a one clears the REF_CNT and TARGET_CNT fields and starts the clock checking function. The CLK_ENA bit remains high while the operation is in progress.
14-7 —	RESERVED
6-0 REF_CNT	Reference Count This count is initialized to zero on the positive transition of CHK_ENA. The test is terminated when REF_CNT equals to 0x0080, which means 0x80 IRC8M clock cycles have been counted. **NOTE:** This counter value is not synchronized to the bus clock, and any value read while CHK_ENA is high should not be considered accurate.

16.4.1.8 External Clock Check Target (CLKCHKT)

16.4.1.8.1 Offset

Register	Offset
CLKCHKT	7h

16.4.1.8.2 Function

Along with the External Clock Check Reference register, this register verifies the operation and relative frequency of the external clock source (target) as compared to the IRC8M clock . Verification can be performed any time the target and reference clocks are enabled; however, the greatest benefit is the ability to verify the external clock source prior to selecting it with the PRECS field.

16.4.1.8.3 Diagram

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

16.4.1.8.4 Fields

Field	Function
15-11 —	RESERVED
10-0 TARGET_CNT	CLKCHKT Target Count Number of external clock cycles that have been counted. This count is terminated when TARGET_CNT equals to 0x7FF, which means (2^11-1) clock cycles have been counted. **NOTE:** This counter value is not synchronized to the bus clock, and any value read while CHK_ENA is high should not be considered accurate. Its capacity is sufficient for verifying external clock (from CLKIN or Crystal Oscillator, via OCCS_OSCTL1[EXT_SEL]) upto 8 times the reference clock with sufficient room for overflow.

16.4.1.9 Protection Register (PROT)

16.4.1.9.1 Offset

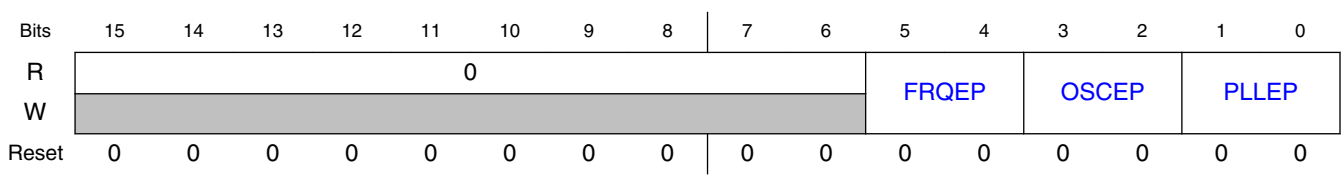
Register	Offset
PROT	8h

16.4.1.9.2 Function

This register provides features for runaway code protection of safety-critical register fields. By choosing an appropriate subset of protection registers, you can define the trade-off between power management and protection of the OCCS operating configuration.

Flexibility is provided so that write-protection control values may themselves be optionally locked (write protected). To this end, protection controls in this register have two bit values. The right bit determines the setting of the control, and the left bit determines whether the value is locked. When a protection control is set to a locked value, it can only be altered by a chip reset which restores its default non-locked value. While a protection control remains set to non-locked values, it can be re-written to any new value.

16.4.1.9.3 Diagram



16.4.1.9.4 Fields

Field	Function
15-6 —	RESERVED
5-4 FRQEP	Frequency Enable Protection Enables write protection of the COD and ZSRC bitfields. 00b - Write protection off (default). 01b - Write protection on. 10b - Write protection off and locked until chip reset. 11b - Write protection on and locked until chip reset.
3-2 OSCEP	Oscillator Enable Protection Enables write protection of the OSCTL1, OSCTL2 , OSCTL3, OSCTL4 registers, and PRECS bitfield. 00b - Write protection off (default). 01b - Write protection on.

Table continues on the next page...

Field	Function
	10b - Write protection off and locked until chip reset. 11b - Write protection on and locked until chip reset.
1-0 PLLEP	PLL Enable Protection Enables write protection of the PLLPD, LOCIE, LORTP, and PLLDB bitfields. When these registers are write protected (PLLPD is 0 and LOCIE is 1), the loss of reference detector cannot be disabled. 00b - Write protection off (default). 01b - Write protection on. 10b - Write protection off and locked until chip reset. 11b - Write protection on and locked until chip reset.

16.4.1.10 Oscillator Control Register 3 (OSCTL3)

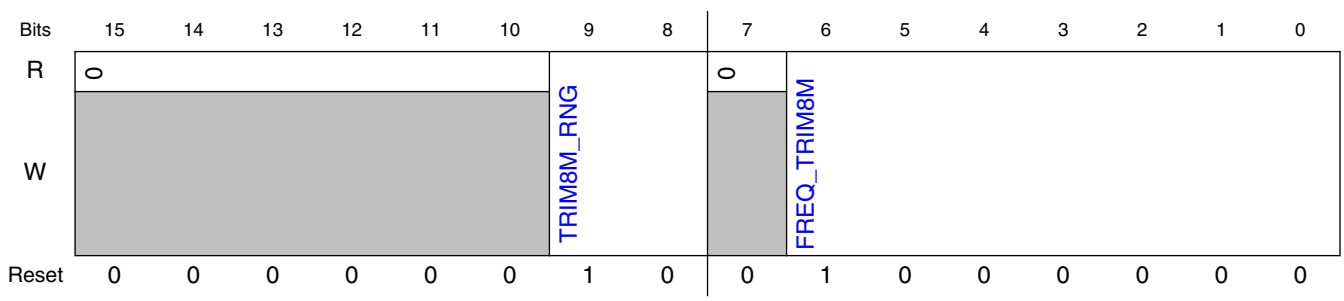
16.4.1.10.1 Offset

Register	Offset
OSCTL3	9h

16.4.1.10.2 Function

This is for the 8MHz RC clock frequency trim. It is loaded from flash memory during reset.

16.4.1.10.3 Diagram



16.4.1.10.4 Fields

Field	Function
15-10	Reserved

Table continues on the next page...

Field	Function
—	
9-8 TRIM8M_RNG	Internal RC Oscillator 8MHz clock Trim Range Enlarge
7 —	Reserved
6-0 FREQ_TRIM8M	Internal RC Oscillator 8MHz clock Trim Code

16.4.1.11 Oscillator Control Register 4 (OSCTL4)

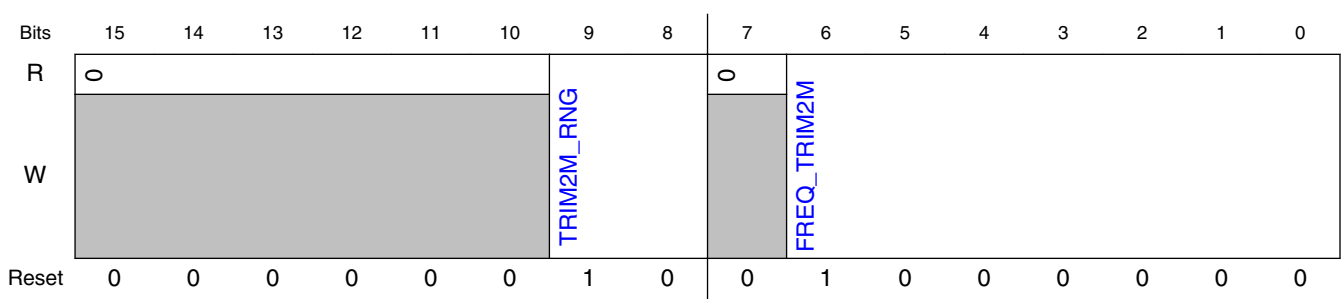
16.4.1.11.1 Offset

Register	Offset
OSCTL4	Ah

16.4.1.11.2 Function

This is for the 2MHz RC clock frequency trim. It is loaded from flash memory during reset.

16.4.1.11.3 Diagram



16.4.1.11.4 Fields

Field	Function
15-10 —	Reserved

Table continues on the next page...

Field	Function
9-8 TRIM2M_RNG	Internal RC Oscillator 2MHz clock Trim Range Enlarge
7 —	Reserved
6-0 FREQ_TRIM2M	Internal RC Oscillator 2MHz clock Trim Code

16.4.1.12 IRC8M High Threshold (IRC8M_MON_THR_HI)

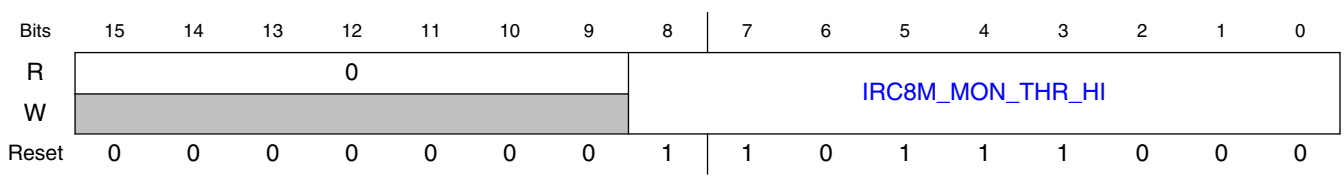
16.4.1.12.1 Offset

Register	Offset
IRC8M_MON_THR_HI	Ch

16.4.1.12.2 Function

This register config the high threshold which prepare to compare with IRC8M counter. We count the IRC8M clock cycles during 10 IRC200K clock cycles, then check if [8Mhz - (200KHz x 40)] has more than +/-10% variation (default).

16.4.1.12.3 Diagram



16.4.1.12.4 Fields

Field	Function
15-9 —	Reserved
8-0 IRC8M_MON_THR_HI	IRC8M Monitor High Threshold This register config the high threshold which prepare to compare with IRC8M counter during 10 200K clock cycles.

16.4.1.13 IRC8M Low Threshold (IRC8M_MON_THR_LO)

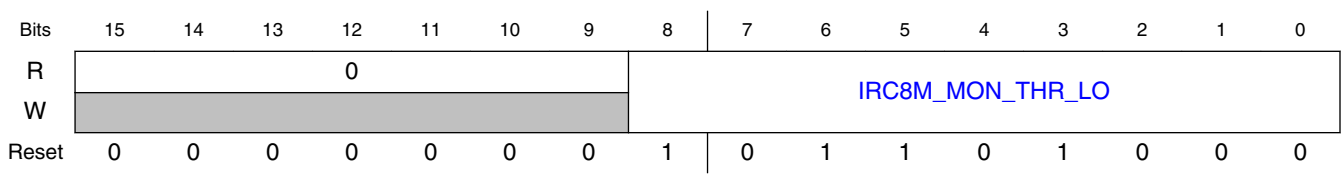
16.4.1.13.1 Offset

Register	Offset
IRC8M_MON_THR_LO	Dh

16.4.1.13.2 Function

This register config the low threshold which prepare to compare with IRC8M counter. We count the IRC8M clock cycles during 10 IRC200K clock cycles, then check if [8Mhz - (200KHz x 40)] has more than +/-10% variation (default).

16.4.1.13.3 Diagram



16.4.1.13.4 Fields

Field	Function
15-9 —	Reserved
8-0 IRC8M_MON_THR_LO	IRC8M Monitor Low Threshold This register config the low threshold which prepare to compare with IRC8M counter during 10 200K clock cycles.

Chapter 17
Flash Memory Controller (FMC)

17.1 Introduction

The Flash Memory Controller (FMC) is a memory acceleration unit that provides:

- an interface between the device and the program flash memory.
- buffers that can accelerate flash memory transfers.

17.1.1 Overview

The Flash Memory Controller manages the interface between the device and the flash memory. The FMC receives status information detailing the configuration of the memory and uses this information to ensure a proper interface. The following table shows the supported 8-bit, 16-bit, and 32-bit read/write operations.

Flash memory type	Read	Write
Program flash memory	x	—[1]

1. A write operation to program flash memory results in a bus error.

In addition, the FMC provides three separate mechanisms for accelerating the interface between the device and the flash memory. A 32-bit speculation buffer can prefetch the next 32-bit flash memory location, and both a 4-way, 4-set cache and a single-entry 32-bit buffer can store previously accessed flash memory data for quick access times.

17.1.2 Features

The FMC's features include:

- Interface between the device and the flash memory:
  - 8-bit, 16-bit, and 32-bit read operations to program flash memory.

    - Read accesses to consecutive 32-bit spaces in memory return the second read data with no wait states. The memory returns 32 bits via the 32-bit bus access.
    - For each crossbar master, access protection setting options for:
        - no access
        - read-only access
- Acceleration of data transfer from program flash memory to the device:
    - 32-bit prefetch speculation buffer with controls for instruction/data access per master
    - 4-way, 4-set, 32-bit line size cache for a total of sixteen 32-bit entries with controls for replacement algorithm and lock per way
    - Single-entry buffer with enable
    - Invalidation control for the speculation buffer and the single-entry buffer

17.2 Modes of operation

The FMC only operates when the device accesses the flash memory.

For any device power mode where the flash memory cannot be accessed, the FMC is disabled.

17.3 External signal description

The FMC has no external signals.

17.4 Functional description

The FMC is a flash acceleration unit with flexible buffers for user configuration. Besides managing the interface between the device and the flash memory, the FMC can be used to restrict access from crossbar switch masters and customize the cache and buffers to provide single-cycle system-clock data-access times. Whenever a hit occurs for the prefetch speculation buffer, the cache, or the single-entry buffer, the requested data is transferred within a single system clock.

Upon system reset, the FMC is configured to provide a significant level of buffering for transfers from the flash memory:

- Prefetch support for data and instructions is enabled for crossbar masters 0, 1, 2.
- The cache is configured for least recently used (LRU) replacement for all four ways.
- The cache is configured for data or instruction replacement.

- The single-entry buffer is enabled.
- The Master 0, 1, and 2 Access Protection fields of PFB0CR are set to 11b. The user must change each of these field values to 00b or 01b.

Though the default configuration provides a high degree of flash acceleration, advanced users may desire to customize the FMC buffer configurations to maximize throughput for their use cases. When reconfiguring the FMC for custom use cases, do not program the FMC's control registers while the flash memory is being accessed. Instead, change the control registers with a routine executing from RAM in supervisor mode.

The FMC's cache and buffering controls within PFB0CR allow the tuning of resources to suit particular applications' needs. The cache and two buffers are each controlled individually. The register controls enable buffering and prefetching per access type (instruction fetch or data reference). The cache also supports three types of LRU replacement algorithms:

- LRU per set across all four ways,
- LRU with ways [0-1] for instruction fetches and ways [2-3] for data fetches, and
- LRU with ways [0-2] for instruction fetches and way [3] for data fetches.

As an application example: if both instruction fetches and data references are accessing the flash memory, control is available to send instruction fetches, data references, or both to the cache or the single-entry buffer. Likewise, speculation can be enabled or disabled for either type of access. If both instruction fetches and data references are cached, the cache's way resources may be divided in several ways between the instruction fetches and data references.

17.5 Memory map and register descriptions

17.5.1 FMC register descriptions

The programming model consists of the FMC control registers and the program visible cache (data and tag/valid entries).

NOTE

Program the registers only while the flash controller is idle (for example, execute from RAM). Changing configuration settings while a flash access is in progress can lead to non-deterministic behavior.

Table 17-1. FMC register access

Registers	Read access		Write access	
	Mode	Length	Mode	Length
Control registers: PFAPR, PFB0CR	Supervisor (privileged) mode or user mode	32 bits	Supervisor (privileged) mode only	32 bits
Cache registers	Supervisor (privileged) mode or user mode	32 bits	Supervisor (privileged) mode only	32 bits

NOTE

Accesses to unimplemented registers within the FMC's address space return a bus error.

The cache entries, both data and tag/valid, can be read at any time.

NOTE

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

The cache is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. The following table elaborates on the tag/valid and data entries.

Table 17-2. Program visible cache registers

Cache storage	Based at offset	Contents of 32-bit read	Nomenclature	Nomenclature example
Tag	080h	12'h0, tag[19:4], 3'h0, valid	In TAGVDWxSy, x denotes the way, and y denotes the set.	TAGVDW1S1 is the 13-bit tag and 1-bit valid for cache entry way 1, set 1.
Data	100h	Data word	In DATAWxSy, x denotes the way, and y denotes the set.	DATAW1S1 represents bits [31:0] of data entry way 1, set 1.

NOTE

In the preceding table and in the remainder of the register information, offset and address data appears in terms of word (16-bit) addressing.

17.5.1.1 FMC memory map

FMC base address: DE00h

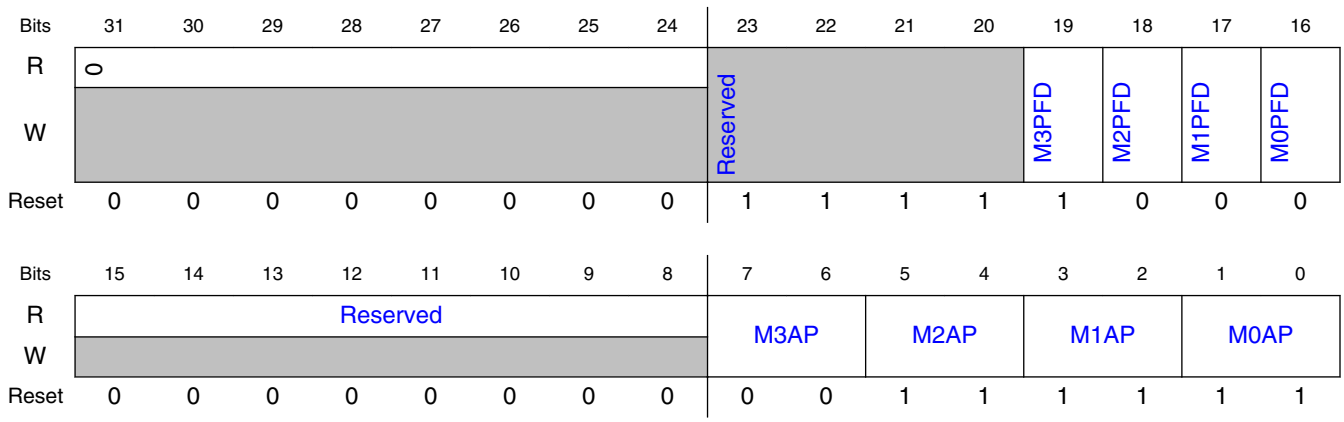
Offset	Register	Width (In bits)	Access	Reset value
0h	Flash Access Protection Register (PFAPR)	32	RW	00F8_003Fh
2h	Flash Control Register (PFB0CR)	32	RW	3000_001Fh
80h	Cache Tag Storage (TAGVDW0S0)	32	RW	0000_0000h
82h	Cache Tag Storage (TAGVDW0S1)	32	RW	0000_0000h
84h	Cache Tag Storage (TAGVDW0S2)	32	RW	0000_0000h
86h	Cache Tag Storage (TAGVDW0S3)	32	RW	0000_0000h
88h	Cache Tag Storage (TAGVDW1S0)	32	RW	0000_0000h
8Ah	Cache Tag Storage (TAGVDW1S1)	32	RW	0000_0000h
8Ch	Cache Tag Storage (TAGVDW1S2)	32	RW	0000_0000h
8Eh	Cache Tag Storage (TAGVDW1S3)	32	RW	0000_0000h
90h	Cache Tag Storage (TAGVDW2S0)	32	RW	0000_0000h
92h	Cache Tag Storage (TAGVDW2S1)	32	RW	0000_0000h
94h	Cache Tag Storage (TAGVDW2S2)	32	RW	0000_0000h
96h	Cache Tag Storage (TAGVDW2S3)	32	RW	0000_0000h
98h	Cache Tag Storage (TAGVDW3S0)	32	RW	0000_0000h
9Ah	Cache Tag Storage (TAGVDW3S1)	32	RW	0000_0000h
9Ch	Cache Tag Storage (TAGVDW3S2)	32	RW	0000_0000h
9Eh	Cache Tag Storage (TAGVDW3S3)	32	RW	0000_0000h
100h	Cache Data Storage (DATAW0S0)	32	RW	0000_0000h
102h	Cache Data Storage (DATAW0S1)	32	RW	0000_0000h
104h	Cache Data Storage (DATAW0S2)	32	RW	0000_0000h
106h	Cache Data Storage (DATAW0S3)	32	RW	0000_0000h
108h	Cache Data Storage (DATAW1S0)	32	RW	0000_0000h
10Ah	Cache Data Storage (DATAW1S1)	32	RW	0000_0000h
10Ch	Cache Data Storage (DATAW1S2)	32	RW	0000_0000h
10Eh	Cache Data Storage (DATAW1S3)	32	RW	0000_0000h
110h	Cache Data Storage (DATAW2S0)	32	RW	0000_0000h
112h	Cache Data Storage (DATAW2S1)	32	RW	0000_0000h
114h	Cache Data Storage (DATAW2S2)	32	RW	0000_0000h
116h	Cache Data Storage (DATAW2S3)	32	RW	0000_0000h
118h	Cache Data Storage (DATAW3S0)	32	RW	0000_0000h
11Ah	Cache Data Storage (DATAW3S1)	32	RW	0000_0000h
11Ch	Cache Data Storage (DATAW3S2)	32	RW	0000_0000h
11Eh	Cache Data Storage (DATAW3S3)	32	RW	0000_0000h

17.5.1.2 Flash Access Protection Register (PFAPR)

17.5.1.2.1 Offset

Register	Offset
PFAPR	0h

17.5.1.2.2 Diagram



17.5.1.2.3 Fields

Field	Function
31-24 —	Reserved
23-20 —	Reserved This read-only bitfield is reserved and is reset to 4'hF. Do not write to this bitfield or indeterminate results will occur.
19 M3PFD	Master 3 Prefetch Disable These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits. 0b - Prefetching for this master is enabled. 1b - Prefetching for this master is disabled.
18 M2PFD	Master 2 Prefetch Disable These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits. 0b - Prefetching for this master is enabled. 1b - Prefetching for this master is disabled.
17 M1PFD	Master 1 Prefetch Disable These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.

Table continues on the next page...

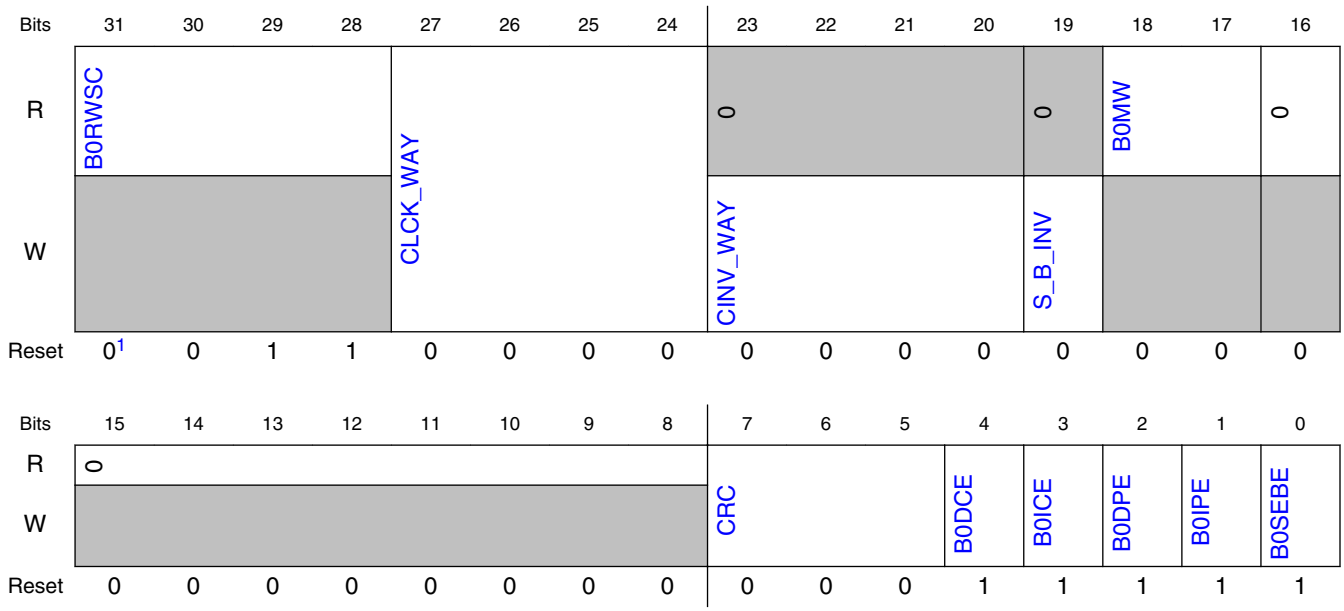
Field	Function
	0b - Prefetching for this master is enabled. 1b - Prefetching for this master is disabled.
16 M0PFD	Master 0 Prefetch Disable These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits. 0b - Prefetching for this master is enabled. 1b - Prefetching for this master is disabled.
15-8 —	Reserved This read-only bitfield is reserved and is reset to zero. Do not write to this bitfield or indeterminate results will occur.
7-6 M3AP	Master 3 Access Protection This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master. 00b - No access may be performed by this master 01b - Only read accesses may be performed by this master 10b - Reserved 11b - Reserved
5-4 M2AP	Master 2 Access Protection This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master. 00b - No access may be performed by this master 01b - Only read accesses may be performed by this master 10b - Reserved 11b - Reserved
3-2 M1AP	Master 1 Access Protection This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master. 00b - No access may be performed by this master 01b - Only read accesses may be performed by this master 10b - Reserved 11b - Reserved
1-0 M0AP	Master 0 Access Protection This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master. 00b - No access may be performed by this master 01b - Only read accesses may be performed by this master 10b - Reserved 11b - Reserved

17.5.1.3 Flash Control Register (PFB0CR)

17.5.1.3.1 Offset

Register	Offset
PFB0CR	2h

17.5.1.3.2 Diagram



1. When the device is operating in normal mode, the reset value of this field is 1h. When the device is operating in fast mode, the reset value of this field is 3h.

17.5.1.3.3 Fields

Field	Function
31-28 B0RWSC	Read Wait State Control This read-only field defines the number of wait states required to access the flash memory. The relationship between the read access time of the flash array (expressed in system clock cycles) and RWSC is defined as: Access time of flash array [system clocks] = RWSC + 1 The FMC automatically calculates this value based on the ratio of the system clock speed to the flash clock speed. For example, when this ratio is 4:1, the field's value is 3h, and when this ratio is 2:1, the field's value is 1h.
27-24 CLCK_WAY	Cache Lock Way x These bits determine if the given cache way is locked such that its contents will not be displaced by future misses. The bit setting definitions are for each bit in the field. 0000b - Cache way is unlocked and may be displaced 0001b - Cache way is locked and its contents are not displaced
23-20 CINV_WAY	Cache Invalidate Way x These bits determine if the given cache way is to be invalidated (cleared). When a bit within this field is written, the corresponding cache way is immediately invalidated: the way's tag, data, and valid contents are cleared. This field always reads as zero.

Table continues on the next page...

Field	Function
	Cache invalidation takes precedence over locking. The cache is invalidated by system reset. System software is required to maintain memory coherency when any segment of the flash memory is programmed or erased. Accordingly, cache invalidations must occur after a programming or erase event is completed and before the new memory image is accessed. The bit setting definitions are for each bit in the field. 0000b - No cache way invalidation for the corresponding cache 0001b - Invalidate cache way for the corresponding cache: clear the tag, data, and vld bits of ways selected
19 S_B_INV	Invalidate Prefetch Speculation Buffer This bit determines if the FMC's prefetch speculation buffer and the single entry page buffer are to be invalidated (cleared). When this bit is written, the speculation buffer and single entry buffer are immediately cleared. This bit always reads as zero. 0b - Speculation buffer and single entry buffer are not affected. 1b - Invalidate (clear) speculation buffer and single entry buffer.
18-17 B0MW	Memory Width This read-only field defines the width of the memory. 00b - 32 bits 01b - 64 bits 1xb - Reserved
16 —	Reserved
15-8 —	Reserved
7-5 CRC	Cache Replacement Control This 3-bit field defines the replacement algorithm for accesses that are cached. 000b - LRU replacement algorithm per set across all four ways 001b - Reserved 010b - Independent LRU with ways [0-1] for ifetches, [2-3] for data 011b - Independent LRU with ways [0-2] for ifetches, [3] for data 1xxb - Reserved
4 B0DCE	Data Cache Enable This bit controls whether data references are loaded into the cache. 0b - Do not cache data references. 1b - Cache data references.
3 B0ICE	Instruction Cache Enable This bit controls whether instruction fetches are loaded into the cache. 0b - Do not cache instruction fetches. 1b - Cache instruction fetches.
2 B0DPE	Data Prefetch Enable This bit controls whether prefetches (or speculative accesses) are initiated in response to data references. 0b - Do not prefetch in response to data references. 1b - Enable prefetches in response to data references.
1 B0IPE	Instruction Prefetch Enable This bit controls whether prefetches (or speculative accesses) are initiated in response to instruction fetches. 0b - Do not prefetch in response to instruction fetches. 1b - Enable prefetches in response to instruction fetches.

Table continues on the next page...

Field	Function
0 B0SEBE	Single Entry Buffer Enable This bit controls whether the single entry page buffer is enabled in response to flash read accesses. A high-to-low transition of this enable forces the page buffer to be invalidated. 0b - Single entry buffer is disabled. 1b - Single entry buffer is enabled.

17.5.1.4 Cache Tag Storage (TAGVDW0S0 - TAGVDW0S3)

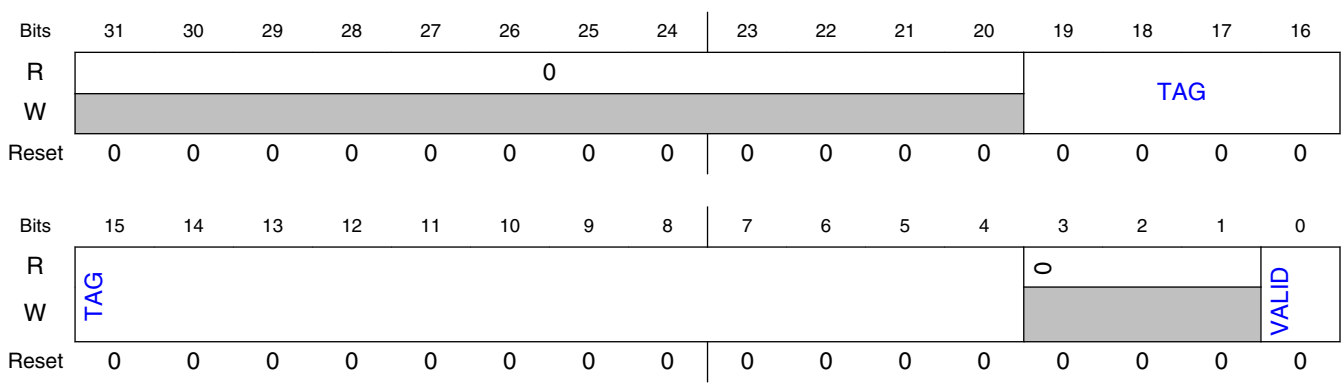
17.5.1.4.1 Offset

Register	Offset
TAGVDW0S0	80h
TAGVDW0S1	82h
TAGVDW0S2	84h
TAGVDW0S3	86h

17.5.1.4.2 Function

The cache of 32-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In TAGVDWxSy, x denotes the way, and y denotes the set. This section represents tag/vld information for all sets in the indicated way.

17.5.1.4.3 Diagram



17.5.1.4.4 Fields

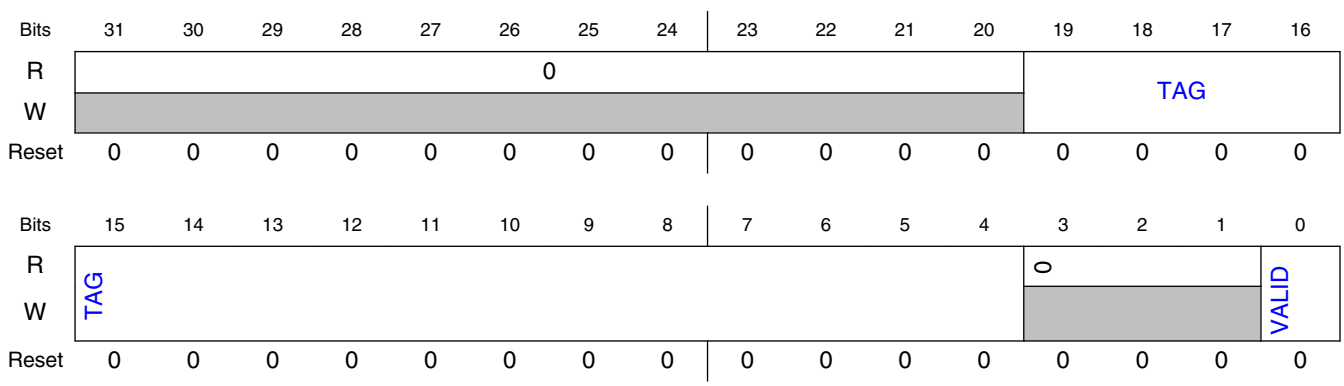
Field	Function
31-20 —	Reserved
19-4 TAG	16-bit17-bit tag for cache entry
3-1 —	Reserved
0 VALID	1-bit valid for cache entry

17.5.1.5 Cache Tag Storage (TAGVDW1S0 - TAGVDW1S3)

17.5.1.5.1 Offset

Register	Offset
TAGVDW1S0	88h
TAGVDW1S1	8Ah
TAGVDW1S2	8Ch
TAGVDW1S3	8Eh

17.5.1.5.2 Diagram



17.5.1.5.3 Fields

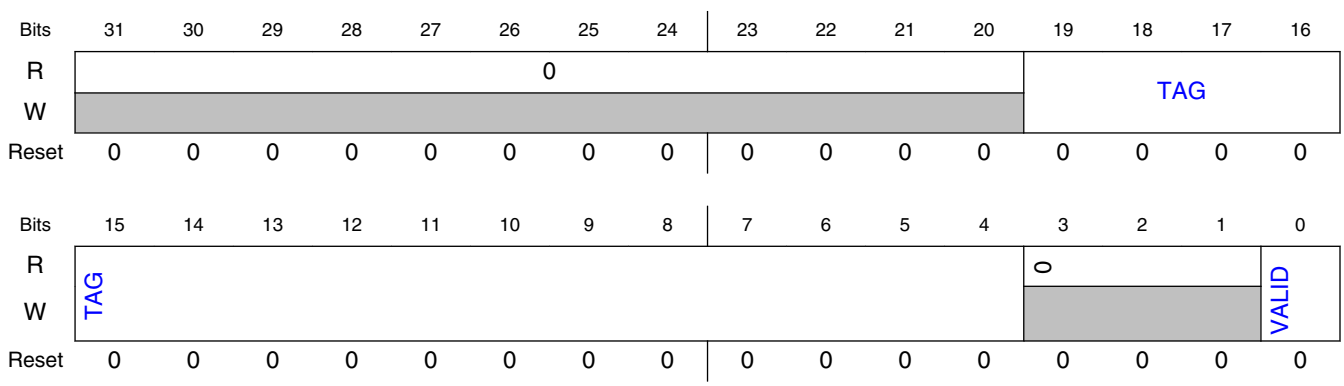
Field	Function
31-20 —	Reserved
19-4 TAG	16-bit17-bit tag for cache entry
3-1 —	Reserved
0 VALID	1-bit valid for cache entry

17.5.1.6 Cache Tag Storage (TAGVDW2S0 - TAGVDW2S3)

17.5.1.6.1 Offset

Register	Offset
TAGVDW2S0	90h
TAGVDW2S1	92h
TAGVDW2S2	94h
TAGVDW2S3	96h

17.5.1.6.2 Diagram



17.5.1.6.3 Fields

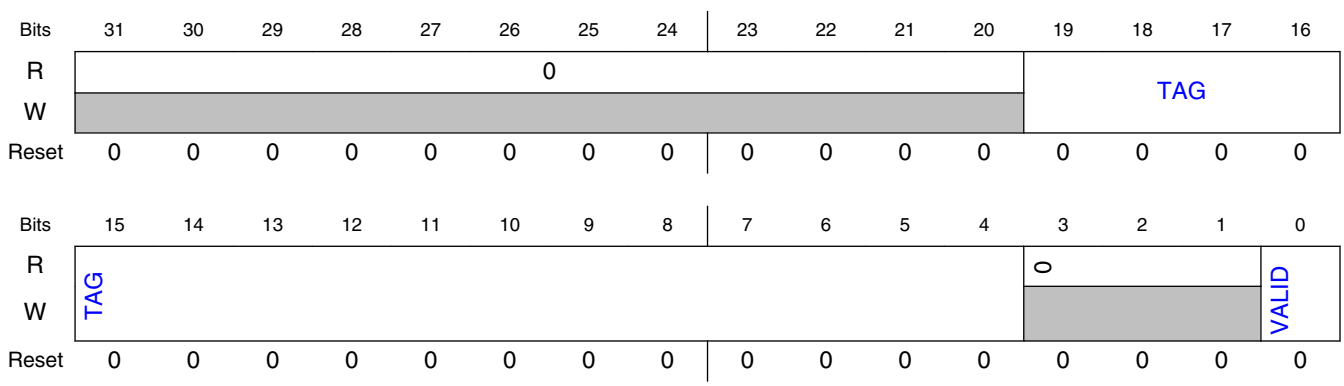
Field	Function
31-20 —	Reserved
19-4 TAG	16-bit17-bit tag for cache entry
3-1 —	Reserved
0 VALID	1-bit valid for cache entry

17.5.1.7 Cache Tag Storage (TAGVDW3S0 - TAGVDW3S3)

17.5.1.7.1 Offset

Register	Offset
TAGVDW3S0	98h
TAGVDW3S1	9Ah
TAGVDW3S2	9Ch
TAGVDW3S3	9Eh

17.5.1.7.2 Diagram



17.5.1.7.3 Fields

Field	Function
31-20 —	Reserved
19-4 TAG	16-bit17-bit tag for cache entry
3-1 —	Reserved
0 VALID	1-bit valid for cache entry

17.5.1.8 Cache Data Storage (DATAW0S0 - DATAW0S3)

17.5.1.8.1 Offset

Register	Offset
DATAW0S0	100h
DATAW0S1	102h
DATAW0S2	104h
DATAW0S3	106h

17.5.1.8.2 Function

The cache of 32-bit entries is a 4-way, set-associative cache with 4 sets. The ways are numbered 0-3 and the sets are numbered 0-3. In DATAWxSy, x denotes the way, and y denotes the set. This section represents data for all sets in the indicated way.

17.5.1.8.3 Diagram

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

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

17.5.1.8.4 Fields

Field	Function
31-0 DATA	Bits [31:0] of data entry

17.5.1.9 Cache Data Storage (DATAW1S0 - DATAW1S3)

17.5.1.9.1 Offset

Register	Offset
DATAW1S0	108h
DATAW1S1	10Ah
DATAW1S2	10Ch
DATAW1S3	10Eh

17.5.1.9.2 Diagram

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

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

17.5.1.9.3 Fields

Field	Function
31-0 DATA	Bits [31:0] of data entry

17.5.1.10 Cache Data Storage (DATAW2S0 - DATAW2S3)

17.5.1.10.1 Offset

Register	Offset
DATAW2S0	110h
DATAW2S1	112h
DATAW2S2	114h
DATAW2S3	116h

17.5.1.10.2 Diagram

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

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

17.5.1.10.3 Fields

Field	Function
31-0 DATA	Bits [31:0] of data entry

17.5.1.11 Cache Data Storage (DATAW3S0 - DATAW3S3)

17.5.1.11.1 Offset

Register	Offset
DATAW3S0	118h
DATAW3S1	11Ah
DATAW3S2	11Ch
DATAW3S3	11Eh

17.5.1.11.2 Diagram

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

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

17.5.1.11.3 Fields

Field	Function
31-0 DATA	Bits [31:0] of data entry

Chapter 18
Flash Memory Module (FTFA)

18.1 Chip-specific information for this module

18.1.1 Flash memory types and terminology

This device contains the following types of flash memory:

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

The following table explains how these types of flash memory align with terminology used in other descriptions of the flash memory.

Table 18-1. Flash memory terminology

Flash memory type	Memory Map	Flash Memory Module
Program flash memory	Primary program/data flash memory	Program flash memory

18.1.2 FOPT Register

The flash memory module's FOPT register allows the user to customize the operation of the MCU at boot time.

FOPT[0] controls whether SIM_PWR or SIM_PWRMODE takes effect.

- FOPT[0]=0: At reset exit, the chip's power modes are controlled through the SIM_PWR register (the SIM_PWRMODE register has no effect).
- FOPT[0]=1: At reset exit, the chip's power modes (including advanced low power modes) are controlled through the SIM_PWRMODE register (the SIM_PWR register has no effect).

18.2 Introduction

The flash memory module includes the following accessible memory regions:

- Program flash memory for vector space and code store

Flash memory is ideal for single-supply applications, permitting in-the-field erase and reprogramming operations without the need for any external high voltage power sources.

The flash memory module includes a memory controller that executes commands to modify flash memory contents. An erased bit reads '1' and a programmed bit reads '0'. The programming operation is unidirectional; it can only move bits from the '1' state (erased) to the '0' state (programmed). Only the erase operation restores bits from '0' to '1'; bits cannot be programmed from a '0' to a '1'.

CAUTION

A flash memory location must be in the erased state before being programmed. Cumulative programming of bits (back-to-back program operations without an intervening erase) within a flash memory location is not allowed. Re-programming of existing 0s to 0 is not allowed as this overstresses the device.

The standard shipping condition for flash memory is erased with security disabled. Data loss over time may occur due to degradation of the erased ('1') states and/or programmed ('0') states. Therefore, it is recommended that each flash block or sector be re-erased immediately prior to factory programming to ensure that the full data retention capability is achieved.

18.2.1 Features

The flash memory module includes the following features.

18.2.1.1 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

18.2.1.2 Other Flash Memory Module Features

- Internal high-voltage supply generator for flash memory program and erase operations
- Optional interrupt generation upon flash command completion
- Supports MCU security mechanisms which prevent unauthorized access to the flash memory contents

18.2.2 Block Diagram

The block diagram of the flash memory module is shown in the following figure.

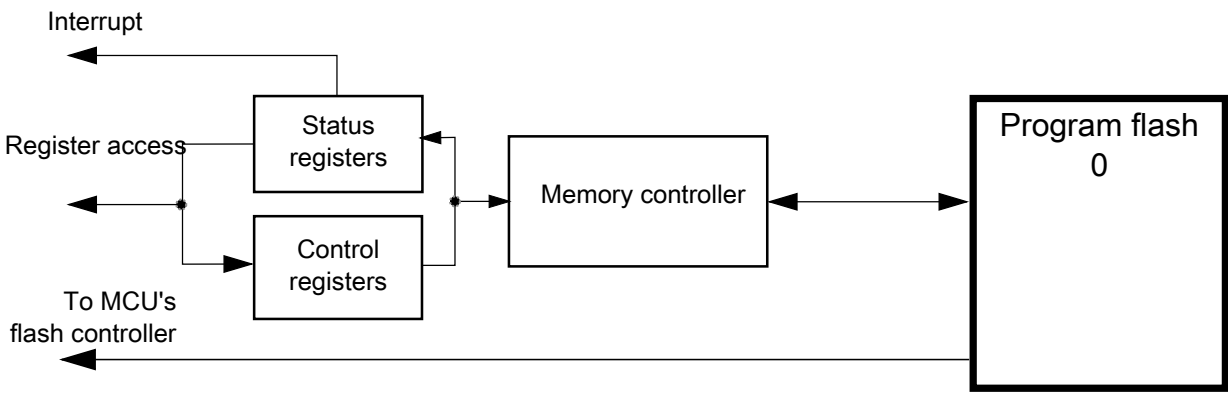


Figure 18-1. Flash Block Diagram

18.2.3 Glossary

Command write sequence — A series of MCU writes to the flash FCCOB register group that initiates and controls the execution of flash algorithms that are built into the flash memory module.

Endurance — The number of times that a flash memory location can be erased and reprogrammed.

FCCOB (Flash Common Command Object) — A group of flash registers that are used to pass command, address, data, and any associated parameters to the memory controller in the flash memory module.

Flash block — A macro within the flash memory module which provides the nonvolatile memory storage.

Flash Memory Module — All flash blocks plus a flash management unit providing high-level control and an interface to MCU buses.

IFR — Nonvolatile information register found in each flash block, separate from the main memory array.

Longword — 32 bits of data with an aligned longword having byte-address[1:0] = 00.

NVM — Nonvolatile memory. A memory technology that maintains stored data during power-off. The flash array is an NVM using NOR-type flash memory technology.

NVM Normal Mode — An NVM mode that provides basic user access to flash memory module resources. The CPU or other bus masters initiate flash program and erase operations (or other flash commands) using writes to the FCCOB register group in the flash memory module.

Program flash — The program flash memory provides nonvolatile storage for vectors and code store.

Program flash Sector — The smallest portion of the program flash memory (consecutive addresses) that can be erased.

Retention — The length of time that data can be kept in the NVM without experiencing errors upon readout. Since erased (1) states are subject to degradation just like programmed (0) states, the data retention limit may be reached from the last erase operation (not from the programming time).

RWW— Read-While-Write. The ability to simultaneously read from one memory resource while commanded operations are active in another memory resource.

Secure — An MCU state conveyed to the flash memory module as described in the Chip Configuration details for this device. In the secure state, reading and changing NVM contents is restricted.

Word — 16 bits of data with an aligned word having byte-address[0] = 0.

18.3 External Signal Description

The flash memory module contains no signals that connect off-chip.

18.4 Memory Map and Registers

This section describes the memory map and registers for the flash memory module.

Data read from unimplemented memory space in the flash memory module is undefined. Writes to unimplemented or reserved memory space (registers) in the flash memory module are ignored.

18.4.1 Flash Configuration Field Description

The program flash memory contains a 16-byte flash configuration field that stores default protection settings (loaded on reset) and security information that allows the MCU to restrict access to the flash memory module.

Flash Configuration Field Offset Address	Size (Bytes)	Field Description
0x0_0400–0x0_0407	8	Backdoor Comparison Key. Refer to Verify Backdoor Access Key Command and Unsecuring the Chip Using Backdoor Key Access.
0x0_0408–0x0_040B	4	Program flash protection bytes. Refer to the description of the Program Flash Protection Registers (FPROT0-3).
0x0_040F	1	Reserved
0x0_040E	1	Reserved
0x0_040D	1	Flash nonvolatile option byte. Refer to the description of the Flash Option Register (FOPT).
0x0_040C	1	Flash security byte. Refer to the description of the Flash Security Register (FSEC).

18.4.2 Program Flash IFR Map

The program flash IFR is nonvolatile information memory that can be read freely, but the user has no erase and limited program capabilities (see the Read Once, Program Once, and Read Resource commands in Read Once Command, Program Once Command and Read Resource Command).

The contents of the program flash IFR are summarized in the table found here and further described in the subsequent paragraphs.

The program flash IFR is located within the program flash 0 memory block .

Address Range	Size (Bytes)	Field Description
0x00 – 0xBF	192	Reserved
0xC0 – 0xFF	64	Program Once Field

18.4.2.1 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).

18.4.3 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).

NOTE

The base address and offsets for these registers are presented in terms of bytes.

FTFA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_C780	Flash Status Register (FTFA_FSTAT)	8	R/W	00h	18.4.3.1/463
1_C781	Flash Configuration Register (FTFA_FCNFG)	8	R/W	00h	18.4.3.2/465
1_C782	Flash Security Register (FTFA_FSEC)	8	R	Undefined	18.4.3.3/466
1_C783	Flash Option Register (FTFA_FOPT)	8	R	Undefined	18.4.3.4/467
1_C784	Flash Common Command Object Registers (FTFA_FCCOB3)	8	R/W	00h	18.4.3.5/468
1_C785	Flash Common Command Object Registers (FTFA_FCCOB2)	8	R/W	00h	18.4.3.5/468

Table continues on the next page...

FTFA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_C786	Flash Common Command Object Registers (FTFA_FCCOB1)	8	R/W	00h	18.4.3.5/468
1_C787	Flash Common Command Object Registers (FTFA_FCCOB0)	8	R/W	00h	18.4.3.5/468
1_C788	Flash Common Command Object Registers (FTFA_FCCOB7)	8	R/W	00h	18.4.3.5/468
1_C789	Flash Common Command Object Registers (FTFA_FCCOB6)	8	R/W	00h	18.4.3.5/468
1_C78A	Flash Common Command Object Registers (FTFA_FCCOB5)	8	R/W	00h	18.4.3.5/468
1_C78B	Flash Common Command Object Registers (FTFA_FCCOB4)	8	R/W	00h	18.4.3.5/468
1_C78C	Flash Common Command Object Registers (FTFA_FCCOBB)	8	R/W	00h	18.4.3.5/468
1_C78D	Flash Common Command Object Registers (FTFA_FCCOBA)	8	R/W	00h	18.4.3.5/468
1_C78E	Flash Common Command Object Registers (FTFA_FCCOB9)	8	R/W	00h	18.4.3.5/468
1_C78F	Flash Common Command Object Registers (FTFA_FCCOB8)	8	R/W	00h	18.4.3.5/468
1_C790	Program Flash Protection Registers (FTFA_FPROT3)	8	R/W	Undefined	18.4.3.6/469
1_C791	Program Flash Protection Registers (FTFA_FPROT2)	8	R/W	Undefined	18.4.3.6/469
1_C792	Program Flash Protection Registers (FTFA_FPROT1)	8	R/W	Undefined	18.4.3.6/469
1_C793	Program Flash Protection Registers (FTFA_FPROT0)	8	R/W	Undefined	18.4.3.6/469

18.4.3.1 Flash Status Register (FTFA_FSTAT)

The FSTAT register reports the operational status of the flash memory module.

The CCIF, RDCOLERR, ACCERR, and FPVIOL bits are readable and writable. The MGSTAT0 bit is read only. The unassigned bits read 0 and are not writable.

NOTE

When set, the Access Error (ACCERR) and Flash Protection Violation (FPVIOL) bits in this register prevent the launch of any more commands until the flag is cleared (by writing a one to it).

Address: 1_C780h base + 0h offset = 1_C780h

Bit	7	6	5	4	3	2	1	0
Read	CCIF	RDCOLERR	ACCERR	FPVIOL	0			MGSTAT0
Write	w1c	w1c	w1c	w1c				
Reset	0	0	0	0	0	0	0	0

FTFA_FSTAT field descriptions

Field	Description
7 CCIF	Command Complete Interrupt Flag Indicates that a flash command has completed. The CCIF flag is cleared by writing a 1 to CCIF to launch a command, and CCIF stays low until command completion or command violation. CCIF is reset to 0 but is set to 1 by the memory controller at the end of the reset initialization sequence. Depending on how quickly the read occurs after reset release, the user may or may not see the 0 hardware reset value. 0 Flash command in progress 1 Flash command has completed
6 RDCOLERR	Flash Read Collision Error Flag Indicates that the MCU attempted a read from a flash memory resource that was being manipulated by a flash command (CCIF=0). Any simultaneous access is detected as a collision error by the block arbitration logic. The read data in this case cannot be guaranteed. The RDCOLERR bit is cleared by writing a 1 to it. Writing a 0 to RDCOLERR has no effect. 0 No collision error detected 1 Collision error detected
5 ACCERR	Flash Access Error Flag Indicates an illegal access has occurred to a flash memory resource caused by a violation of the command write sequence or issuing an illegal flash command. While ACCERR is set, the CCIF flag cannot be cleared to launch a command. The ACCERR bit is cleared by writing a 1 to ACCERR while CCIF is set. Writing a 0 to the ACCERR bit has no effect. 0 No access error detected 1 Access error detected
4 FPVIOL	Flash Protection Violation Flag Indicates an attempt was made to program or erase an address in a protected area of program flash memory during a command write sequence . While FPVIOL is set, the CCIF flag cannot be cleared to launch a command. The FPVIOL bit is cleared by writing a 1 to FPVIOL while CCIF is set. Writing a 0 to the FPVIOL bit has no effect. 0 No protection violation detected 1 Protection violation detected
3–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 MGSTAT0	Memory Controller Command Completion Status Flag The MGSTAT0 status flag is set if an error is detected during execution of a flash command or during the flash reset sequence. As a status flag, this field cannot (and need not) be cleared by the user like the other error flags in this register.

Table continues on the next page...

FTFA_FSTAT field descriptions (continued)

Field	Description
	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.

18.4.3.2 Flash Configuration Register (FTFA_FCNFG)

This register provides information on the current functional state of the flash memory module.

The erase control bits (ERSAREQ and ERSSUSP) have write restrictions. The unassigned bits read as noted and are not writable.

Address: 1_C780h base + 1h offset = 1_C781h

Bit	7	6	5	4	3	2	1	0
Read	CCIE	RDCOLLIE	ERSAREQ	ERSSUSP	0	0	0	0
Write								
Reset	0	0	0	0	0	0	0	0

FTFA_FCNFG field descriptions

Field	Description
7 CCIE	Command Complete Interrupt Enable Controls interrupt generation when a flash command completes. 0 Command complete interrupt disabled 1 Command complete interrupt enabled. An interrupt request is generated whenever the FSTAT[CCIF] flag is set.
6 RDCOLLIE	Read Collision Error Interrupt Enable Controls interrupt generation when a flash memory read collision error occurs. 0 Read collision error interrupt disabled 1 Read collision error interrupt enabled. An interrupt request is generated whenever a flash memory read collision error is detected (see the description of FSTAT[RDCOLERR]).
5 ERSAREQ	Erase All Request Issues a request to the memory controller to execute the Erase All Blocks command and release security. ERSAREQ is not directly writable but is under indirect user control. Refer to the device's Chip Configuration details on how to request this command. ERSAREQ sets when an erase all request is triggered external to the flash memory module and CCIF is set (no command is currently being executed). ERSAREQ is cleared by the flash memory module when the operation completes. 0 No request or request complete 1 Request to: 1. run the Erase All Blocks command,

Table continues on the next page...

FTFA_FCNFG field descriptions (continued)

Field	Description
	2. verify the erased state, 3. program the security byte in the Flash Configuration Field to the unsecure state, and 4. release MCU security by setting the FSEC[SEC] field to the unsecure state.
4 ERSSUSP	Erase Suspend Allows the user to suspend (interrupt) the Erase Flash Sector command while it is executing. 0 No suspend requested 1 Suspend the current Erase Flash Sector command execution.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

18.4.3.3 Flash Security Register (FTFA_FSEC)

This read-only register holds all bits associated with the security of the MCU and flash memory module.

During the reset sequence, the register is loaded with the contents of the flash security byte in the Flash Configuration Field located in program flash memory. The flash basis for the values is signified by X in the reset value.

Address: 1_C780h base + 2h offset = 1_C782h

Bit	7	6	5	4	3	2	1	0
Read	KEYEN		MEEN		FSLACC		SEC	
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FTFA_FSEC field descriptions

Field	Description
7–6 KEYEN	Backdoor Key Security Enable Enables or disables backdoor key access to the flash memory module. 00 Backdoor key access disabled 01 Backdoor key access disabled (preferred KEYEN state to disable backdoor key access)

Table continues on the next page...

FTFA_FSEC field descriptions (continued)

Field	Description
	10 Backdoor key access enabled 11 Backdoor key access disabled
5–4 MEEN	Mass Erase Enable Enables and disables mass erase capability of the flash memory module. When SEC is set to unsecure, the MEEN setting does not matter. 00 Mass erase is enabled 01 Mass erase is enabled 10 Mass erase is disabled 11 Mass erase is enabled
3–2 FSLACC	Factory Security Level Access Code Enables or disables access to the flash memory contents during returned part failure analysis at NXP. When SEC is secure and FSLACC is denied, access to the program flash contents is denied and any failure analysis performed by NXP 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), NXP factory testing has visibility of the current flash contents. The state of the FSLACC bits is only relevant when SEC is set to secure. When SEC is set to unsecure, the FSLACC setting does not matter. 00 NXP factory access granted 01 NXP factory access denied 10 NXP factory access denied 11 NXP factory access granted
SEC	Flash Security Defines the security state of the MCU. In the secure state, the MCU limits access to flash memory module resources. The limitations are defined per device and are detailed in the Chip Configuration details. If the flash memory module is unsecured using backdoor key access, SEC is forced to 10b. 00 MCU security status is secure. 01 MCU security status is secure. 10 MCU security status is unsecure. (The standard shipping condition of the flash memory module is unsecure.) 11 MCU security status is secure.

18.4.3.4 Flash Option Register (FTFA_FOPT)

The flash option register allows the MCU to customize its operations by examining the state of these read-only bits, which are loaded from NVM at reset. The function of the bits is defined in the device's Chip Configuration details.

All bits in the register are read-only .

During the reset sequence, the register is loaded from the flash nonvolatile option byte in the Flash Configuration Field located in program flash memory. The flash basis for the values is signified by X in the reset value. However, the register is written to 0xFF if the contents of the flash nonvolatile option byte are 0x00.

Address: 1_C780h base + 3h offset = 1_C783h

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.

FTFA_FOPT field descriptions

Field	Description
OPT	Nonvolatile Option These bits are loaded from flash to this register at reset. Refer to the device's Chip Configuration details for the definition and use of these bits.

18.4.3.5 Flash Common Command Object Registers (FTFA_FCCOB*n*)

The FCCOB register group provides 12 bytes for command codes and parameters. The individual bytes within the set append a 0-B hex identifier to the FCCOB register name: FCCOB0, FCCOB1, ..., FCCOBB.

Address: 1_C780h 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

FTFA_FCCOB*n* field descriptions

Field	Description
CCOBn	The FCCOB register provides a command code and relevant parameters to the memory controller. The individual registers that compose the FCCOB data set can be written in any order, but you must provide all needed values, which vary from command to command. First, set up all required FCCOB fields and then initiate the command's execution by writing a 1 to the FSTAT[CCIF] bit. This clears the CCIF bit, which locks all FCCOB parameter fields and they cannot be changed by the user until the command completes (CCIF returns to 1). No command buffering or queueing is provided; the next command can be loaded only after the current command completes. 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.

FTFA_FCCOB*n* field descriptions (continued)

Field	Description
	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 1 byte, the most significant data resides in the lowest FCCOB register number. The FCCOB register group may be read and written as individual bytes, aligned words (2 bytes) or aligned longwords (4 bytes).

18.4.3.6 Program Flash Protection Registers (FTFA_FPROT*n*)

The FPROT registers define which program flash regions are protected from program and erase operations. Protected flash regions cannot have their content changed; that is, these regions cannot be programmed and cannot be erased by any flash command. Unprotected regions can be changed by program and erase operations.

The four FPROT registers allow up to 32 protectable regions. Each bit protects a 1/32 region of the program flash memory except for memory configurations with less than 32 KB of program flash where each assigned bit protects 1 KB. For configurations with 96 KB or 48 KB of program flash memory, FPROT0 is not used and each assigned bit protects 4 KB or 2 KB respectively. For configurations with 24 KB of program flash memory or less, FPROT0 is not used. For configurations with 16 KB of program flash memory or less, FPROT1 is not used. For configurations with 8 KB of program flash memory, FPROT2 is not used. The bitfields are defined in each register as follows:

Program flash protection register	Program flash protection bits
FPROT0	PROT[31:24]
FPROT1	PROT[23:16]
FPROT2	PROT[15:8]
FPROT3	PROT[7:0]

During the reset sequence, the FPROT registers are loaded with the contents of the program flash protection bytes in the Flash Configuration Field as indicated in the following table.

Program flash protection register	Flash Configuration Field offset address
FPROT0	0x000B
FPROT1	0x000A
FPROT2	0x0009
FPROT3	0x0008

To change the program flash protection that is loaded during the reset sequence, unprotect the sector of program flash memory that contains the Flash Configuration Field. Then, reprogram the program flash protection byte.

Address: 1_C780h base + 10h offset + (1d × i), where i=0d to 3d

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

* Notes:
- x = Undefined at reset.

FTFA_FPROT*n* field descriptions

Field	Description
PROT	Program Flash Region Protect Each program flash region can be protected from program and erase operations by setting the associated PROT bit. 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. **Restriction:** The user must never write to any FPROT register while a command is running (CCIF=0). Trying to alter data in any protected area in the program flash memory results in a protection violation error and sets the FSTAT[FPVIOL] bit. A full block erase of a program flash block is not possible if it contains any protected region. Each bit in the 32-bit protection register represents 1/32 of the total program flash except for memory configurations with less than 32 KB of program flash where each assigned bit protects 1 KB .

FTFA_FPROT*n* field descriptions (continued)

Field	Description
	0 Program flash region is protected. 1 Program flash region is not protected

18.5 Functional Description

The information found here describes functional details of the flash memory module.

18.5.1 Flash Protection

Individual regions within the flash memory can be protected from program and erase operations.

Protection is controlled by the following registers:

- FPROT*n* —
  - For 2^n program flash sizes, four registers typically protect 32 regions of the program flash memory as shown in the following figure

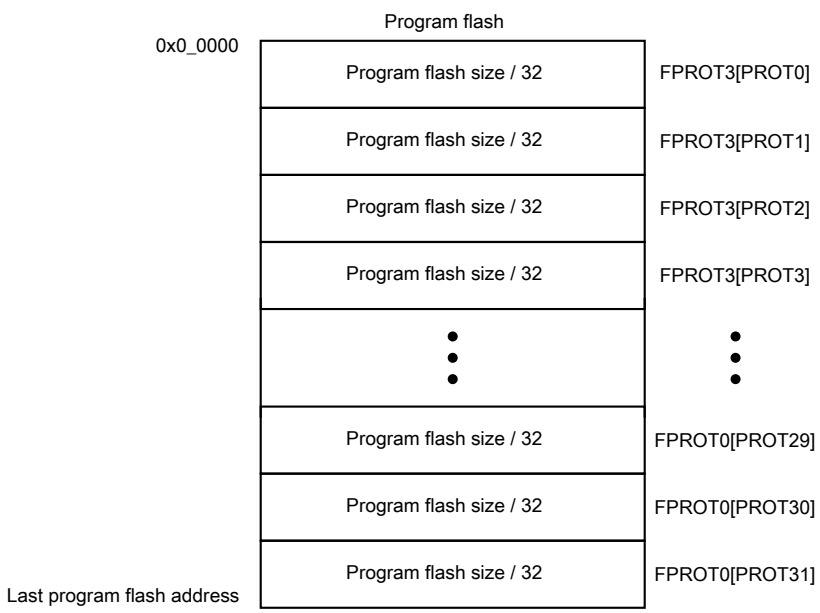


Figure 18-2. Program flash protection

NOTE

Flash protection features are discussed further in AN4507: Using the Kinetis Security and Flash Protection Features also

applicable for MWxx family . Not all features described in the application note are available on this device.

18.5.2 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 18-2. 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.

18.5.3 Flash Operation in Low-Power Modes

18.5.3.1 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).

18.5.3.2 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.

18.5.4 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).

18.5.5 Read While Write (RWW)

The following simultaneous accesses are not allowed:

- Reading from program flash memory space while a flash command is active (CCIF=0).

18.5.6 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.

18.5.7 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

18.5.7.1 Command Write Sequence

Flash commands are specified using a command write sequence illustrated in Figure 18-3. The flash memory module performs various checks on the command (FCCOB) content and continues with command execution if all requirements are fulfilled.

Before launching a command, the ACCERR and FPVIOL bits in the FSTAT register must be zero and the CCIF flag must read 1 to verify that any previous command has completed. If CCIF is zero, the previous command execution is still active, a new command write sequence cannot be started, and all writes to the FCCOB registers are ignored.

Attempts to launch a flash command in VLP mode will be ignored.

18.5.7.1.1 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.

18.5.7.1.2 Launch the Command by Clearing CCIF

Once all relevant command parameters have been loaded, the user launches the command by clearing FSTAT[CCIF] by writing a '1' to it. FSTAT[CCIF] remains 0 until the flash command completes.

The FSTAT register contains a blocking mechanism that prevents a new command from launching (can't clear FSTAT[CCIF]) if the previous command resulted in an access error (FSTAT[ACCERR]=1) or a protection violation (FSTAT[FPVIOL]=1). In error scenarios, two writes to FSTAT are required to initiate the next command: the first write clears the error flags, the second write clears CCIF.

18.5.7.1.3 Command Execution and Error Reporting

The command processing has several steps:

1. The flash memory module reads the command code and performs a series of parameter checks and protection checks, if applicable, which are unique to each command.

   If the parameter check fails, the FSTAT[ACCERR] (access error) flag is set. FSTAT[ACCERR] reports invalid instruction codes and out-of bounds addresses. Usually, access errors suggest that the command was not set-up with valid parameters in the FCCOB register group.

   Program and erase commands also check the address to determine if the operation is requested to execute on protected areas. If the protection check fails, FSTAT[FPVIOL] (protection error) flag is set.

   Command processing never proceeds to execution when the parameter or protection step fails. Instead, command processing is terminated after setting FSTAT[CCIF].

2. If the parameter and protection checks pass, the command proceeds to execution. Run-time errors, such as failure to erase verify, may occur during the execution phase. Run-time errors are reported in FSTAT[MGSTAT0]. A command may have access errors, protection errors, and run-time errors, but the run-time errors are not seen until all access and protection errors have been corrected.
3. Command execution results, if applicable, are reported back to the user via the FCCOB and FSTAT registers.
4. The flash memory module sets FSTAT[CCIF] signifying that the command has completed.

The flow for a generic command write sequence is illustrated in the following figure.

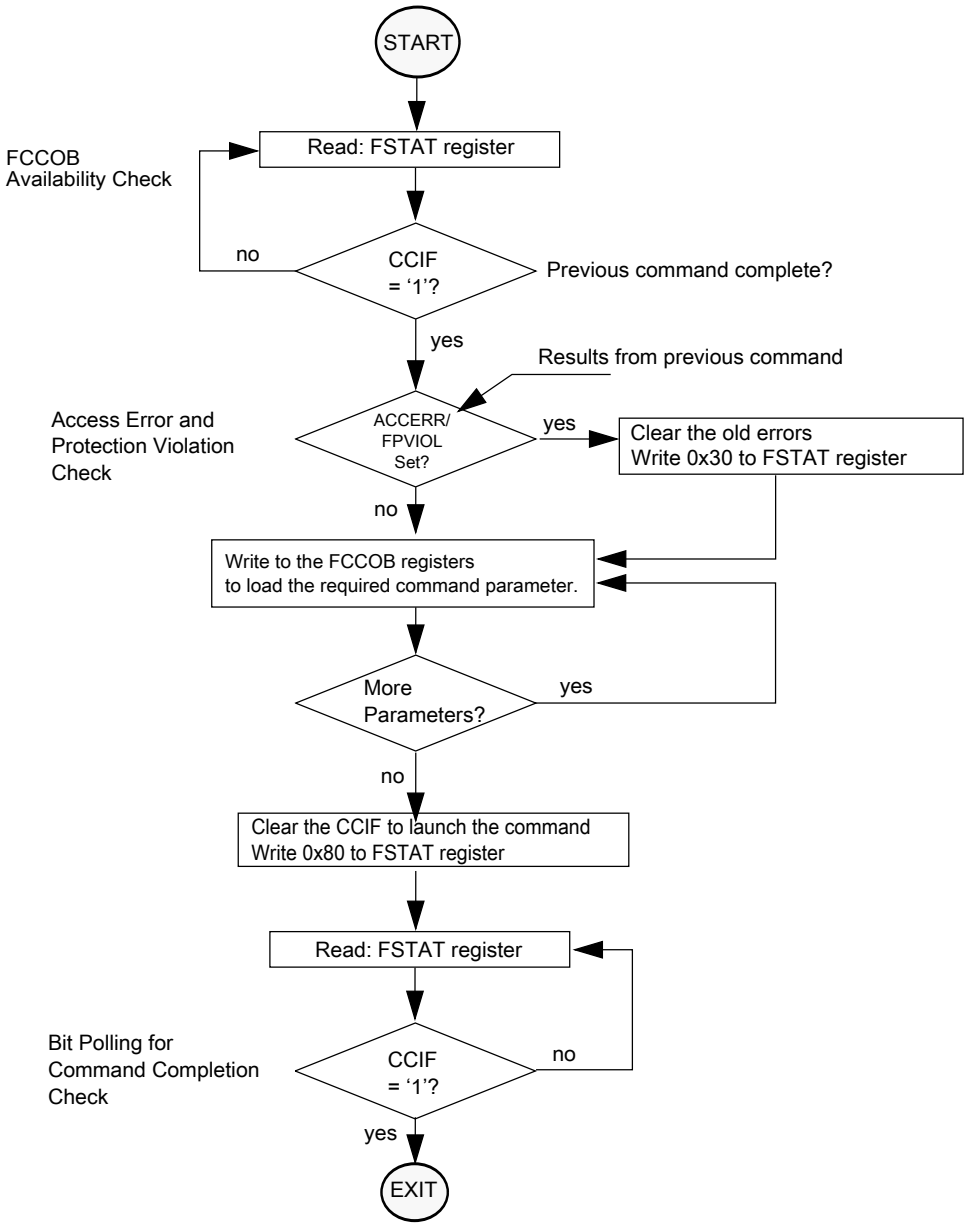


Figure 18-3. Generic flash command write sequence flowchart

18.5.7.2 Flash Commands

The following table summarizes the function of all flash commands.

FCMD	Command	Program flash	Function
0x01	Read 1s Section	×	Verify that a given number of program flash locations from a starting address are erased.

Table continues on the next page...

FCMD	Command	Program flash	Function
0x02	Program Check	×	Tests previously-programmed locations at margin read levels.
0x03	Read Resource	IFR, ID	Read 4 bytes from program flash IFR or version ID.
0x06	Program Longword	×	Program 4 bytes in a program flash block.
0x09	Erase Flash Sector	×	Erase all bytes in a program flash sector.
0x40	Read 1s All Blocks	×	Verify that the program flash block is erased then release MCU security.
0x41	Read Once	IFR	Read 4 bytes of a dedicated 64 byte field in the program flash 0 IFR.
0x43	Program Once	IFR	One-time program of 4 bytes of a dedicated 64-byte field in the program flash 0 IFR.
0x44	Erase All Blocks	×	Erase the program flash block, verify-erase and release MCU security. **NOTE:** An erase is only possible when all memory locations are unprotected.
0x45	Verify Backdoor Access Key	×	Release MCU security after comparing a set of user-supplied security keys to those stored in the program flash.
0x49	Erase All Blocks Unsecure	×	Erase the program flash block, verify-erase, program security byte to unsecure state, release MCU security.

18.5.8 Margin Read Commands

The Read-1s commands (Read 1s All Blocks, 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 array reads performed by these commands. Using the preset 'user' and 'factory' margin levels, these commands perform their associated read operations at tighter tolerances than a 'normal' read. These non-standard read levels are applied only during the command execution. Basic flash array reads use the standard, un-margined, read reference level.

Only the 'normal' read level should be employed during normal flash usage. The non-standard, 'user' and 'factory' margin levels should be employed only in special cases. They can be used during special diagnostic routines to gain confidence that the device is not suffering from the end-of-life data loss customary of flash memory devices.

Erased ('1') and programmed ('0') bit states can degrade due to elapsed time and data cycling (number of times a bit is erased and re-programmed). The lifetime of the erased states is relative to the last erase operation. The lifetime of the programmed states is measured from the last program time.

The 'user' and 'factory' levels become, in effect, a minimum safety margin; i.e. if the reads pass at the tighter tolerances of the 'user' and 'factory' margins, then the 'normal' reads have at least this much safety margin before they experience data loss.

The 'user' margin is a small delta to the normal read reference level. 'User' margin levels can be employed to check that flash memory contents have adequate margin for normal level read operations. If unexpected read results are encountered when checking flash memory contents at the 'user' margin levels, loss of information might soon occur during 'normal' readout.

The 'factory' margin is a bigger deviation from the norm, a more stringent read criteria that should only be attempted immediately (or very soon) after completion of an erase or program command, early in the cycling life. 'Factory' margin levels can be used to check that flash memory contents have adequate margin for long-term data retention at the normal level setting. If unexpected results are encountered when checking flash memory contents at 'factory' margin levels, the flash memory contents should be erased and reprogrammed.

CAUTION

Factory margin levels must only be used during verify of the initial factory programming.

18.5.9 Flash Command Description

This section describes all flash commands that can be launched by a command write sequence.

The flash memory module sets the FSTAT[ACCERR] bit and aborts the command execution if any of the following illegal conditions occur:

- There is an unrecognized command code in the FCCOB FCMD field.
- There is an error in a FCCOB field for the specific commands. Refer to the error handling table provided for each command.

Ensure that FSTAT[ACCERR] and FSTAT[FPVIOL] are cleared prior to starting the command write sequence. As described in Launch the Command by Clearing CCIF, a new command cannot be launched while these error flags are set.

Do not attempt to read a flash block while the flash memory module is running a command (FSTAT[CCIF] = 0) on that same block. The flash memory module may return invalid data to the MCU with the collision error flag (FSTAT[RDCOLERR]) set.

CAUTION

Flash data must be in the erased state before being programmed. Cumulative programming of bits (adding more zeros) is not allowed.

18.5.9.1 Read 1s Section Command

The Read 1s Section command checks if a section of program flash memory is erased to the specified read margin level. The Read 1s Section command defines the starting address and the number of longwords to be verified.

Table 18-3. 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 18-4 and then reads all locations within the specified section of flash memory. If the flash memory module fails to read all 1s (that is, the flash section is not erased), FSTAT[MGSTAT0] is set. FSTAT[CCIF] sets after the Read 1s Section operation completes.

Table 18-4. Margin Level Choices for Read 1s Section

Read Margin Choice	Margin Level Description
0x00	Use the 'normal' read level for 1s
0x01	Apply the 'User' margin to the normal read-1 level
0x02	Apply the 'Factory' margin to the normal read-1 level

Table 18-5. Read 1s Section Command Error Handling

Error condition	Error bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid margin code is supplied.	FSTAT[ACCERR]
An invalid flash address is supplied.	FSTAT[ACCERR]
Flash address is not longword aligned.	FSTAT[ACCERR]
The requested section crosses a Flash block boundary.	FSTAT[ACCERR]
The requested number of longwords is 0.	FSTAT[ACCERR]
Read-1s fails.	FSTAT[MGSTAT0]

18.5.9.2 Program Check Command

The Program Check command tests a previously programmed program flash longword to see if it reads correctly at the specified margin level.

Table 18-6. Program Check Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x02 (PGMCHK)
1	Flash address [23:16]
2	Flash address [15:8]
3	Flash address [7:0][1]
4	Margin Choice
8	Byte 0 expected data
9	Byte 1 expected data
A	Byte 2 expected data
B	Byte 3 expected data

1. Must be longword aligned (Flash address [1:0] = 00).

Upon clearing CCIF to launch the Program Check command, the flash memory module sets the read margin for 1s according to Table 18-7, reads the specified longword, and compares the actual read data to the expected data provided by the FCCOB. If the comparison at margin-1 fails, FSTAT[MGSTAT0] is set.

The flash memory module then sets the read margin for 0s, re-reads, and compares again. If the comparison at margin-0 fails, FSTAT[MGSTAT0] is set. FSTAT[CCIF] is set after the Program Check operation completes.

The supplied address must be longword aligned (the lowest two bits of the byte address must be 00):

- Byte 3 data is written to the supplied byte address ('start'),
- Byte 2 data is programmed to byte address start+0b01,
- Byte 1 data is programmed to byte address start+0b10,
- Byte 0 data is programmed to byte address start+0b11.

NOTE

See the description of margin reads, Margin Read Commands

Table 18-7. Margin Level Choices for Program Check

Read Margin Choice	Margin Level Description
0x01	Read at 'User' margin-1 and 'User' margin-0
0x02	Read at 'Factory' margin-1 and 'Factory' margin-0

Table 18-8. Program Check Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid flash address is supplied	FSTAT[ACCERR]
Flash address is not longword aligned	FSTAT[ACCERR]
An invalid margin choice is supplied	FSTAT[ACCERR]
Either of the margin reads does not match the expected data	FSTAT[MGSTAT0]

18.5.9.3 Read Resource Command

The Read Resource command allows the user to read data from special-purpose memory resources located within the flash memory module. The special-purpose memory resources available include program flash IFR space and the Version ID field. Each resource is assigned a select code as shown in Table 18-10.

Table 18-9. Read Resource Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x03 (RDRSRC)
1	Flash address [23:16]
2	Flash address [15:8]

Table continues on the next page...

Table 18-9. 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 18-10)

1. Must be longword aligned (Flash address [1:0] = 00).

Table 18-10. Read Resource Select Codes

Resource Select Code	Description	Resource Size	Local Address Range
0x00	Program Flash 0 IFR	256 Bytes	0x00_0000–0x00_00FF
0x01[1]	Version ID	8 Bytes	0x00_0000–0x00_0007

1. Located in program flash 0 reserved space.

After clearing CCIF to launch the Read Resource command, four consecutive bytes are read from the selected resource at the provided relative address and stored in the FCCOB register. The CCIF flag sets after the Read Resource operation completes. The Read Resource command exits with an access error if an invalid resource code is provided or if the address for the applicable area is out-of-range.

Table 18-11. Read Resource Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid resource code is entered	FSTAT[ACCERR]
Flash address is out-of-range for the targeted resource.	FSTAT[ACCERR]
Flash address is not longword aligned	FSTAT[ACCERR]

18.5.9.4 Program Longword Command

The Program Longword command programs four previously-erased bytes in the program flash memory using an embedded algorithm.

CAUTION

A flash memory location must be in the erased state before being programmed. Cumulative programming of bits (back-to-back program operations without an intervening erase) within a flash memory location is not allowed. Re-programming of existing 0s to 0 is not allowed as this overstresses the device.

Table 18-12. Program Longword Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x06 (PGM4)
1	Flash address [23:16]
2	Flash address [15:8]
3	Flash address [7:0][1]
4	Byte 0 program value
5	Byte 1 program value
6	Byte 2 program value
7	Byte 3 program value

1. Must be longword aligned (Flash address [1:0] = 00).

Upon clearing CCIF to launch the Program Longword command, the flash memory module programs the data bytes into the flash using the supplied address. The targeted flash locations must be currently unprotected (see the description of the FPROT registers) to permit execution of the Program Longword operation.

The programming operation is unidirectional. It can only move NVM bits from the erased state ('1') to the programmed state ('0'). Erased bits that fail to program to the '0' state are flagged as errors in FSTAT[MGSTAT0]. The CCIF flag is set after the Program Longword operation completes.

The supplied address must be longword aligned (flash address [1:0] = 00):

- Byte 3 data is written to the supplied byte address ('start'),
- Byte 2 data is programmed to byte address start+0b01,
- Byte 1 data is programmed to byte address start+0b10, and
- Byte 0 data is programmed to byte address start+0b11.

Table 18-13. Program Longword Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid flash address is supplied	FSTAT[ACCERR]
Flash address is not longword aligned	FSTAT[ACCERR]
Flash address points to a protected area	FSTAT[FPVIOL]

Table continues on the next page...

Table 18-13. Program Longword Command Error Handling (continued)

Error Condition	Error Bit
Any errors have been encountered during the verify operation	FSTAT[MGSTAT0]

18.5.9.5 Erase Flash Sector Command

The Erase Flash Sector operation erases all addresses in a flash sector.

Table 18-14. Erase Flash Sector Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x09 (ERSSCR)
1	Flash address [23:16] in the flash sector to be erased
2	Flash address [15:8] in the flash sector to be erased
3	Flash address [7:0][1] in the flash sector to be erased

1. Must be 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 sector and then verifies that it is erased. The Erase Flash Sector command aborts if the selected sector is protected (see the description of the FPROT registers). If the erase-verify fails the FSTAT[MGSTAT0] bit is set. The CCIF flag is set after the Erase Flash Sector operation completes. The Erase Flash Sector command is suspendable (see the FCNFG[ERSSUSP] bit and Figure 18-4).

Table 18-15. Erase Flash Sector Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid Flash address is supplied	FSTAT[ACCERR]
Flash address is not longword aligned	FSTAT[ACCERR]
The selected program 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.

18.5.9.5.1 Suspending an Erase Flash Sector Operation

To suspend an Erase Flash Sector operation set the FCNFG[ERSSUSP] bit when CCIF, ACCERR, and FPVIOL are clear and the CCOB command field holds the code for the Erase Flash Sector command. During the Erase Flash Sector operation (see Erase Flash Sector Command), the flash memory module samples the state of the ERSSUSP bit at convenient points. If the flash memory module detects that the ERSSUSP bit is set, the

Erase Flash Sector operation is suspended and the flash memory module sets CCIF. While ERSSUSP is set, all writes to flash registers are ignored except for writes to the FSTAT and FCNFG registers.

If an Erase Flash Sector operation effectively completes before the flash memory module detects that a suspend request has been made, the flash memory module clears the ERSSUSP bit prior to setting CCIF. When an Erase Flash Sector operation has been successfully suspended, the flash memory module sets CCIF and leaves the ERSSUSP bit set. While CCIF is set, the ERSSUSP bit can only be cleared to prevent the withdrawal of a suspend request before the flash memory module has acknowledged it.

18.5.9.5.2 Resuming a Suspended Erase Flash Sector Operation

If the ERSSUSP bit is still set when CCIF is cleared to launch the next command, the previous Erase Flash Sector operation resumes. The flash memory module acknowledges the request to resume a suspended operation by clearing the ERSSUSP bit. A new suspend request can then be made by setting ERSSUSP. A single Erase Flash Sector operation can be suspended and resumed multiple times.

There is a minimum elapsed time limit of 4.3 msec between the request to resume the Erase Flash Sector operation (CCIF is cleared) and the request to suspend the operation again (ERSSUSP is set). This minimum time period is required to ensure that the Erase Flash Sector operation will eventually complete. If the minimum period is continually violated, i.e. the suspend requests come repeatedly and too quickly, no forward progress is made by the Erase Flash Sector algorithm. The resume/suspend sequence runs indefinitely without completing the erase.

18.5.9.5.3 Aborting a Suspended Erase Flash Sector Operation

The user may choose to abort a suspended Erase Flash Sector operation by clearing the ERSSUSP bit prior to clearing CCIF for the next command launch. When a suspended operation is aborted, the flash memory module starts the new command using the new FCCOB contents.

Note

Aborting the erase leaves the bitcells in an indeterminate, partially-erased state. Data in this sector is not reliable until a new erase command fully completes.

The following figure shows how to suspend and resume the Erase Flash Sector operation.

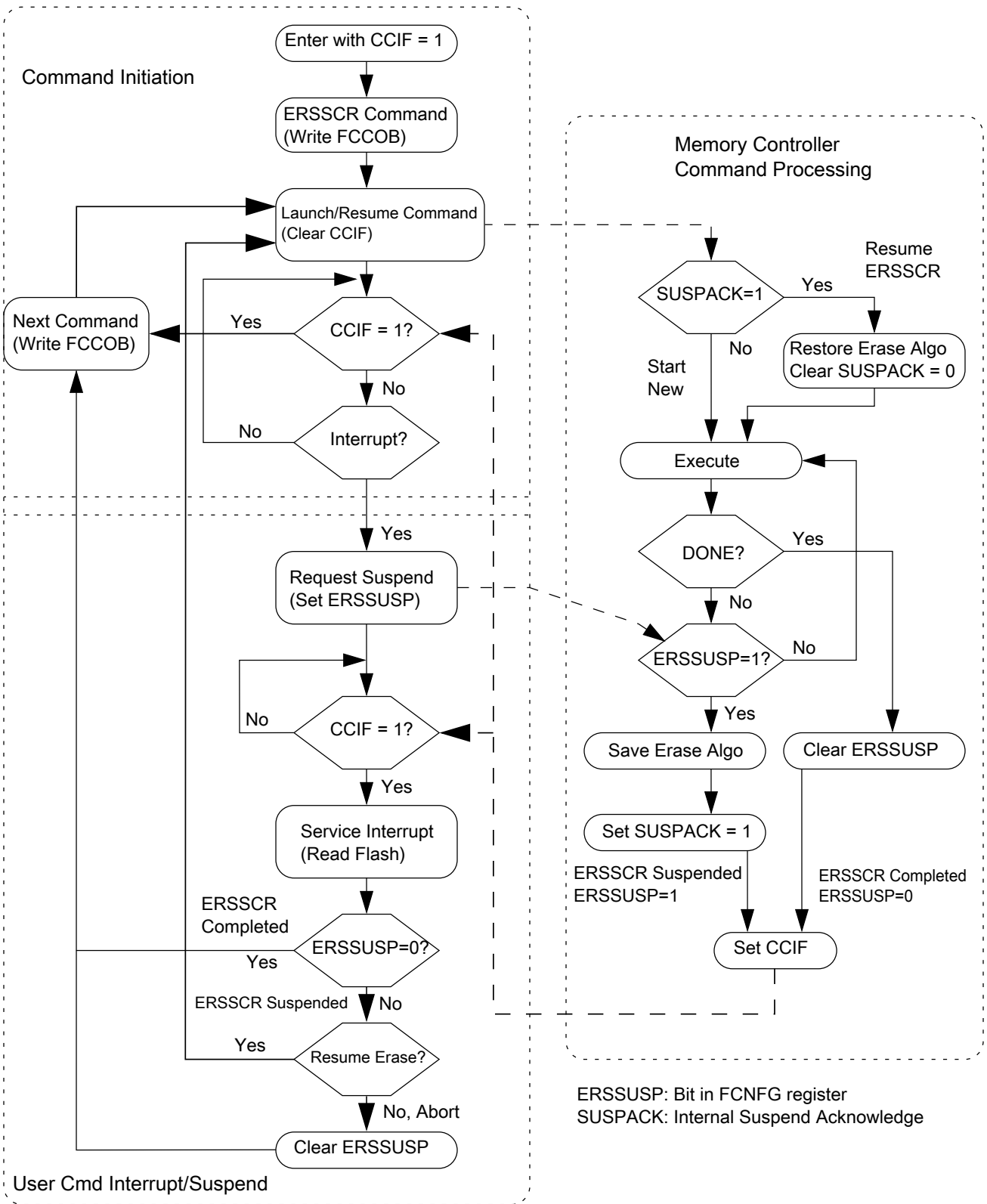


Figure 18-4. Suspend and Resume of Erase Flash Sector Operation

18.5.9.6 Read 1s All Blocks Command

The Read 1s All Blocks command checks if the program flash blocks 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 18-16. Read 1s All Blocks Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x40 (RD1ALL)
1	Read-1 Margin Choice

After clearing CCIF to launch the Read 1s All Blocks command, the flash memory module :

- sets the read margin for 1s according to Table 18-17,
- checks the contents of the program flash 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 CCIF flag sets after the Read 1s All Blocks operation has completed.

Table 18-17. Margin Level Choices for Read 1s All Blocks

Read Margin Choice	Margin Level Description
0x00	Use the 'normal' read level for 1s
0x01	Apply the 'User' margin to the normal read-1 level
0x02	Apply the 'Factory' margin to the normal read-1 level

Table 18-18. Read 1s All Blocks Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid margin choice is specified	FSTAT[ACCERR]
Read-1s fails	FSTAT[MGSTAT0]

18.5.9.7 Read Once Command

The Read Once command provides read access to special 64-byte fields located in the program flash 0 IFR (see Program Flash IFR Map and Program Once Field). Access to the Program Once field is via 16 records (index values 0x00 - 0x0F), each 4 bytes long. These fields are programmed using the Program Once command described in Program Once Command.

Table 18-19. Read Once Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x41 (RDONCE)
1	Program Once record index (0x00 - 0x0F)
2	Not used
3	Not used
Returned Values	
4	Program Once byte 0 value
5	Program Once byte 1 value
6	Program Once byte 2 value
7	Program Once byte 3 value

After clearing CCIF to launch the Read Once command, a 4-byte Program Once record is read and stored in the FCCOB register. The CCIF flag is set after the Read Once operation completes. Valid record index values for the Read Once command range from 0x00 - 0x0F. During execution of the Read Once command, any attempt to read addresses within the program flash block containing the selected record index returns invalid data. The Read Once command can be executed any number of times.

Table 18-20. Read Once Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid record index is supplied	FSTAT[ACCERR]

18.5.9.8 Program Once Command

The Program Once command enables programming to special 64-byte fields in the program flash 0 IFR (see Program Flash IFR Map and Program Once Field). Access to the Program Once field is via 16 records (index values 0x00 - 0x0F), each 4 bytes long. These records can be read using the Read Once command (see Read Once Command) or using the Read Resource command (see Read Resource Command). These records can be programmed only once since the program flash 0 IFR cannot be erased.

Table 18-21. 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.

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 - 0x0F. During execution of the Program Once command, any attempt to read addresses within the program flash block containing the selected record index returns invalid data.

Table 18-22. Program Once Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid record index is supplied	FSTAT[ACCERR]
The requested record has already been programmed to a non-FFFF value[1]	FSTAT[ACCERR]
Any errors have been encountered during the verify operation	FSTAT[MGSTAT0]

1. If a Program Once record is initially programmed to 0xFFFF_FFFF, the Program Once command is allowed to execute again on that same record.

18.5.9.9 Erase All Blocks Command

The Erase All Blocks operation erases all flash memory, verifies all memory contents, and releases MCU security.

Table 18-23. Erase All Blocks Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x44 (ERSALL)

After clearing CCIF to launch the Erase All Blocks command, the flash memory module erases all program flash memory, then verifies that all are erased.

If the flash memory module verifies that all flash memories were properly erased, security is released by setting the FSEC[SEC] field to the unsecure state. The Erase All Blocks command aborts if any flash region is protected. The security byte and all other contents of the flash configuration field (see Flash Configuration Field Description) are erased by the Erase All Blocks command. If the erase-verify fails, the FSTAT[MGSTAT0] bit is set. The CCIF flag is set after the Erase All Blocks operation completes.

Table 18-24. Erase All Blocks Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
Any region of the program flash memory is protected	FSTAT[FPVIOL]
Any errors have been encountered during the verify operation[1]	FSTAT[MGSTAT0]

1. User margin read may be run using the Read 1s All Blocks command to verify all bits are erased.

18.5.9.9.1 Triggering an Erase All External to the Flash Memory Module

The functionality of the Erase All Blocks/Erase All Blocks Unsecure command is also available in an uncommanded fashion outside of the flash memory. Refer to the device's Chip Configuration details for information on this functionality.

Before invoking the external erase all function, the FSTAT[ACCERR and PVIOL] flags must be cleared and the FCCOB0 register must not contain 0x44. When invoked, the erase-all function erases all program flash memory regardless of the protection settings. If the post-erase verify passes, the routine then releases security by setting the FSEC[SEC] field register to the unsecure state. The security byte in the Flash Configuration Field is also programmed to the unsecure state. The status of the erase-all request is reflected in the FCNFG[ERSAREQ] bit. The FCNFG[ERSAREQ] bit is cleared once the operation completes and the normal FSTAT error reporting is available, except FPVIOL, as described in Erase All Blocks Command/Erase All Blocks Unsecure Command.

18.5.9.10 Verify Backdoor Access Key Command

The Verify Backdoor Access Key command only executes if the mode and security conditions are satisfied (see Flash Commands by Mode). Execution of the Verify Backdoor Access Key command is further qualified by the FSEC[KEYEN] bits. The Verify Backdoor Access Key command releases security if user-supplied keys in the FCCOB match those stored in the Backdoor Comparison Key bytes of the Flash

Configuration Field (see Flash Configuration Field Description). The column labelled Flash Configuration Field offset address shows the location of the matching byte in the Flash Configuration Field.

Table 18-25. Verify Backdoor Access Key Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]	Flash Configuration Field Offset Address
0	0x45 (VFYKEY)	
1-3	Not Used	
4	Key Byte 0	0x0_0003
5	Key Byte 1	0x0_0002
6	Key Byte 2	0x0_0001
7	Key Byte 3	0x0_0000
8	Key Byte 4	0x0_0007
9	Key Byte 5	0x0_0006
A	Key Byte 6	0x0_0005
B	Key Byte 7	0x0_0004

After clearing CCIF to launch the Verify Backdoor Access Key command, the flash memory module checks the FSEC[KEYEN] bits to verify that this command is enabled. If not enabled, the flash memory module sets the FSTAT[ACCERR] bit and terminates. If the command is enabled, the flash memory module compares the key provided in FCCOB to the backdoor comparison key in the Flash Configuration Field. If the backdoor keys match, the FSEC[SEC] field is changed to the unsecure state and security is released. If the backdoor keys do not match, security is not released and all future attempts to execute the Verify Backdoor Access Key command are immediately aborted and the FSTAT[ACCERR] bit is (again) set to 1 until a reset of the flash memory module module occurs. If the entire 8-byte key is all zeros or all ones, the Verify Backdoor Access Key command fails with an access error. The CCIF flag is set after the Verify Backdoor Access Key operation completes.

Table 18-26. Verify Backdoor Access Key Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
The supplied key is all-0s or all-Fs	FSTAT[ACCERR]
An incorrect backdoor key is supplied	FSTAT[ACCERR]
Backdoor key access has not been enabled (see the description of the FSEC register)	FSTAT[ACCERR]
This command is launched and the backdoor key has mismatched since the last power down reset	FSTAT[ACCERR]

18.5.9.11 Erase All Blocks Unsecure Command

The Erase All Blocks Unsecure operation erases all flash memory, verifies all memory contents, programs the security byte in the Flash Configuration Field to the unsecure state, and releases MCU security.

Table 18-27. Erase All Blocks Unsecure Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x49 (ERSALLU)

After clearing CCIF to launch the Erase All Blocks Unsecure command, the flash memory module erases all program flash memory, then verifies that all are erased.

If the flash memory module verifies that all program flash memory was properly erased, security is released by setting the FSEC[SEC] field to the unsecure state, and the security byte (see Flash Configuration Field Description) is programmed to the unsecure state by the Erase All Blocks Unsecure command. If the erase or program verify fails, the FSTAT[MGSTAT0] bit is set. The CCIF flag is set after the Erase All Blocks Unsecure operation completes.

Table 18-28. Erase All Blocks Unsecure Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
Any errors have been encountered during erase or program verify operations	FSTAT[MGSTAT0]

18.5.10 Security

The flash memory module provides security information to the MCU based on contents of the FSEC security register.

The MCU then limits access to flash memory resources as defined in the device's Chip Configuration details. During reset, the flash memory module initializes the FSEC register using data read from the security byte of the Flash Configuration Field (see Flash Configuration Field Description).

The following fields are available in the FSEC register. The settings are described in the Flash Security Register (FTFA_FSEC) details.

Flash security features are discussed further in AN4507: Using the Kinetis Security and Flash Protection Features also applicable for MWxx family. Note that not all features described in the application note are available on this device.

Table 18-29. FSEC register fields

FSEC field	Description
KEYEN	Backdoor Key Access
MEEN	Mass Erase Capability
FSLACC	Factory Security Level Access
SEC	MCU security

18.5.10.1 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.

18.5.10.1.1 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.

18.5.11 Reset Sequence

On each system reset the flash memory module executes a sequence which establishes initial values for the flash block configuration parameters, FPROT, FOPT, and FSEC registers.

FSTAT[CCIF] is cleared throughout the reset sequence. The flash memory module holds off CPU access during the reset sequence. Flash reads are possible when the hold is removed. Completion of the reset sequence is marked by setting CCIF which enables flash user commands.

If a reset occurs while any flash command is in progress, that command is immediately aborted. The state of the word being programmed or the sector/block being erased is not guaranteed. Commands and operations do not automatically resume after exiting reset.

Chapter 19
Computer Operating Properly (COP) Watchdog

19.1 Chip-specific information for this module

19.1.1 WCOP low power clocks

The COP_CTRL[CLKSEL] bitfield selects among the options for the clock source of the WCOP module's counter. For each value of the bitfield, the following table correlates between the module-level and chip-level names of the clock options.

Table 19-1. COP counter clock selection

COP_CTRL[CLKSEL] value	Clock input	Clock source
00b	lpo_clk[0]	8 MHz / 2 MHz IRC
01b	lpo_clk[1]	Crystal Oscillator
10b	lpo_clk[2]	IPS Bus clock[1]
11b	lpo_clk[3]	200 KHz IRC (Fosc200KHz)

1. Do not select the bus clock to clock the counter if the application requires the WCOP to wake the device from stop mode.

19.2 Introduction

The computer operating properly (COP) module is used to help software recover from runaway code. The COP is a free-running down counter that, once enabled, is designed to generate a reset upon reaching zero. Software must periodically service the COP in order to reload the counter and prevent a reset.

19.2.1 Features

The COP module includes these distinctive features:

- Programmable prescaler
- Programmable timeout period = (cop_prescaler*(TIMEOUT + 1)) clock cycles, where TIMEOUT can be from 0x0000 to 0xFFFF
- Programmable interrupt timing that can occur for any count less than the TIMEOUT value
- Programmable window timing to ensure that servicing doesn't occur too soon
- Programmable wait and stop operation
- COP timer is disabled while the DSC is in debug mode
- Causes loss of reference reset 128 cycles after loss of reference clock to the PLL is detected
- Choice of clock sources for counter
- Integrated low speed oscillator

19.2.2 Block Diagram

The block diagram of the COP module follows.

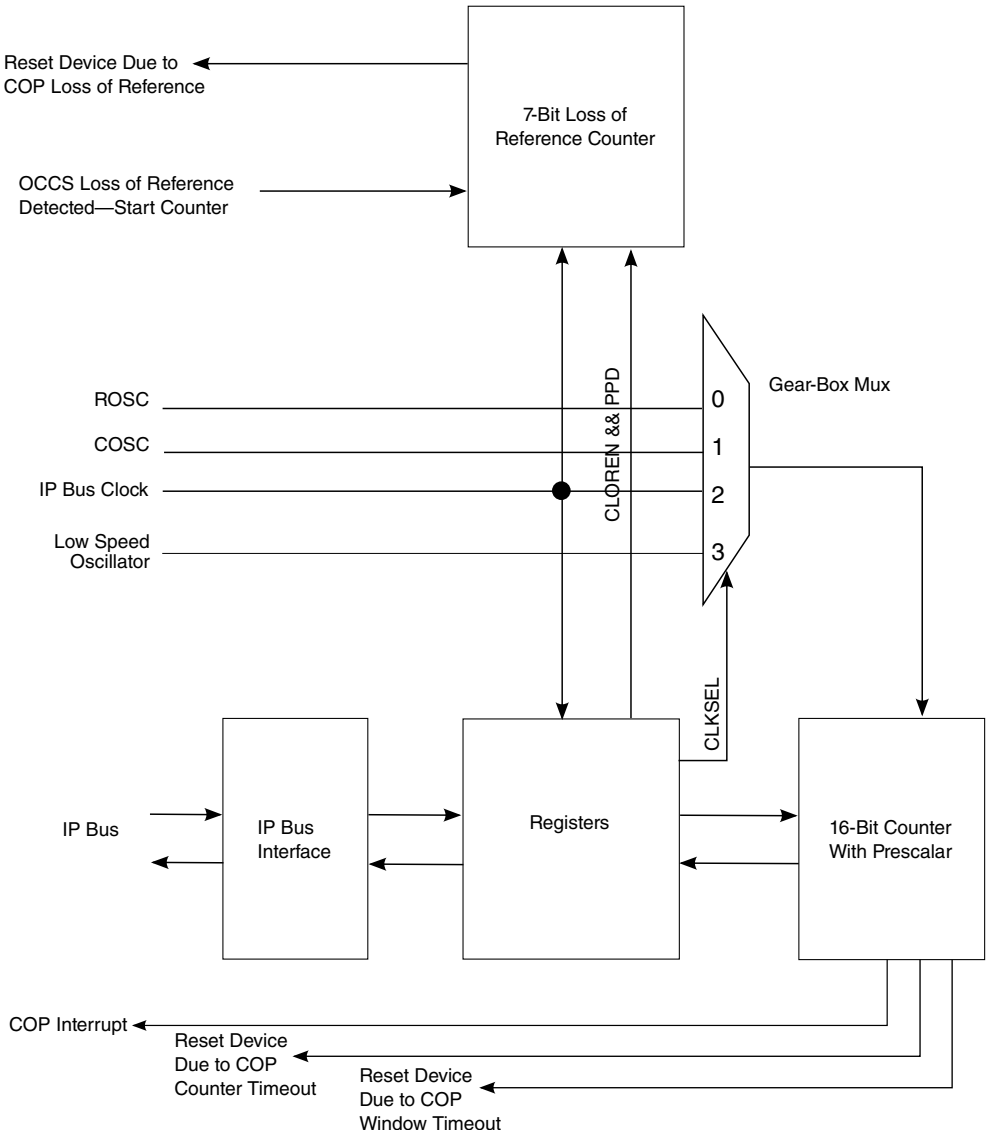


Figure 19-1. COP Module Block Diagram with Low Speed Clock

19.3 Memory Map and Registers

COP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E320	COP Control Register (COP_CTRL)	16	R/W	0302h	19.3.1/498
E321	COP Timeout Register (COP_TOUT)	16	R/W	FFFFh	19.3.2/499
E322	COP Counter Register (COP_CNTR)	16	R/W	FFFFh	19.3.3/500
E323	COP Interrupt Value Register (COP_INTVAL)	16	R/W	00FFh	19.3.4/501
E324	COP Window Timeout Register (COP_WINDOW)	16	R/W	FFFFh	19.3.5/501

19.3.1 COP Control Register (COP_CTRL)

Address: E320h base + 0h offset = E320h

Bit	15	14	13	12	11	10	9	8
Read	0						PSS	
Write								
Reset	0	0	0	0	0	0	1	1

Bit	7	6	5	4	3	2	1	0
Read Write	INTEN	CLKSEL		CLOREN	CSEN	CWEN	CEN	CWP
Reset	0	0	0	0	0	0	1	0

COP_CTRL field descriptions

Field	Description
15–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–8 PSS	Prescaler Select This two bit field determines the value of the clock divider (prescaler). You may divide the source clock by 1, 16, 256, or 1024. Generally, you use a lower prescaler value for lower frequency clock sources, but any combination of PSS and timeout may be used as long as they yield the desired timeout value. **Restriction:** This field can be changed only when CWP is set to zero. 00 No division 01 Divide by 16 10 Divide by 256 11 Divide by 1024
7 INTEN	Interrupt Enable This bit is used to enable an interrupt when the counter value reaches the value of the INTVAL register. Clear the interrupt by servicing the counter, disabling the COP, or clearing this bit. **Restriction:** This field can be changed only when CWP is set to zero. 0 COP interrupt is disabled. (default) 1 COP interrupt is enabled.
6–5 CLKSEL	Clock Source Select This bitfield selects the clock source for the COP counter. Some safety applications require the watchdog counter to use a clock source different than the system clock. **Restriction:** This field can be changed only when CWP is set to zero. It also should be changed only when CEN is clear. 00 Relaxation oscillator output (ROSC) is used to clock the counter (default) 01 Crystal oscillator output (COSC) is used to clock the counter 10 IP bus clock is used to clock the counter **Restriction:** Do not select the IP bus clock to clock the counter if the application requires the COP to wake the device from stop mode. 11 Low speed oscillator is used to clock the counter

Table continues on the next page...

COP_CTRL field descriptions (continued)

Field	Description
4 CLOREN	COP Loss of Reference Enable This bit enables the operation of the COP loss of reference counter. **Restriction:** This bit can be changed only when CWP is set to zero. 0 COP loss of reference counter is disabled. (default) 1 COP loss of reference counter is enabled.
3 CSEN	COP Stop Mode Enable This bit controls the operation of the COP counter in stop mode. **Restriction:** This bit can be changed only when CWP is set to zero. 0 COP counter stops in stop mode. (default) 1 COP counter runs in stop mode if CEN is set to one.
2 CWEN	COP Wait Mode Enable This bit controls the operation of the COP counter in wait mode. **Restriction:** This bit can be changed only when CWP is set to zero. 0 COP counter stops in wait mode. (default) 1 COP counter runs in wait mode if CEN is set to one.
1 CEN	COP Enable This bit controls the operation of the COP counter. This bit always reads as zero when the chip is in debug mode. **Restriction:** This bit can be changed only when CWP is set to zero. 0 COP counter is disabled. 1 COP counter is enabled. (default)
0 CWP	COP Write Protect This bit controls the write protection feature of the COP control (CTRL) register, the COP interrupt value (INTVAL) register, the COP window (WINDOW) register and the COP timeout (TOUT) register. Once set, this bit can be cleared only by resetting the module. 0 The CTRL, INTVAL, WINDOW and TOUT registers are readable and writable. (default) 1 The CTRL, INTVAL, WINDOW and TOUT registers are read-only.

19.3.2 COP Timeout Register (COP_TOUT)

The value in this register determines the timeout period of the COP counter.

Considerations about setting the timeout value follow:

1. TIMEOUT should be written before the COP is enabled. Changing TIMEOUT while the COP is enabled results in a timeout period that differs from the expected value.
2. After the COP has been enabled:

- The recommended procedure for changing TIMEOUT is to disable the COP, write to TOUT, and then re-enable the COP. This procedure ensures that the new TIMEOUT is loaded into the counter.
- Alternatively, the CPU can write to TOUT and then write the proper patterns to CNTR to cause the counter to reload with the new TIMEOUT value.

3. A minimum time difference is required between the TIMEOUT value and the value of INTVAL[INTERRUPT_VALUE]. When the bus clock is the source for the COP counter, a typical minimum time difference is 40 cycles.

Address: E320h base + 1h offset = E321h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	TIMEOUT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

COP_TOUT field descriptions

Field	Description
TIMEOUT	COP Timeout Period The value in this register determines the timeout period of the COP counter. **Restriction:** These bits can be changed only when CWP is set to zero.

19.3.3 COP Counter Register (COP_CNTR)

Address: E320h base + 2h offset = E322h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	COUNT_SERVICE															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

COP_CNTR field descriptions

Field	Description
COUNT_ SERVICE	COP Count/Service COP Count: When this register is read, its value is the current value of the COP counter as it counts down from the timeout value to zero. A reset is issued when this count reaches zero or when the counter is serviced when the count is greater than the WINDOW value. COP Service: Write to this register to service the counter. When enabled, the COP requires that a service sequence be performed periodically to clear the COP counter and prevent a reset from being issued. • This routine consists of writing 0x5555 to CNTR followed by writing 0xAAAA before the timeout period expires. • These writes to CNTR must be performed in the correct order, but any number of other instructions (and writes to other registers) may be executed between the two writes.

19.3.4 COP Interrupt Value Register (COP_INTVAL)

Address: E320h base + 3h offset = E323h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	INTERRUPT_VALUE															
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

COP_INTVAL field descriptions

Field	Description
INTERRUPT_ VALUE	COP Interrupt Value When the count value is equal to this interrupt value, an interrupt is generated if it is enabled via the CTRL[INTEN] bit. The interrupt can be cleared by perfoming the service routine to the counter, by disabling the COP (setting the CTRL[CEN] bit to 0), or by clearing the interrupt enable (setting the CTRL[INTEN] bit to 0). Servicing the counter requires time to resynchronize to the clock used by the counter. Allow time for this resynchronization to occur before exiting the interrupt service routine; otherwise, a new interrupt may be issued for the same event. To confirm that the resychronization has occurred, verify that CNTR is greater than INTVAL prior to exiting the ISR. **Restriction:** Do not change this field's value while the counter is enabled. **Restriction:** These bits can be changed only when the CTRL[CWP] bit is 0.

19.3.5 COP Window Timeout Register (COP_WINDOW)

Address: E320h base + 4h offset = E324h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	WINDOW_VALUE															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

COP_WINDOW field descriptions

Field	Description
WINDOW_ VALUE	COP Window Timeout Value This register specifies an upper bound on the CNTR value that must be crossed prior to the CNTR being serviced. If CNTR is above this value when a service occurs, then a COP window reset is generated. If the CNTR value is less than or equal to this value at the time of the service, then the service is allowed to occur normally. This function can be used to ensure that the software not only services the CNTR within a certain time period, but also not until a minimum period has passed. This further verifies that the code is executing properly and not just looping on the service routine. Leaving this register at its default value of 0xFFFF effectively disables the window function since the CNTR value will always be equal to or less than the WINDOW value. **Restriction:** Do not change this field's value while the counter is enabled.

COP_WINDOW field descriptions (continued)

Field	Description
	Restriction: These bits can be changed only when the CTRL[CWP] bit is 0.

19.4 Functional Description

When the COP is enabled, each positive edge of the prescaled clock (COSC, ROSC, low speed oscillator, or IP Bus clock) causes the counter to decrement by one. If the count reaches a value equal to the INTVAL register, then an interrupt may be issued. If the count reaches a value of 0x0000, then the device is reset. For the DSC core to show that it is operating properly, it must perform a service routine before the count reaches 0x0000 but after the count reaches the WINDOW value. The service routine consists of writing 0x5555 followed by 0xAAAA to the CNTR register. This service routine also clears the interrupt.

19.4.1 COP after Reset

CEN is set out of reset. Thus the counter is enabled by default. In addition, the TOUT register is set to its maximum value of 0xFFFF and PRESCALER is set to 1024 (CTRL[PSS]=11) during reset so the counter is loaded with a maximum timeout period when reset is released.

If the IP bus clock to the COP is not enabled by default after reset, then allow 2 clock cycles to occur after enabling it before performing a write access to the COP.

19.4.2 Wait Mode Operation

If wait mode is entered with both CEN and CWEN set to 1, then the COP counter continues to count down. In that case, a COP reset is issued to wake the device after the counter reaches zero. A COP interrupt may wake the device prior to the counter reaching zero if the interrupt is properly programmed and enabled. If either CEN or CWEN is cleared to 0 when wait mode is entered, then the counter is disabled and reloads using the value in the TOUT register.

19.4.3 Stop Mode Operation

If stop mode is entered with both CEN and CSEN set to 1, then the COP counter continues to count down. In that case, a COP reset is issued to wake the device after the counter reaches zero. A COP interrupt may wake the device prior to the counter reaching zero if the interrupt is properly programmed and enabled. If either CEN or CSEN is cleared to 0 when stop mode is entered, then the counter is disabled and reloads using the value in the TOUT register.

19.4.4 Debug Mode Operation

The COP counter is not allowed to count when the device is in debug mode. In addition, the CEN bit in the CTRL register always reads as zero when the device is in debug. The actual value of CEN is unaffected by debug, however, and resumes its previously set value upon exiting debug.

19.4.5 Loss of Reference Operation

When the OCCS signals the COP that a loss of the reference clock has occurred and the CLOREN bit is set, then the COP starts a 7-bit counter that runs off of the IP bus clock (which continues to be produced by the PLL for at least 1000 cycles upon losing its reference). The counter continues to count even if the loss of reference clock condition goes away. When this counter reaches 0x7F, it causes a loss of reference reset that resets the entire device. If the software has safely shut down the device and does not want a full reset, then the loss of reference timeout count can be delayed by servicing the COP counter in the standard manner of writing 0x5555 followed by 0xAAAA or stopped by setting CLOREN to zero.

19.5 Resets

Any system reset forces all registers to their reset state and clears the COP_RST_B, COP_WNDW_RST_B, or LOR_RST_B signals if they are asserted. The counter is loaded with its maximum value of 0xFFFF, the prescaler is set to 1024, and the counter restarts when reset is released because CEN is enabled by default.

19.6 Clocks

The COP timer base is the COSC, ROSC, low speed oscillator, or IP Bus clock divided by the prescaler value (1 - 1024).

19.7 Interrupts

The COP module generates a single interrupt that occurs when the counter matches the value of the INTVAL register. Enabling this interrupt is controlled by CTRL[INTEN].

The COP module generates the chip-wide COP reset signal when the counter reaches a value of 0x0000. The COP module generates the chip-wide COP window reset (COP_WNDW_RST_B) signal when a servicing routine occurs prior to the counter reaching the WINDOW value. It can also generate a chip-wide reset signal 128 IP Bus cycles after the loss of the reference clock is detected.

NOTE

The chip reset vector base address is located at P:00h. The COP reset vector base address is located at P:02h. For more details, refer to the interrupt vector table.

Chapter 20
External Watchdog Monitor (EWM)

20.1 Chip-specific information for this module

20.1.1 EWM low power clocks

The EWM_CLKCTRL[CLKSEL] bitfield selects among the options for the EWM's low power clock source. For each value of the bitfield, the following table correlates between the module-level and chip-level names of the clock options.

Table 20-1. EWM counter clock selection

EWM_CLKCTRL[CLKSEL] value	Clock input	Clock source
00b	lpo_clk[0]	8 MHz / 2MHz IRC
01b	lpo_clk[1]	Crystal Oscillator
10b	lpo_clk[2]	IPS Bus clock
11b	lpo_clk[3]	200 KHz IRC (Fosc200KHz)

20.1.2 EWM_OUT_b pin state in Low Power Modes

During Wait and Stop modes, the EWM_OUT_b pin enters a high-impedance state. A user has the option to control the logic state of the pin using an external pull device or by configuring the internal pull device. When the CPU enters a Run mode from Wait or Stop recovery, the pin resumes the state it had prior to the entry to Wait or Stop mode.

20.2 Introduction

For safety, a redundant watchdog system, External Watchdog Monitor (EWM), is designed to monitor external circuits, as well as the MCU software flow. This provides a back-up mechanism to the internal watchdog that resets the MCU's CPU and peripherals.

The watchdog is generally used to monitor the flow and execution of embedded software within an MCU. The watchdog consists of a counter that if allowed to overflow, forces an internal reset (asynchronous) to all on-chip peripherals and optionally assert the $\overline{\text{RESET}}$ pin to reset external devices/circuits. The overflow of the watchdog counter must not occur if the software code works well and services the watchdog to re-start the actual counter.

The EWM differs from the internal watchdog in that it does not reset the MCU's CPU and peripherals. The EWM provides an independent $\overline{\text{EWM\_out}}$ signal that when asserted resets or places an external circuit into a safe mode. The $\overline{\text{EWM\_out}}$ signal is asserted upon the EWM counter time-out. An optional external input EWM_in is provided to allow additional control of the assertion of $\overline{\text{EWM\_out}}$ signal.

20.2.1 Features

Features of EWM module include:

- Independent LPO_CLK clock source
- Programmable time-out period specified in terms of number of EWM LPO_CLK clock cycles.
- Windowed refresh option
  - Provides robust check that program flow is faster than expected.
  - Programmable window.
  - Refresh outside window leads to assertion of $\overline{\text{EWM\_out}}$.
- Robust refresh mechanism
  - Write values of 0xB4 and 0x2C to EWM Refresh Register within 15 (*EWM_refresh_time*) peripheral bus clock cycles.

- One output port, $\overline{\text{EWM\_out}}$, when asserted is used to reset or place the external circuit into safe mode.
- One Input port, EWM_in, allows an external circuit to control the assertion of the $\overline{\text{EWM\_out}}$ signal.

20.2.2 Modes of Operation

This section describes the module's operating modes.

20.2.2.1 Stop Mode

When the EWM is in stop mode, the CPU refreshes to the EWM cannot occur. On entry to stop mode, the EWM's counter freezes.

There are two possible ways to exit from Stop mode:

- On exit from stop mode through a reset, the EWM remains disabled.
- On exit from stop mode by an interrupt, the EWM is re-enabled, and the counter continues to be clocked from the same value prior to entry to stop mode.

Note the following if the EWM enters the stop mode during CPU refresh mechanism: At the exit from stop mode by an interrupt, refresh mechanism state machine starts from the previous state which means, if first refresh command is written correctly and EWM enters the stop mode immediately, the next command has to be written within the next 15 (*EWM_refresh_time*) peripheral bus clocks after exiting from stop mode. User must mask all interrupts prior to executing EWM refresh instructions.

20.2.2.2 Wait Mode

The EWM module treats the stop and wait modes as the same. EWM functionality remains the same in both of these modes.

20.2.2.3 Debug Mode

Entry to debug mode has no effect on the EWM.

- If the EWM is enabled prior to entry of debug mode, it remains enabled.
- If the EWM is disabled prior to entry of debug mode, it remains disabled.

20.2.3 Block Diagram

This figure shows the EWM block diagram.

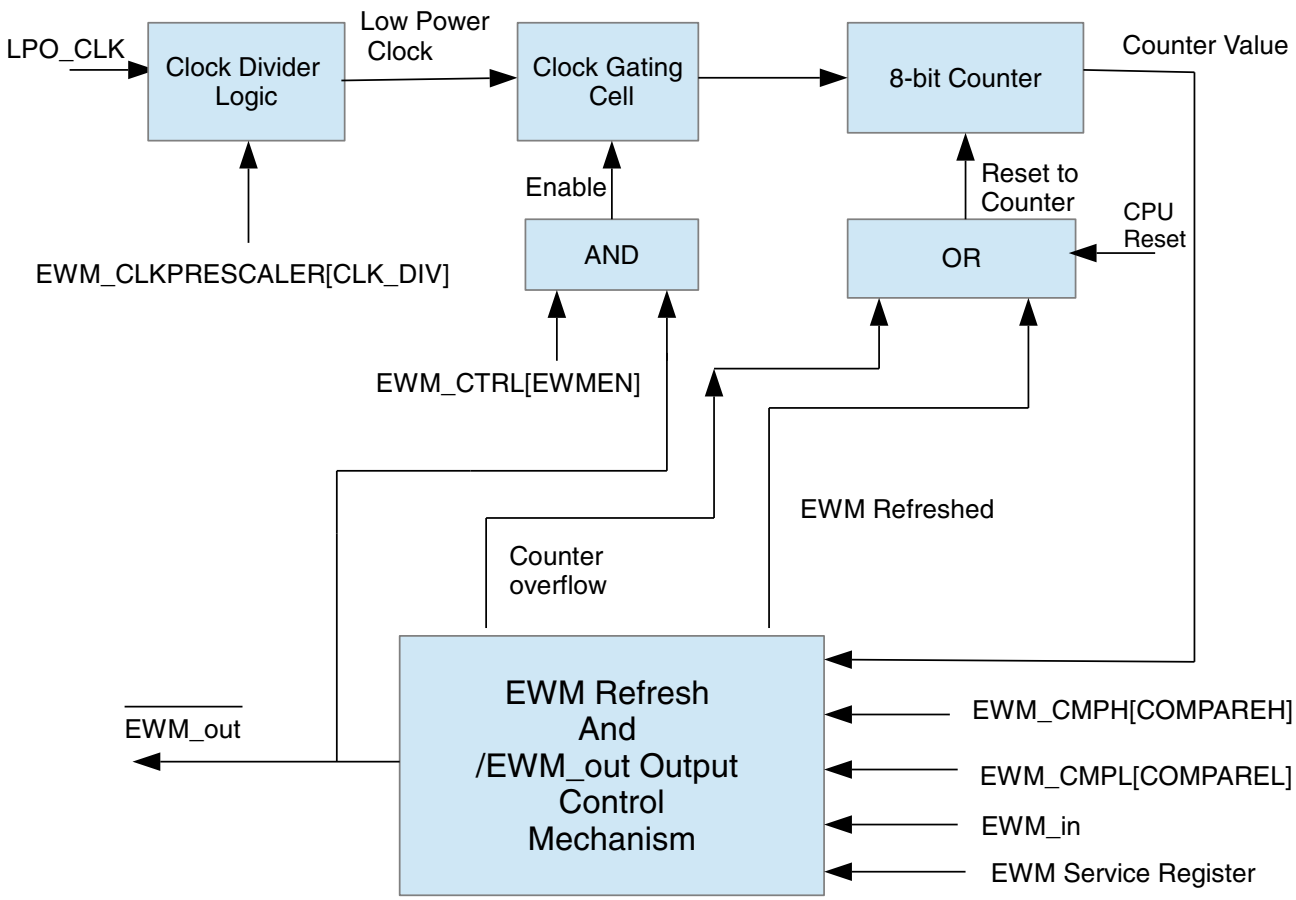


Figure 20-1. EWM Block Diagram

20.3 EWM Signal Descriptions

The EWM has two external signals and internal options for the counter clock sources, as shown in the following table.

Table 20-2. EWM Signal Descriptions

Signal	Description	I/O
EWM_in	EWM input for safety status of external safety circuits. The polarity of EWM_in is programmable using the EWM_CTRL[ASSIN] bit. The default polarity is active-low.	I
$\overline{\text{EWM\_out}}$	EWM reset out signal	O
lpo_clk[3:0]	Low power clock sources for running counter	I

20.4 Memory Map/Register Definition

This section contains the module memory map and registers.

NOTE

Each 8-bit register occupies bits 0-7 of a 16-bit width. The other 8 bits are read-only and always read 0.

EWM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E330	Control Register (EWM_CTRL)	16	R/W	0000h	20.4.1/509
E331	Service Register (EWM_SERV)	16	W (always reads 0)	0000h	20.4.2/510
E332	Compare Low Register (EWM_CMPL)	16	R/W	0000h	20.4.3/511
E333	Compare High Register (EWM_CMPH)	16	R/W	00FFh	20.4.4/511
E334	Clock Control Register (EWM_CLKCTRL)	16	R/W	0000h	20.4.5/512
E335	Clock Prescaler Register (EWM_CLKPRESCALER)	16	R/W	0000h	20.4.6/513

20.4.1 Control Register (EWM_CTRL)

The CTRL register is cleared by any reset.

NOTE

INEN, ASSIN and EWMEN bits can be written once after a CPU reset. Modifying these bits more than once, generates a bus transfer error.

Address: E330h base + 0h offset = E330h

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0				INTEN	INEN	ASSIN	EWMEN
Write								
Reset	0	0	0	0	0	0	0	0

EWM_CTRL field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 INTEN	Interrupt Enable. This bit when set and $\overline{\text{EWM\_out}}$ is asserted, an interrupt request is generated. To de-assert interrupt request, user should clear this bit by writing 0.
2 INEN	Input Enable. This bit when set, enables the EWM_in port.
1 ASSIN	EWM_in's Assertion State Select. Default assert state of the EWM_in signal is logic zero. Setting the ASSIN bit inverts the assert state of EWM_in signal to a logic one.
0 EWMEN	EWM enable. This bit when set, enables the EWM module. This resets the EWM counter to zero and deasserts the $\overline{\text{EWM\_out}}$ signal. This bit when unset, keeps the EWM module disabled. It cannot be re-enabled until a next reset, due to the write-once nature of this bit.

20.4.2 Service Register (EWM_SERV)

The SERV register provides the interface from the CPU to the EWM module. It is write-only and reads of this register return zero.

Address: E330h base + 1h offset = E331h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								0							
Write									SERVICE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EWM_SERV field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SERVICE	The EWM refresh mechanism requires the CPU to write two values to the SERV register: a first data byte of 0xB4, followed by a second data byte of 0x2C. The EWM refresh is invalid if either of the following conditions is true. • The first or second data byte is not written correctly. • The second data byte is not written within a fixed number of peripheral bus cycles of the first data byte. This fixed number of cycles is called *EWM_refresh_time*.

20.4.3 Compare Low Register (EWM_CMPL)

The CMPL register is reset to zero after a CPU reset. This provides no minimum time for the CPU to refresh the EWM counter.

NOTE

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

Address: E330h base + 2h offset = E332h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								COMPAREL							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EWM_CMPL field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COMPAREL	To prevent runaway code from changing this field, software should write to this field after a CPU reset even if the (default) minimum refresh time is required.

20.4.4 Compare High Register (EWM_CMPH)

The CMPH register is reset to 0x00FF after a CPU reset. This provides a maximum of 256 clocks time, for the CPU to refresh the EWM counter.

NOTE

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

NOTE

The valid values for CMPH are up to 0x00FE because the EWM counter never expires when CMPH = 0x00FF. The expiration happens only if EWM counter is greater than CMPH.

Address: E330h base + 3h offset = E333h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								COMPAREH							
Write																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

EWM_CMPH field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COMPAREH	To prevent runaway code from changing this field, software should write to this field after a CPU reset even if the (default) maximum refresh time is required.

20.4.5 Clock Control Register (EWM_CLKCTRL)

This CLKCTRL register is reset to 0x00 after a CPU reset.

NOTE

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

NOTE

User should select the required low power clock before enabling the EWM.

Address: E330h base + 4h offset = E334h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								0						CLKSEL	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EWM_CLKCTRL field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

EWM_CLKCTRL field descriptions (continued)

Field	Description
CLKSEL	EWM has 4 possible low power clock sources for running EWM counter. One of the clock source can be selected by writing into this field. 00 lpo_clk[0] will be selected for running EWM counter. 01 lpo_clk[1] will be selected for running EWM counter. 10 lpo_clk[2] will be selected for running EWM counter. 11 lpo_clk[3] will be selected for running EWM counter.

20.4.6 Clock Prescaler Register (EWM_CLKPRESCALER)

This CLKPRESCALER register is reset to 0x00 after a CPU reset.

NOTE

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

NOTE

Write the required prescaler value before enabling the EWM.

NOTE

The implementation of this register is chip-specific. See the chip-specific information for details.

Address: E330h base + 5h offset = E335h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								CLK_DIV							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EWM_CLKPRESCALER field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLK_DIV	Selected low power clock source for running the EWM counter can be prescaled as below. • Prescaled clock frequency = low power clock source frequency / (1 + CLK_DIV)

20.5 Functional Description

The following sections describe functional details of the EWM module.

NOTE

When the BUS_CLK is lost, then EWM module doesn't generate the $\overline{\text{EWM\_out}}$ signal and no refresh operation is possible

20.5.1 The $\overline{\text{EWM\_out}}$ Signal

The $\overline{\text{EWM\_out}}$ is a digital output signal used to gate an external circuit (application specific) that controls critical safety functions. For example, the $\overline{\text{EWM\_out}}$ could be connected to the high voltage transistors circuits.

The $\overline{\text{EWM\_out}}$ signal remains deasserted when the EWM is being regularly refreshed by the CPU within the programmable refresh window, indicating that the application code is executed as expected.

The $\overline{\text{EWM\_out}}$ signal is asserted in any of the following conditions:

- The EWM refresh occurs when the counter value is less than CMPL value.
- The EWM counter value reaches the CMPH value, and no EWM refresh has occurred.
- If functionality of EWM_in pin is enabled and EWM_in pin is asserted while refreshing the EWM.
- After any reset (by the virtue of the external pull-down mechanism on the $\overline{\text{EWM\_out}}$ pin)

The $\overline{\text{EWM\_out}}$ is asserted after any reset by the virtue of the external pull-down mechanism on the $\overline{\text{EWM\_out}}$ signal. Then, to deassert the $\overline{\text{EWM\_out}}$ signal, set EWMEN bit in the CTRL register to enable the EWM.

If the $\overline{\text{EWM\_out}}$ signal shares its pad with a digital I/O pin, on reset this actual pad defers to being an input signal. The pad state is controlled by the $\overline{\text{EWM\_out}}$ signal only after the EWM is enabled by the EWMEN bit in the CTRL register.

Note

$\overline{\text{EWM\_out}}$ pad must be in pull down state when EWM functionality is used and when EWM is under Reset.

20.5.2 The EWM_in Signal

The EWM_in is a digital input signal for safety status of external safety circuits, that allows an external circuit to control the assertion of the EWM_out signal. For example, in the application, an external circuit monitors a critical safety function, and if there is fault with safety function, the external circuit can then actively initiate the EWM_out signal that controls the gating circuit.

The EWM_in signal is ignored if the EWM is disabled, or if INEN bit of CTRL register is cleared, as after any reset.

On enabling the EWM (setting the CTRL[EWMEN] bit) and enabling EWM_in functionality (setting the CTRL[INEN] bit), the EWM_in signal must be in the deasserted state prior to the CPU start refreshing the EWM. This ensures that the EWM_out stays in the deasserted state; otherwise, the EWM_out output signal is asserted.

Note

The user must update the CMPH and CMPL registers prior to enabling the EWM. After enabling the EWM, the counter resets to zero, therefore the user shall provide a reasonable time after a power-on reset for the external monitoring circuit to stabilize. The user shall also ensure that the EWM_in pin is deasserted.

20.5.3 EWM Counter

It is an 8-bit ripple counter fed from a clock source that is independent of the peripheral bus clock source. As the preferred time-out is between 1 ms and 100 ms the actual clock source should be in the kHz range.

The counter is reset to zero after the CPU reset, or when EWM refresh action completes, or at counter overflow. The counter value is not accessible to the CPU.

20.5.4 EWM Compare Registers

The compare registers CMPL and CMPH are write-once after a CPU reset and cannot be modified until another CPU reset occurs.

The EWM compare registers are used to create a refresh window to refresh the EWM module.

It is illegal to program CMPL and CMPH with same value. In this case, as soon as counter reaches (CMPL + 1), EWM_out is asserted.

20.5.5 EWM Refresh Mechanism

Other than the initial configuration of the EWM, the CPU can only access the EWM by the EWM Service Register. The CPU must access the EWM service register with correct write of unique data within the windowed time frame as determined by the CMPL and CMPH registers for correct EWM refresh operation. Therefore, three possible conditions can occur:

Table 20-3. EWM Refresh Mechanisms

Condition	Mechanism
An EWM refresh action completes when: CMPL < Counter < CMPH.	The software behaves as expected and the EWM counter is reset to zero. The $\overline{\text{EWM\_out}}$ output signal remains in the deasserted state if, during the EWM refresh action, the EWM_in input has been in deasserted state..
An EWM refresh action completes when Counter < CMPL	The software refreshes the EWM before the windowed time frame, the counter is reset to zero and the $\overline{\text{EWM\_out}}$ output signal is asserted irrespective of the input $\overline{\text{EWM\_in}}$.
Counter value reaches CMPH prior to completion of EWM refresh action.	Software has not refreshed the EWM. The EWM counter is reset to zero and the $\overline{\text{EWM\_out}}$ output signal is asserted irrespective of the input $\overline{\text{EWM\_in}}$.

20.5.6 EWM Interrupt

When $\overline{\text{EWM\_out}}$ is asserted, an interrupt request is generated to indicate the assertion of the EWM reset out signal. This interrupt is enabled when CTRL[INTEN] is set. Clearing this bit clears the interrupt request but does not affect $\overline{\text{EWM\_out}}$. The $\overline{\text{EWM\_out}}$ signal can be deasserted only by forcing a system reset.

20.5.7 Selecting the EWM counter clock

There are four possible low power clock sources for the EWM counter. Select one of the available clock sources by programming CLKCTRL[CLKSEL].

20.5.8 Counter clock prescaler

The EWM counter clock source can be prescaled by a clock divider, by programming CLKPRESCALER[CLK_DIV]. This divided clock is used to run the EWM counter.

NOTE

The divided clock used to run the EWM counter must be no more than half the frequency of the bus clock.

Chapter 21
Cyclic Redundancy Check (CRC)

21.1 Introduction

The cyclic redundancy check (CRC) module generates 16/32-bit CRC code for error detection.

The CRC module provides a programmable polynomial and other parameters required to implement a 16-bit or 32-bit CRC standard.

The 16/32-bit code is calculated for 32 bits of data at a time.

21.1.1 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

21.1.2 Block diagram

The following is a block diagram of the CRC.

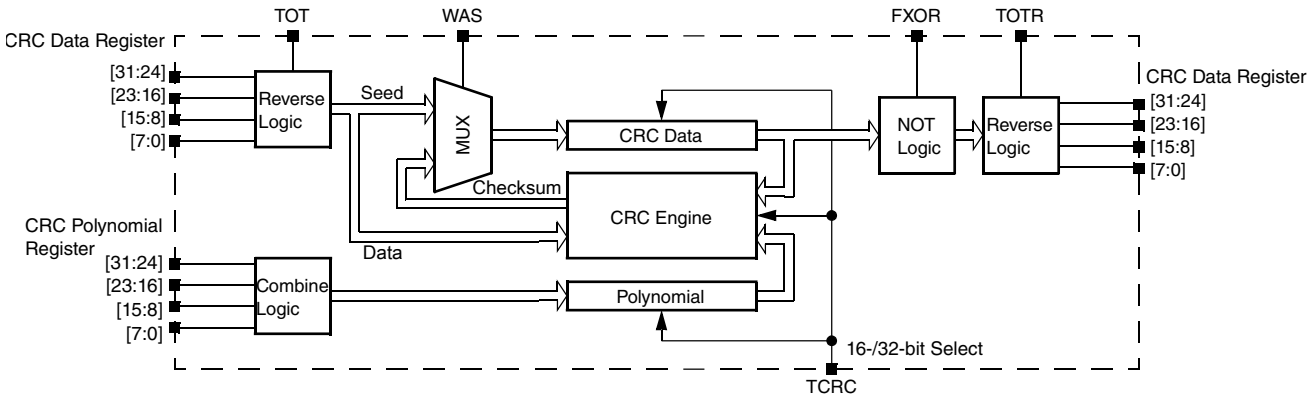


Figure 21-1. Programmable cyclic redundancy check (CRC) block diagram

21.1.3 Modes of operation

Various MCU modes affect the CRC module's functionality.

21.1.3.1 Run mode

This is the basic mode of operation.

21.1.3.2 Low-power modes (Wait or Stop)

Any CRC calculation in progress stops when the MCU enters a low-power mode that disables the module clock. It resumes after the clock is enabled or via the system reset for exiting the low-power mode. Clock gating for this module is dependent on the MCU.

21.2 Memory map and register descriptions

CRC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E3A0	CRC Data register (CRC_DATA)	32	R/W	FFFF_FFFFh	21.2.1/521
E3A2	CRC Polynomial register (CRC_GPOLY)	32	R/W	0000_1021h	21.2.2/522
E3A4	CRC Control register (CRC_CTRL)	32	R/W	0000_0000h	21.2.3/523

21.2.1 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.

NOTE

The address for this module on this chip is presented in terms on 16-bit words. However, in the case of an 8-bit write to this register, use its address in terms of bytes (which is double the value of the 16-bit word address).

Address: E3A0h base + 0h offset = E3A0h

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.

Table continues on the next page...

CRC_DATA field descriptions (continued)

Field	Description
15–8 LU	CRC Low Upper Byte When CTRL[WAS] is 1, values written to this field are part of the seed value. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation.
LL	CRC Low Lower Byte When CTRL[WAS] is 1, values written to this field are part of the seed value. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation.

21.2.2 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: E3A0h base + 2h offset = E3A2h

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	HIGH																LOW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1

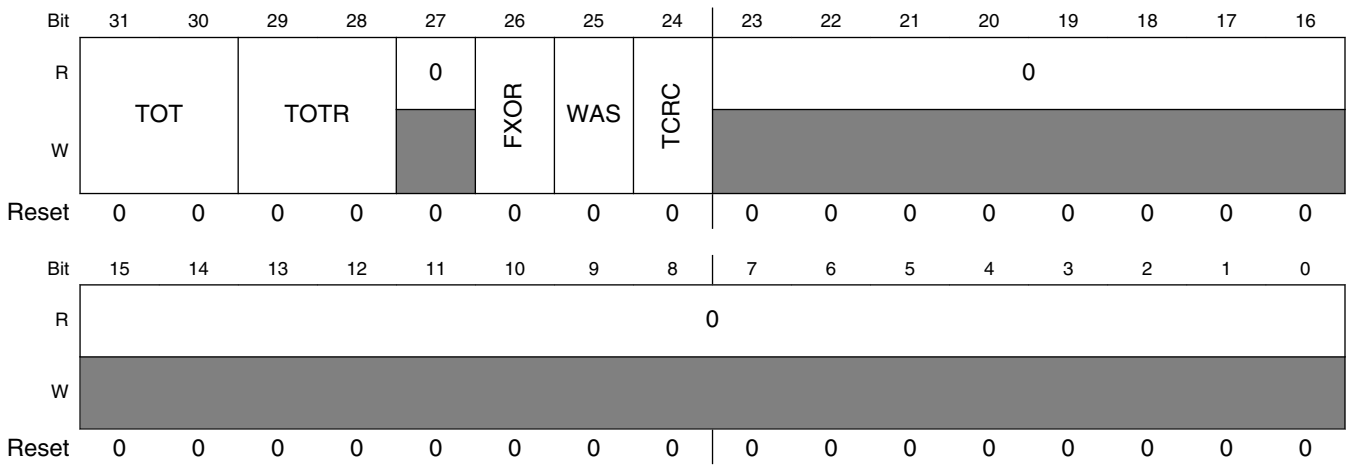
CRC_GPOLY field descriptions

Field	Description
31–16 HIGH	High Polynominal Half-word Writable and readable in 32-bit CRC mode (CTRL[TCRC] is 1). This field is not writable in 16-bit CRC mode (CTRL[TCRC] is 0).
LOW	Low Polynominal Half-word Writable and readable in both 32-bit and 16-bit CRC modes.

21.2.3 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: E3A0h base + 4h offset = E3A4h



CRC_CTRL field descriptions

Field	Description
31–30 TOT	Type Of Transpose For Writes Defines the transpose configuration of the data written to the CRC data register. See the description of the transpose feature for the available transpose options. 00 No transposition. 01 Bits in bytes are transposed; bytes are not transposed. 10 Both bits in bytes and bytes are transposed. 11 Only bytes are transposed; no bits in a byte are transposed.
29–28 TOTR	Type Of Transpose For Read Identifies the transpose configuration of the value read from the CRC Data register. See the description of the transpose feature for the available transpose options. 00 No transposition. 01 Bits in bytes are transposed; bytes are not transposed. 10 Both bits in bytes and bytes are transposed. 11 Only bytes are transposed; no bits in a byte are transposed.
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 FXOR	Complement Read Of CRC Data Register Some CRC protocols require the final checksum to be XORed with 0xFFFFFFFF or 0xFFFF. Asserting this bit enables on the fly complementing of read data.

Table continues on the next page...

CRC_CTRL field descriptions (continued)

Field	Description
	0 No XOR on reading. 1 Invert or complement the read value of the CRC Data register.
25 WAS	Write CRC Data Register As Seed When asserted, a value written to the CRC data register is considered a seed value. When deasserted, a value written to the CRC data register is taken as data for CRC computation. 0 Writes to the CRC data register are data values. 1 Writes to the CRC data register are seed values.
24 TCRC	Width of CRC protocol. 0 16-bit CRC protocol. 1 32-bit CRC protocol.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

21.3 Functional description

21.3.1 CRC initialization/reinitialization

To enable the CRC calculation, the user must program CRC_CTRL[WAS], CRC_GPOLY,necessary parameters for transposition and CRC result inversion in the applicable registers. Asserting CRC_CTRL[WAS] enables the programming of the seed value into the CRC_DATA register.

After a completed CRC calculation, the module can be reinitialized for a new CRC computation by reasserting CRC_CTRL[WAS] and programming a new, or previously used, seed value. All other parameters must be set before programming the seed value and subsequent data values.

21.3.2 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.

21.3.2.1 16-bit CRC

To compute a 16-bit CRC:

1. Clear CRC_CTRL[TCRC] to enable 16-bit CRC mode.
2. Program the transpose and complement options in the CTRL register as required for the CRC calculation. See Transpose feature and CRC result complement for details.
3. Write a 16-bit polynomial to the CRC_GPOLY[LOW] field. The CRC_GPOLY[HIGH] field is not usable in 16-bit CRC mode.
4. Set CRC_CTRL[WAS] to program the seed value.
5. Write a 16-bit seed to CRC_DATA[LU:LL]. CRC_DATA[HU:HL] are not used.
6. Clear CRC_CTRL[WAS] to start writing data values.
7. Write data values into CRC_DATA[HU:HL:LU:LL]. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC_DATA[LU:LL].
8. When all values have been written, read the final CRC result from CRC_DATA[LU:LL].

Transpose and complement operations are performed on the fly while reading or writing values. See Transpose feature and CRC result complement for details.

21.3.2.2 32-bit CRC

To compute a 32-bit CRC:

1. Set CRC_CTRL[TCRC] to enable 32-bit CRC mode.
2. Program the transpose and complement options in the CTRL register as required for the CRC calculation. See Transpose feature and CRC result complement for details.
3. Write a 32-bit polynomial to CRC_GPOLY[HIGH:LOW].
4. Set CRC_CTRL[WAS] to program the seed value.
5. Write a 32-bit seed to CRC_DATA[HU:HL:LU:LL].
6. Clear CRC_CTRL[WAS] to start writing data values.
7. Write data values into CRC_DATA[HU:HL:LU:LL]. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC_DATA[HU:HL:LU:LL].
8. When all values have been written, read the final CRC result from CRC_DATA[HU:HL:LU:LL]. The CRC is calculated bytewise, and two clocks are required to complete one CRC calculation.

Transpose and complement operations are performed on the fly while reading or writing values. See Transpose feature and CRC result complement for details.

21.3.3 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.

21.3.3.1 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]}

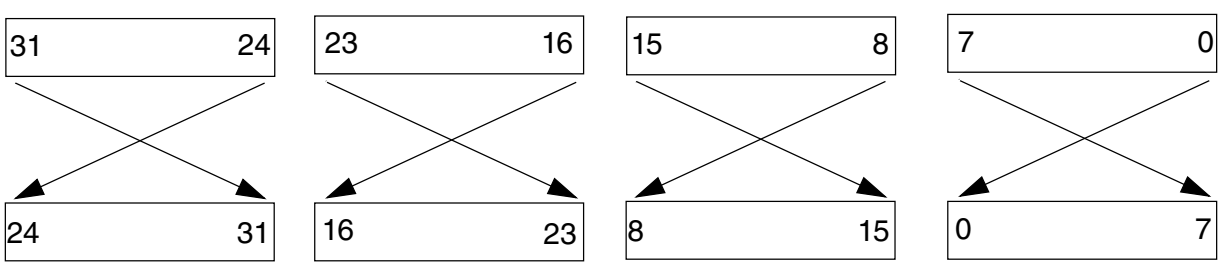


Figure 21-2. Transpose type 01

3. CTRL[TOT] or CTRL[TOTR] is 10.

   Both bits in bytes and bytes are transposed.

   reg[31:0] becomes = {reg[0:7], reg[8:15],reg[16:23], reg[24:31]}

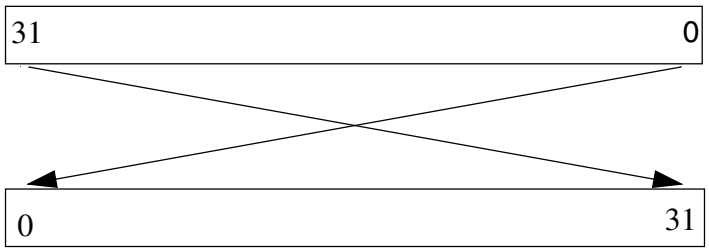


Figure 21-3. Transpose type 10

4. CTRL[TOT] or CTRL[TOTR] is 11.

   Bytes are transposed, but bits are not transposed.

   reg[31:0] becomes {reg[7:0], reg[15:8], reg[23:16], reg[31:24]}

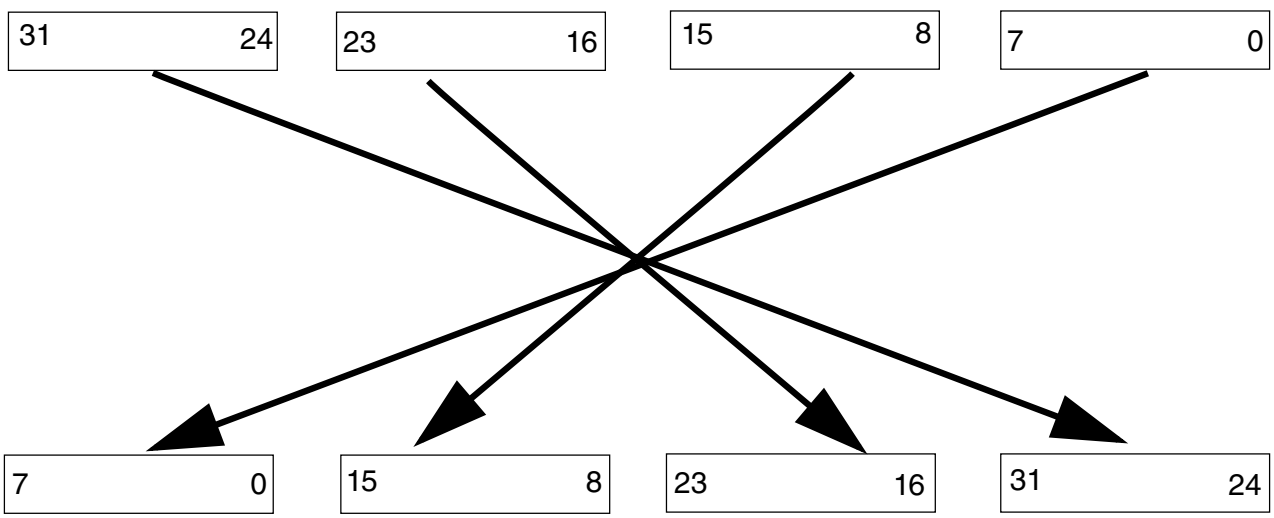


Figure 21-4. Transpose type 11

NOTE

- For 8-bit and 16-bit write accesses to the CRC data register, the data is transposed with zeros on the unused byte or bytes (taking 32 bits as a whole), but the CRC is calculated on the valid byte(s) only.
- When reading the CRC data register for a 16-bit CRC result and using transpose options 10 and 11, the resulting value after transposition resides in the CRC[HU:HL] fields. The user software must account for this situation when reading the 16-bit CRC result, so reading 32 bits is preferred.

21.3.4 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 22
12-bit Cyclic Analog-to-Digital Converter (ADC)

22.1 Chip-specific information for this module

22.1.1 Cyclic ADC Instantiation

This cyclic ADC module is also named as ADC12. It is a dual ADC. The signals of its first ADC are labeled A, as in ANA, ADCA, VREFLA, and VREFHA. The signals of its second ADC are labeled B, as in ANB, ADCB, VREFLB, and VREFHB.

NOTE

The expansion MUX select function is not available in the ADC block of this device, but the auxiliary control (AUXCTRL) function still exists. See the section "Expansion MUX and auxiliary control (AUXCTRL) function".

22.1.2 Cyclic ADC SYNC Signal Connections

The XBARA module's outputs XBAR_OUT12 and XBAR_OUT13 are connected to ADCA's SYNC input and ADCB's SYNC input, respectively. Through these XBARA connections, each ADC can be synchronized to another module, such as a PWM.

22.1.3 Cyclic ADC and PWM Connections

Within the chip, the cyclic ADC has internal connections for PWM control.

- PWM_SM0_EXTB is connected to cyclic ADC's conversion result 6 high/low limit
- PWM_SM1_EXTB is connected to cyclic ADC's conversion result 14 high/low limit[1]

- PWM_SM2_EXTB is connected to cyclic ADC's conversion result 7 high/low limit[1]
- PWM_SM3_EXTB is connected to cyclic ADC's conversion result 15 high/low limit[1]

22.1.4 On-chip analog signal input for ADC

- For ADCA, the on-chip analog signal input (as Ch17 in ADC) is OPAMPA output.
- For ADCB, the on-chip analog signal input (as Ch19 in ADC) is OPAMPB output.

22.1.5 Expose mode for ADC

When expose mode is enabled for ADCA:

- ANA4 is replaced with 12-bit DAC output;
- ANA5 is replaced with bandgap output (need to enable bandgap output buffer in PMC);
- ANA6 is replaced with internal test signal;
- ANA7 is replaced with internal test signal.

When expose mode is enabled for ADCB:

- ANB4 is replaced with 12-bit DAC output;
- ANB5 is replaced with bandgap output (need to enable bandgap output buffer in PMC);
- ANB6 is replaced with internal test signal;
- ANB7 is replaced with internal test signal.

22.2 Overview

The ADC module includes two 12-bit ADCs in which each ADC has a separate voltage reference and can operate simultaneously or independently. Digital control circuit schedules the ADC inputs sequence and the start time of ADC conversion. ADC conversion can be started by external trigger signals or software. Number of conversion per start can be programmed.

1. The signal state is decided by ADC conversion result in the result register RSLT[n].
   - If the ADC conversion result in RSLT[n] is greater than the value programmed into the High Limit register HILIM[n], this signal clears.
   - If the ADC conversion result in RSLT[n] is less than the value programmed into the Low Limit register LOLIM[n], this signal sets.
   - No change if the ADC conversion result in RSLT[n] is between the high and the low limits.

22.2.1 Block Diagram

The following figure illustrates the dual ADC configuration.

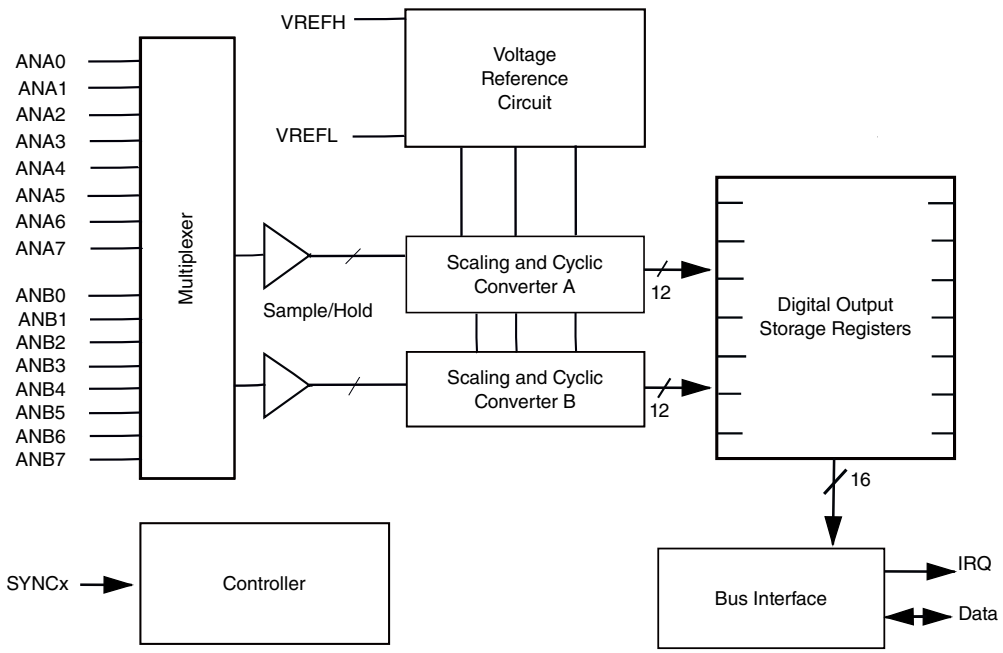


Figure 22-1. Dual ADC Block Diagram

22.2.2 Features

This dual 12-bit ADC consists of two separate converters, each with eight analog inputs and its own sample and hold circuit. A common digital control module configures and controls the functioning of the converters. ADC features include:

- 12-bit resolution
- Maximum ADC clock frequency of 12.5 MHz with 80 ns period
- Sampling rate up to 3.125 million samples per second[1]
- Single conversion time of 10 ADC clock cycles (10 × 80 ns = 800 ns)
- Additional conversion time of 8 ADC clock cycles (8 × 80 ns = 640 ns)
- Eight conversions in 34 ADC clock cycles (34 × 80 ns = 2.72 μs) using parallel mode

1. In loop mode, the time between each conversion is 8 ADC clock cycles (640 ns). Using simultaneous conversion, two samples can be obtained in 640 ns. Samples per second is calculated according to ns per two samples or 3,125,000 samples per second.

- Can be synchronized to other peripherals that are connected to an internal Inter-Peripheral Crossbar module, such as the PWM, through the SYNC0/1 input signal
- Sequentially scans and stores up to sixteen measurements
- Scans and stores up to eight measurements, each on two ADC converters operating simultaneously and in parallel
- Scans and stores up to eight measurements, each on two ADC converters operating asynchronously to each other in parallel
- A scan can pause and await new SYNC input prior to continuing
- Gains the input signal by x1, x2, or x4
- Optional interrupts at end of scan if an out-of-range limit is exceeded or there is a zero crossing
- Optional DMA function to transfer conversion data at the end of a scan or when a sample is ready to be read
- Optional sample correction by subtracting a pre-programmed offset value
- Signed or unsigned result
- Single-ended or differential inputs
- PWM outputs with hysteresis for three of the analog inputs
- Auxiliary control signals can be used to reconfigure other analog circuit, such as OPAMP or PGA, prior to ADC sampling.

22.3 Functional Description

ADC consists of two eight-channel input select functions, which are two independent sample and hold (S/H) circuits feeding two separate 12-bit converters. The two separate converters store their results in an accessible buffer, awaiting further processing. ANA[6:7] pins and ANB[6:7] pins are able to be substituted by each converter's on-chip analog signal inputs in the conversion scans.

The conversion process is initiated either by a SYNC signal or by writing a logic 1 to CTRLn[STARTn].

Starting a single conversion actually begins a sequence of conversions, or a scan. ADC operates in either sequential scan mode or parallel scan mode. In sequential scan mode, scan sequence is determined by defining twenty sample slots that are processed in order, SAMPLE0-19. In parallel scan mode, converter A processes SAMPLE0-7 plus

SAMPLE16-17 in order, and converter B processes SAMPLE8-15 plus SAMPLE18-19 in order. SAMPLE slots can be disabled using register SDIS and SDIS2 to terminate a scan early.

In the sequential scan mode (see the following figure), a scan takes up to sixteen single-ended or differential samples, one at a time. Optionally, up to four additional single-ended samples can be appended to the scan, specifically for sampling on-chip generated analog signals. These additional sample slots can be enabled or disabled using the SDIS2 register. In this scan mode, both ADCs operate alternatively depending on which input is being converted.

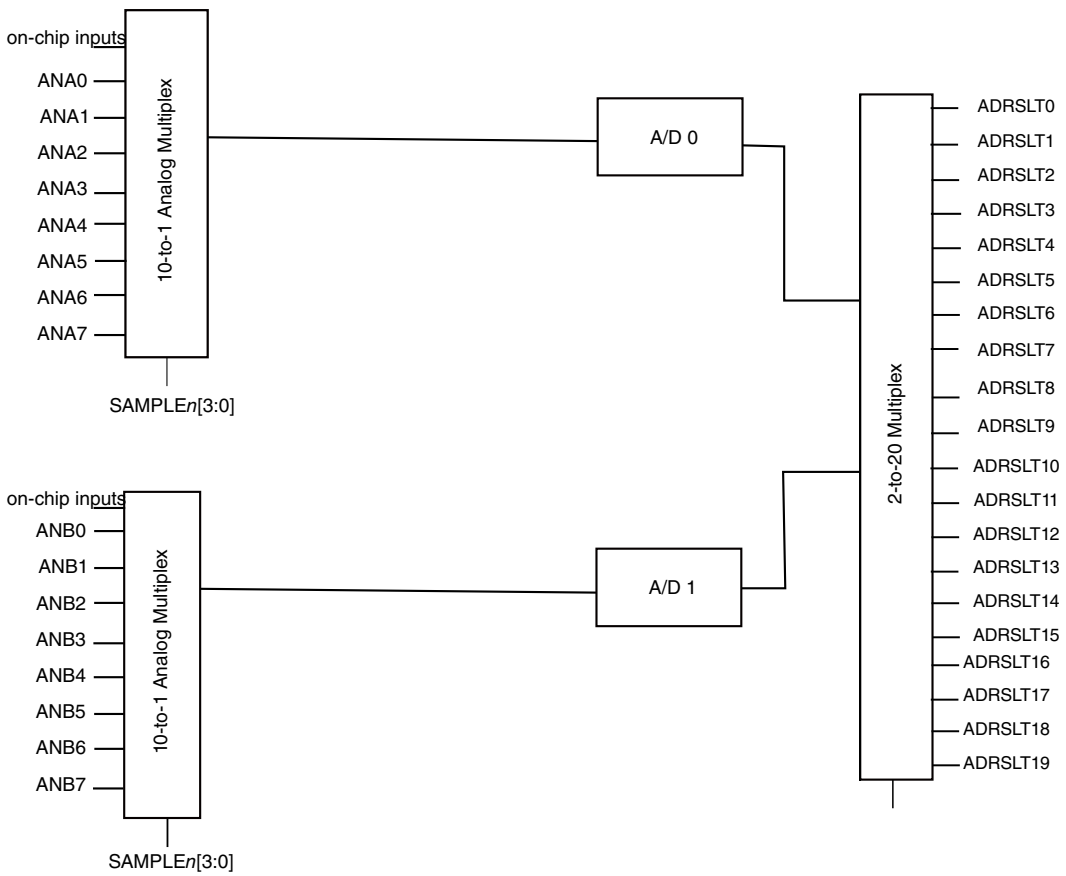


Figure 22-2. ADC Sequential Scan Mode

In parallel scan mode (see the following figure), eight of the sixteen samples are allocated to converter A and eight are allocated to converter B. Optionally, up to two additional single-ended samples can be appended to each ADC's scan. Two converters operate in parallel, and each can take at most eight samples with optional two on-chip generated input signals. Converter A can sample only analog inputs ANA[0:7], and converter B can sample only analog inputs ANB[0:7]. If additional on-chip samples are enabled , then each converter can sample its own temperature sensor and an on-chip analog signal.

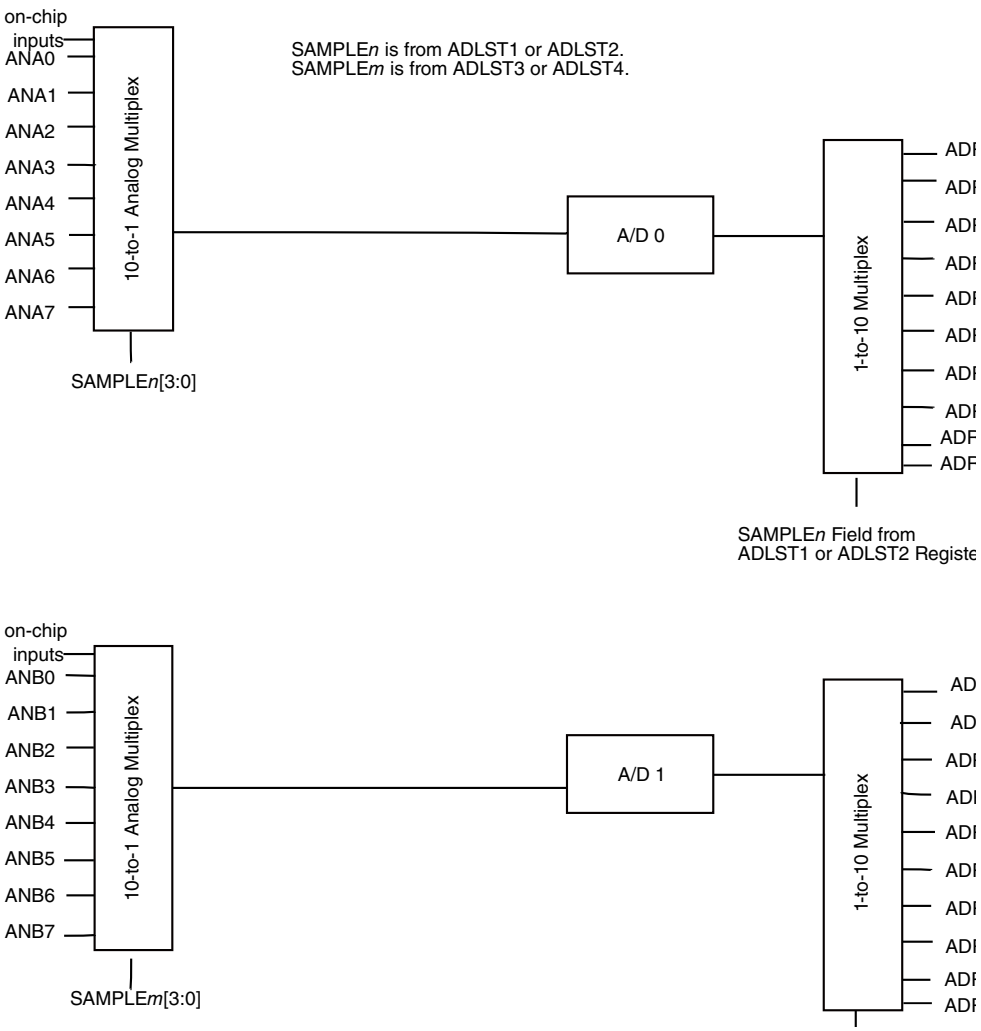


Figure 22-3. ADC Parallel Scan Mode

Each of the following pairs of analog inputs can be configured as a differential pair:

- ANA[0:1], ANA[2:3], ANA[4:5], ANA[6:7]
- ANB[0:1], ANB[2:3], ANB[4:5], ANB[6:7]

When they are configured as such, a reference to either member of the differential pair by a sample slot results in a differential measurement using that differential pair.

Parallel scan mode can be simultaneous or non-simultaneous.

- In simultaneous scan mode, the parallel scans in the two converters occur simultaneously and result in simultaneous pairs of conversions, one by converter A and one by converter B. The two converters share the same start, stop, sync, end-of-scan interrupt enable control, and interrupts. Scanning in both converters terminates when either converter encounters a disabled sample.
- In non-simultaneous scan mode, the parallel scans in the two converters occur independently. Each converter has its own start, stop, sync, end-of-scan interrupt enable controls, and interrupts. Scanning in either converter terminates only when that converter encounters a disabled sample.

The ADC can be configured to perform a single scan and halt, perform a scan whenever triggered, or perform the scan sequence repeatedly until manually stopped. The single scan (once mode) differs from the triggered mode only in that SYNC input signals must be re-armed after each use and subsequent SYNC inputs are ignored until the SYNC input is re-armed. This arming can occur any time after the SYNC pulse, including while the scan is still in process.

Optional interrupts can be generated at the end of a scan sequence. Interrupts are available simply to indicate that a scan has ended, that a selected conversion has completed, that a sample is out of range, or several different zero crossing conditions. Range is determined by the high and low limit registers.

22.3.1 Input Multiplex Function

The following figure shows the input multiplex function. The ChannelSelect and Single-Ended (or Differential) switches are indirectly controlled by settings within the following registers and bitfields:

- CLIST1, CLIST2, CLIST3, CLIST4, and SDIS registers
- CTRL1[CHNCFG_L] bitfield
- CTRL2[CHNCFG_H] bitfield

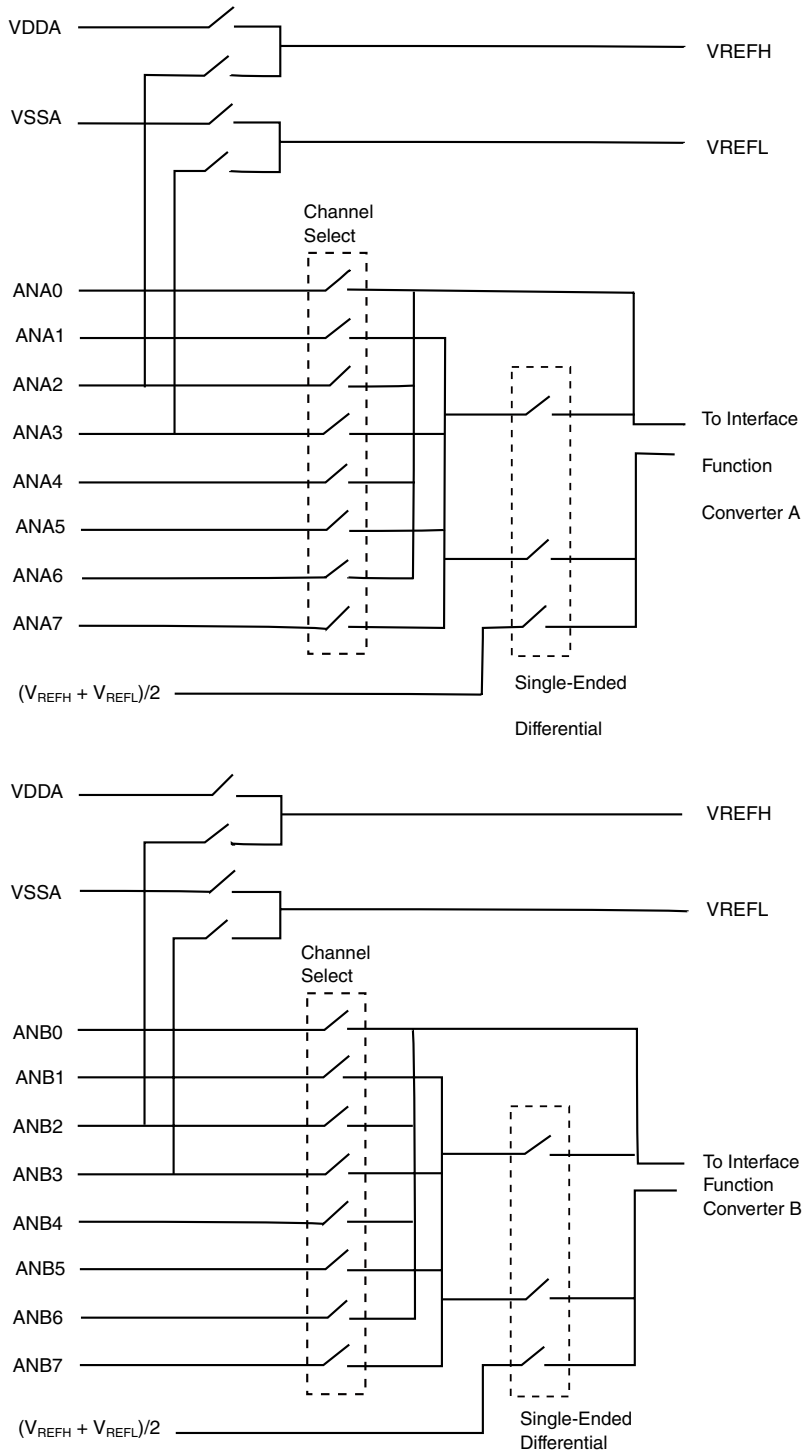


Figure 22-4. Input Select Multiplex

The multiplexing for conversions in different operating modes is as follows:

- Sequential, single-ended mode conversions — During each conversion cycle (sample), any one input can be directed to its corresponding sample and hold (S/H) circuit.

- Sequential, differential mode conversions — During any conversion cycle (sample), either member of a differential pair may be referenced, resulting in a differential measurement on that pair.
- Parallel, single-ended mode conversions — During any conversion cycle (sample), any of ANA[0:7] can be directed to the converter A output and any of ANB[0:7] can be directed to the converter B output. If additional on-chip samples are enabled, then each converter can sample its own temperature sensor and an on-chip analog signal.
- Parallel, differential mode conversions — During any conversion cycle (sample), either member of any possible differential pair—ANA0/1, ANA2/3, ANA4/5, ANA6/7, ANB0/1, ANB2/3, ANB4/5, and ANB6/7—can be referenced, resulting in a differential measurement of that pair at the converter A output (for ANA pairs) or converter B output (for ANB pairs).>If additional on-chip samples are enabled, then each converter can sample its own temperature sensor and an on-chip analog signal. But these inputs are single-end only.

22.3.2 ADC Sample Modes

The ADC module consists of two independent cyclic (or named algorithmic) ADCs. Each ADC uses two recursive sub-range section - Redundant Sign Digit (RSD) - architecture to resolve two conversion bits during each ADC clock cycle of the conversion process. The conversion process requires six ADC clocks for conversion plus two ADC clocks for sample and gain amplifier settling. Each ADC runs at a maximum clock speed of 12.5 MHz, so a complete 12-bit conversion can be accommodated in 640 (which is 8×80) ns, not including post-processing time.

22.3.2.1 Normal Mode Operation

ADC has two normal operating modes: single-ended mode and differential mode. For a given sample, the mode of operation is determined by CTRLn[CHNCFG_x]:

- Single-ended mode (CTRLn[CHNCFG_x]=0). The input multiplex of the ADC selects one of the 8 analog inputs and directs it to the plus terminal of the A/D core. The minus terminal of the A/D core is connected to the V_{REFL} reference. The ADC measures the voltage of the selected analog input and compares it against the (V_{REFH} - V_{REFL}) reference voltage range.
- Differential mode (CTRLn[CHNCFG_x]=1). The ADC measures the voltage difference between two analog inputs and compares that value against the (V_{REFH}– V_{REFL}) voltage range. The input is selected as an input pair: ANA0/1, ANA2/3,

ANA4/5, ANA6/7, ANB0/1, ANB2/3, ANB4/5, or ANB6/7. The plus terminal of the A/D core is connected to the even analog input, and the minus terminal is connected to the odd analog input.

A mix and match combination of single-ended and differential configurations may exist, example as below:

- ANA[0:1] differential; ANA[2:3] single-ended
- ANA[4:5] differential; ANA[6:7] single-ended
- ANB[0:1] differential; ANB[2:3] single-ended
- ANB[4:5] differential; ANB[6:7] single-ended

22.3.2.1.1 Single-Ended Samples

ADC performs a ratio metric conversion. For single-ended measurements, the digital result is proportional to the ratio of the analog input to the reference voltage:

$$SingleEndedValue = round(((V_{IN} - V_{REFL})/(V_{REFH} - V_{REFL})) \times 4096) \times 8$$

V_{IN} is the applied voltage at the input pin.

V_{REFH} and V_{REFL} are the voltages at the external reference pins on the device (typically $V_{REFH}=V_{DDA}$ and $V_{REFL}=V_{SSA}$).

Note: The 12-bit result is rounded to the nearest LSB.

Note: ADC is a 12-bit function with 4096 possible states. However, the 12 bits have been left shifted by 3 bits on the 16-bit data bus. As a result, the magnitude of this function, as read from the data bus, is now 32760.

22.3.2.1.2 Differential Samples

For differential measurements, the digital result is proportional to the ratio of the difference in the inputs to the difference in the reference voltages (V_{REFH} and V_{REFL}).

Differential measurements can be configured to a bipolar differential or unipolar differential using CTRL3[UPDEN_x]. Bipolar differential conversions can swing both positive and negative, while unipolar conversions are only positive but use the full code range of ADC. The following formula represent conversion result in different differential measurements.

$$DifferentialValue = round\left(\left(\frac{VIN1-VIN2}{VREFH-VREFL} \times 2048\right) + 2048\right) \times 8$$

$$UnipolarDifferentialValue = round\left(\frac{VIN1-VIN2}{VREFH-VREFL} \times 4096\right) \times 8$$

V_{IN} is the applied voltage at the input pin.

V_{REFH} and V_{REFL} are the voltage at the external reference pins on the device (typically $V_{REFH}=V_{DDA}$ and $V_{REFL}=V_{SSA}$).

Note: The 12-bit result is rounded to the nearest LSB.

Note: ADC is a 12-bit function with 4096 possible states. However, the 12 bits have been left shifted three bits on the 16-bit data bus, so the magnitude of this function, as read from the data bus, is now 32760.

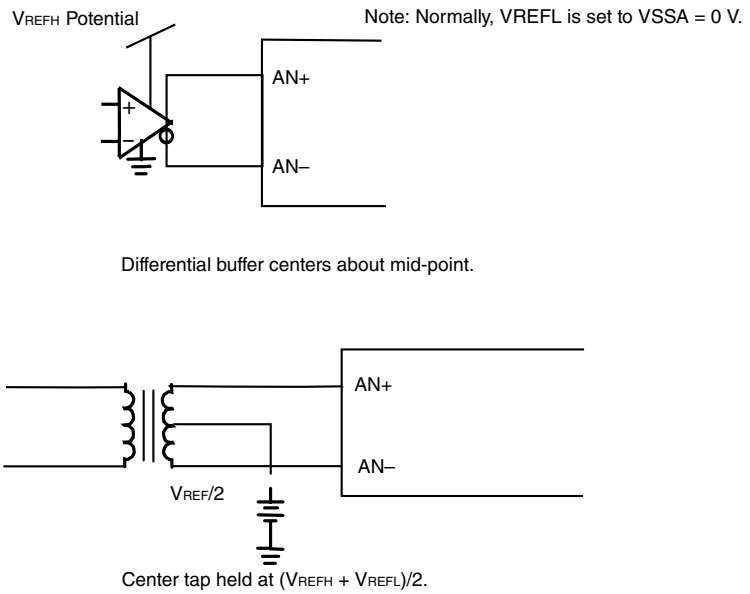


Figure 22-5. Typical Connections for Differential Measurements

22.3.3 ADC Data Processing

The result of an ADC conversion process is normally sent to an adder for offset correction, as the following figure shows. The adder subtracts the OFFSTn register value from each sample, and the resultant value is stored in the result register (RSLTn). The raw ADC value and the RSLTn values are checked for limit violations and zero-crossing, as shown. Appropriate interrupts are asserted, if enabled.

The result value sign is determined from the ADC unsigned result minus the respective offset register value. If the offset register is programmed with a value of zero, the result register value is unsigned and equals the cyclic converter unsigned result. The range of the result (RSLTn) is 0000h–7FF8h, assuming that the offset register (OFFSTn) is cleared to all zeros. This is equal to the raw value of the ADC core.

The processor can write the result registers used for the results of a scan when CTRLn[STOPx] for that scan is asserted. For example, if CTRL1[STOP0] is set to one and the processor writes to RSLT5, the data written to RSLT5 is multiplexed to the ADC digital logic inputs, processed, and stored into RSLT5 .

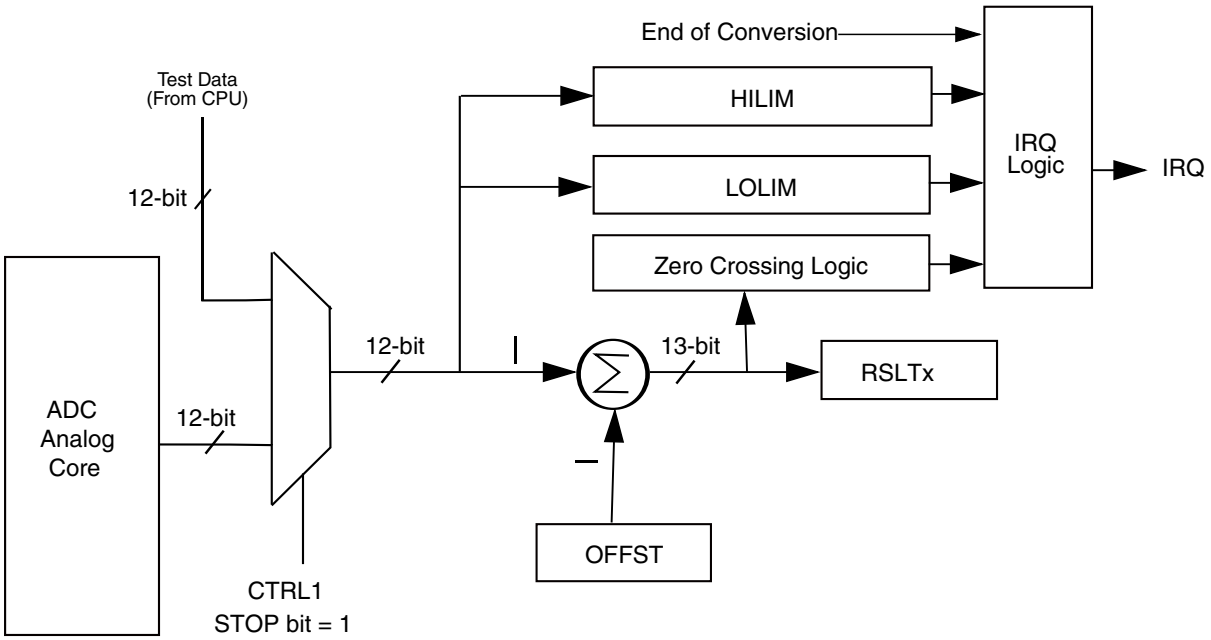


Figure 22-6. Result Register Data Manipulation

22.3.4 Sequential versus Parallel Sampling

All scan modes use the sixteen sample slots in the CLIST1–4 registers. The slots are used to define which input or differential pair to measure at each step in a scan sequence. The SDIS register defines which sample slots are enabled. An optional four additional sample slots in the CLIST5 register can be enabled with the SDIS2 register. Input pairs ANA0/1, ANA2/3, ANA4/5, ANA6/7, ANB0/1, ANB2/3, ANB4/5, and ANB6/7 can be set to be measured differentially using CTRLn[CHNCFG_x]. If a sample refers to an input that is not configured as a member of a differential pair, a single-ended measurement is made. If a sample refers to either member of a differential pair, a differential measurement is made.

Scan are either sequential or parallel. In sequential scans, up to sixteen sample slots are sampled one at a time in order SAMPLE0-15. Each sample refers to any of the sixteen analog inputs ANA0–ANB7, so the same input can be referenced by more than one sample slot. The four additional sample slots are sampled at the end of a scan in order SAMPLE16-19. Each of these sample slots refers to an ADC temperature sensor output or an on-chip generated analog signal. All samples have the full functionality of offset subtraction and high/low limit compare. Scanning is initiated when CTRL1[START0] is written with a 1, or when CTRL1[SYNC0] is set and the SYNC0 input goes high. A scan ends when the first disabled sample slot is encountered per the SDIS register. Completion of the scan triggers the STAT[EOSI0] interrupt if the CTRL1[EOSIEN0] interrupt enable

is set. If CTRL1[DMAEN0] is set and CTRL3[DMASRC]=0, a DMA transfer of the result data is initiated. The CTRL1[START0] bit and SYNC0 input are ignored while a scan is in process. Scanning stops and cannot be initiated when CTRL1[STOP0] is set.

Parallel scans differ in that converter A performs up to eight samples (SAMPLE[0:7]) in parallel with converter B (SAMPLE[8:15]). Constraints are as follows:

- SAMPLE[0:7] can reference only the ANA[0:7] inputs.
- SAMPLE[8:15] can reference only the ANB[0:7] inputs.

Likewise if the additional sample slots are enabled, converter A performs up to two extra samples (SAMPLE[16:17]) in parallel with converter B (SAMPLE[18:19]). SAMPLE[16:17] can reference only the ADCA temperature sensor and on-chip generated analog signal, while SAMPLE[18:19] can reference only the ADCB temperature sensor and on-chip generated analog signal.

Within these constraints, any sample can reference any pin, and more than one sample slot can reference the same sample and the same input. All samples have the full functionality of offset subtraction and high/low limit compare. By default (when CTRL2[SIMULT]=1), the scans in both converters are initiated when CTRL1[START0] is written with a 1, or when CTRL1[SYNC0] has a value of 1 and the SYNC0 input goes high. The scan in both converters terminates when either converter encounters a disabled sample slot. Completion of a scan triggers the STAT[EOSI0] interrupt if the CTRL1[EOSIEN0] interrupt enable is set. If CTRL1[DMAEN0] is set and CTRL3[DMASRC]=0, then a DMA transfer of the result data is initiated. Samples are always taken simultaneously in both the A and B converters. Setting CTRL1[STOP0] stops and prevents the initiation of scanning in both converters.

Setting CTRL2[SIMULT]=0 (non-simultaneous mode) causes parallel scanning to operate independently in the A and B converter. Each converter has its own set of START, STOP, SYNC, DMAEN, and EOSIEN control bits, SYNC input, EOSI interrupt, and CIP status indicators (suffix 0 for converter A and suffix 1 for converter B). Though still operating in parallel, the scans in the A and B converters start and stop independently according to their own controls and can be simultaneous, phase shifted, or asynchronous, depending on when scans are initiated on the respective converter. The A and B converters can be of different length (still up to a maximum of 8), and each converter's scan completes when a disabled sample is encountered in that converters sample list only. CTRL1[STOP0] stops the A converter only, while CTRL2[STOP1] stops the B converter only. Looping scan modes iterate independently. Each converter independently restarts its scan after completing its list or encountering a disabled sample slot.

22.3.5 Scan Sequencing

The sequential and parallel scan modes fall into three types based on how they repeat:

- Once scan. A once scan executes a sequential or parallel scan only once each time it is started. It differs from a triggered scan in that sync inputs must be re-armed after each use.
- Triggered scan. Identical to the corresponding once scan modes, except that resetting CTRLn[SYNCx] is not necessary.
- Looping scan. Automatically restarts a scan, either parallel or sequential, as soon as the previous scan completes. In parallel looping scan modes, the A converter scan restarts as soon as the A converter scan completes, and the B converter scan restarts as soon as the B converter scan completes. All subsequent start and sync pulses are ignored after the scan begins unless the scan is paused by SCTRL[SC] and SCTRL2[SC]. Scanning can only be terminated by setting CTRLn[STOPx].

All scan modes ignore sync pulses while a scan is in process, unless the scan is paused by SCTRL[SC] and SCTRL2[SC]. Once scan modes continue to ignore sync pulses even after the scan completes, until CTRLn[SYNCx] is set again. However, a reset can occur any time including during the scan. The SYNC0 input is re-armed by setting CTRL1[SYNC0], and the SYNC1 input is reset by setting CTRL2[SYNC1]. A reset can be performed any time after a scan starts.

22.3.6 Scan Halt

Scan can be paused by SCTRL[SC] and SCTRL2[SC], capable to halt the scan and await a new sync to continue the scan until sample disable (defined in SDISx registers) encounters. The SC bitfield is used to determine whether a sample in a scan occurs immediately following previous sample, or the sample waits for a sync inputs to occur. This feature is useful when the sample time depends on the input signal validation, for example synchronizing to a variable frequency PWM waveform.

22.3.7 Enabling additional sample slots for on-chip signals

ADC supports up to four extra sample slots (two for each converter) that are appended to the end of a scan on the ADC channels. These extra sample slots can only be assigned to on-chip generated analog signals, which include the temperature sensor and the voltage regulator bandgap reference. None of the ADC channels can be assigned to these sample slots. The assignment to the extra sample slots is configured by CLIST5 and enabled using SDIS2. As conversion results for these samples become available, the RDY2 status bits are asserted and the conversion data is stored in ADC Result Registers 2 with sign

extension (RSLT216-RSLT219). Dedicated registers for ADC data processing (offset, zero crossing, limit checking) and ADC control (scan control, gain control) are supported for the extra sample slots.

NOTE

Enabling the four extra sample slots by themselves (SDIS=FFFFh, and samples enabled in SDIS2) is supported only in once sequential mode. Sequential loop, and parallel (both sequential and loop) modes are not supported. It is suggested to poll the RDY2 register to check for conversion completion for this case.

22.3.8 Expansion MUX and auxiliary control (AUXCTRL) function

As shown in Figure 22-7, each ADC includes an 8-input expansion MUX whose output feeds to AN*x*4 of each ADC. Two auxiliary control signals (AUX*x*_SEL0 and AUX*x*_SEL1), as the extension of MUX control, provide the controls to other peripherals, such as pre-configuring a PGA. EXPAUX4x registers, which controls AUX*x*_SEL0 and AUX*x*_SEL1, are the extension to EXPMUX4x0 and EXPMUX4x1 registers, providing the ability to control other peripherals that require to synchronize with ADC conversion, from example of alternating OPAMP gains. EXPMUX4x0[MUXSEL0] and EXPAUX4x[AUXSEL0] take effect simultaneously, then subsequent EXPMUX4x0[MUXSEL1] and EXPAUX4x[AUXSEL1], and so on.

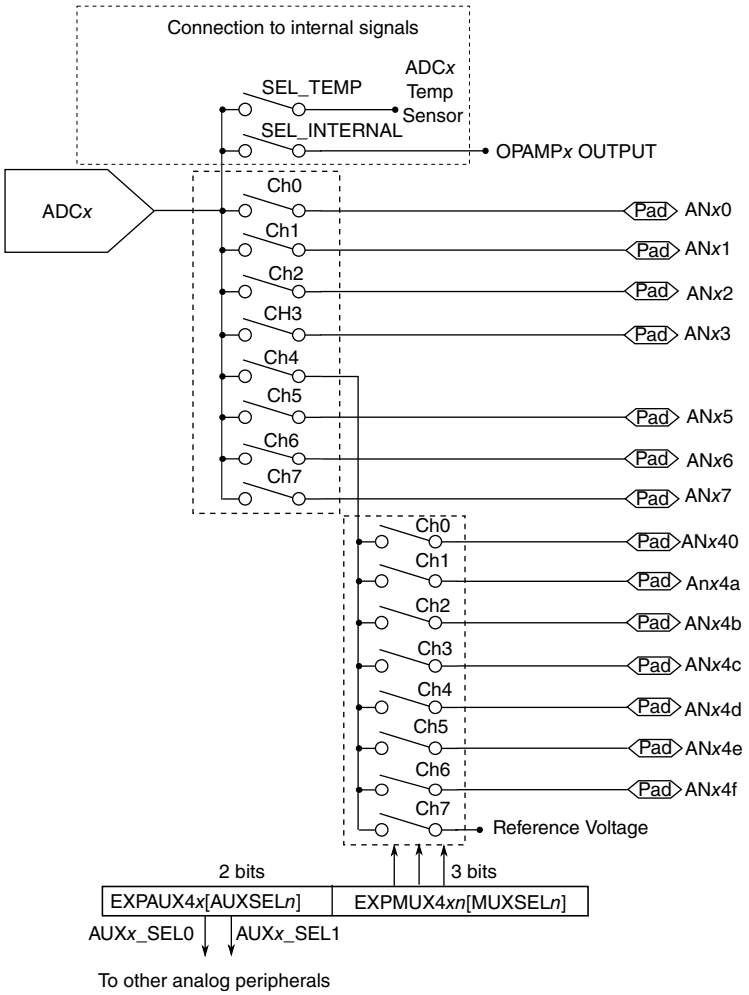


Figure 22-7. Expansion MUX

22.3.8.1 Utilizing auxiliary control registers (EXPAUX4x)

As shown in Figure 22-7, AUX*x*_SEL0 and AUX*x*_SEL1 can be used for other peripherals to toggle between different configurations. These two signals decode four combination states - 00b, 01b, 10b and 11b. You can utilize these two control signals to configure other peripherals prior to ADC sampling.

For example, it can be used to toggle the on-chip OPAMP between four configurations, such as gain amplifier, low-pass filter, difference amplifier and follower. The order sequence of different desired OPAMP configurations can be programmed into EXPAUX4x registers. During the ADC channel scan, AUX*x*_SEL0 and AUX*x*_SEL1 select OPAMP configuration to the desired one before ANx4 or ANx7 is sampled. This allows the output of OPAMP new configuration having enough time to be stabilized.

22.3.9 Power Management

The three supported power modes are discussed in order from the highest to the lowest power usage at the expense of increased conversion latency and/or startup delay. Changes to the SIM and OCCS modules that affect the power modes should be made while PWR[PD0] and PWR[PD1] are both asserted. See the Clocks section for details on the various clocks referenced here.

22.3.9.1 Low Power Modes

In the following table, the low-power modes are discussed in order from the highest to the lowest power usage.

Mode	Description
Normal power	At least one ADC converter is powered up (PWR[PD0 or PD1] is 0), the PWR[APD] bit as 0, and the SIM_PCEx[CYCADC] bit is 1. ADC uses the conversion clock as the ADC clock source in either active or idle. The conversion clock should be configured at or near 12.5 MHz to minimize conversion latency, when lower conversion frequencies are acceptable. No startup delay (PWR[PUDELAY]) is imposed.
Auto-Powerdown	At least one ADC converter is powered up (PWR[PD0 or PD1] is 0), PWR[APD] is 1, and the SIM_PCEx[CYCADC] bit is 1. ADC uses the conversion clock when active. For maximum power savings, it gates off the conversion clock and powers down the converters when idle. At the start of all scans, there is a startup delay of PWR[PUDELAY] ADC clocks to stabilize normal current mode from a completely powered off condition.
Powerdown	Both ADC converters and voltage references are powered down (PWR[PD0 and PD1] are both 1) and the SIM_PCEx[CYCADC] bit is 0. In this configuration, the clock trees to ADC and all of its analog components are shut down and power utilization is eliminated.

22.3.9.2 Startup in Different Power Modes

The ADC voltage reference and converters are powered down (PWR[PD*n*]=1) on reset. Individual converters and voltage references can be manually powered down when not in use (PWR[PD0 or PD1]=1). When the ADC reference is powered down, the output reference voltages are set to low (V_{SSA}) and the ADC data output is driven low.

A delay of PWR[PUDELAY] ADC clock cycles is imposed, when PWR[PD0 or PD1] is cleared, to power up a regulator and also to transition from an idle state in which neither converter has a scan in process to an active state in which at least one converter has a scan in process.

To start up in normal mode, perform the following steps, which provide a full power-up delay before scans begin.

1. Set PWR[PUDELAY] to the large power-up value.
2. Clear PWR[APD].

3. Clear PWR[PDn] to power up the required converters.
4. Poll the status bits until all required converters are powered up.
5. Start the scan operations.

In normal mode, PWR[PUDELAY] is not used at the start of scan, so no further delay is imposed.

To start up in auto-powerdown mode, perform the following steps:
1. Set PWR[PUDELAY] to the large power-up value.
2. Set PWR[APD].
3. Clear PWR[PDn] for the required converters.

Converters remain powered off until scanning goes active. Before a scan starts, there is a large power up delay time (PWR[PUDELAY]) to go from the powered down to the fully powered state.

To avoid ambiguity and ensure that the proper delays are applied when powering up or starting scans, both regulators should be powered off (PWR[PDn]=1) when the clock or power controls are configured.

Attempts to start a scan during the power up delay time (PWR[PUDELAY]) are ignored until the appropriate PWR[PSTSn] bits are cleared.

NOTE

There is a risk of ADC power up failure when changing the ADC configuration during its power up delay time (after PWR[PD*n*] is cleared and before PWR [PSTS*n*] is deasserted). Follow either of the guidance below to avoid such risk.

- Option 1: clear PWR[PD0] or [PD1] at the end of ADC initialization process.
- Option 2: clear PWR[PD0] or [PD1] at any other phase during the ADC initialization process, wait till PWR[PSTS0] or [PSTS1] is deasserted, and then do the rest configurations.

Any attempt to use a converter, when it is powered down or with the voltage references disabled, will cause invalid results. For example, it is possible to read ADC result registers after converter power down for results calculated before power-down. A new scan sequence must be started with a SYNC pulse or a write to the START bit before new valid results are available.

In auto-powerdown mode, when ADC goes from idle to active, a converter is powered up only if it is required for the scan as determined by the CLIST1-CLIST4 and SDIS registers.

22.3.9.3 Stop Mode of Operation

Any conversion sequence can be stopped by setting the relevant CTRLn[STOPx] bit. Any further SYNC pulses, or writes to the CTRLn[STARTx] bit, are ignored until CTRLn[STOPx] is cleared. In stop mode, the results registers can be modified by writes from the processor. Any write to the result register in the ADC stop mode is treated as if the analog core supplied the data, so limit checking, zero crossing and associated interrupts can occur if enabled.

22.3.10 Clocking

The following table shows external clock input(s) to drive clock domain(s) within ADC.

Table 22-1. Clock Summary

Clock input	Source	Characteristics
IP Clock (IP_CLK)	SIM	Maximum rate is 50 MHz. When the PLL is on and selected, it is PLL output divided by 4. When PLL is not selected, it is MSTR_OSC/2. When the device is in low-power mode, OCCS_OSCTL1[ROSB]=1, the rate is 1 MHz.

The IP clock rate is determined by the OCCS module configuration, which is highly programmable. One of two sources (an external clock pin, or the ROSC) can be selected using the PRECS control to generate the IP_CLK. The maximum rate of the IP clock to ADC is therefore either:

- 50 MHz based on PLL output
- A maximum external clock rate of 100 MHz divided by 2

The IP_CLK is enabled only when SIM_PCEx[CYCADC] is set. This clock enable bit must be set before ADC can be used.

The conversion clock is the primary source for the ADC clock and is always selected as the ADC clock when conversions are in process. The clock source controls in the OCCS module (PRECS, ROPD, ROSB), and CTRL2[DIV0] and PWR2[DIV1] should be configured so that conversion clock frequency falls between 100 kHz and 12.5 MHz. Operating the ADC at out-of-spec conversion clock frequencies or reconfiguring the parameters that affect clock rates or power modes while the regulators are powered up (PWR[PDn]=0) negatively affects conversion accuracy.

The conversion clock that the ADC uses for sampling is calculated using the IP bus clock and CTRL2[DIV0]. The ADC clock is active, 100 percent of the time, in looping modes or in normal power mode. It is also active during all ADC powerup sequences for a period of time determined by PWR[PUDELAY]. If a conversion is initiated in power savings mode, then the ADC clock continues until the conversion sequence completes.

The following diagram shows the structure of the clocking system.

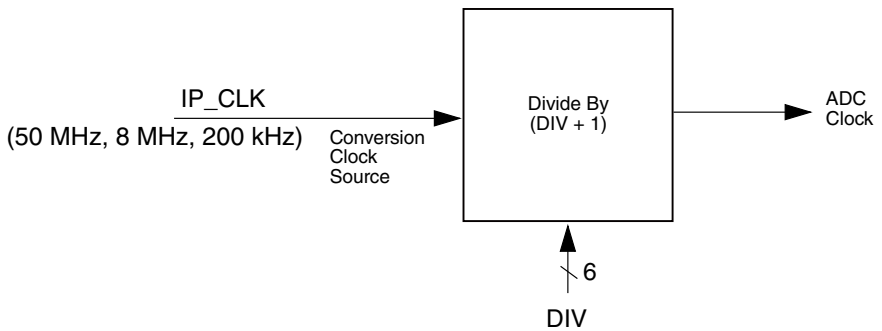


Figure 22-8. ADC Clock Generation

The ADC clock (ADC_CLK) is an output of the gasket used to operate the two converters during scan operations. It is derived by the conversion clock (divided version of the conversion clock source). This clock can be selected in the SIM module for external output, for debug and failure analysis.

22.3.11 Reset

At reset, all the registers return to the reset state. The source of the single reset signal is the SIM module.

22.3.12 Interrupts

The following table summarizes the ADC interrupts.

Table 22-2. Interrupt Summary

Interrupt	Source	Description
ADC_ERR_INT_B	STAT[ZCI], STAT[LLMTI], STAT[HLMTI]	Zero crossing, low limit, and high limit interrupt
ADC_CC0_INT_B	STAT[EOSI0] RDY[RDY] RDY2[RDY]	Conversion Complete and Scan Interrupt for any scan type except converter B scan in non-simultaneous parallel scan mode (see STAT[EOSI0])

Table continues on the next page...

Table 22-2. Interrupt Summary (continued)

Interrupt	Source	Description
	EXPSTAT[MUXAIRQ]	
ADC_CC1_INT_B	STAT[EOSI1] RDY[RDY] RDY2[RDY] EXPSTAT[MUXBIRQ]	Conversion Complete and Scan Interrupt for converter B scan in non-simultaneous parallel scan mode (see STAT[EOSI1])

ADC interrupts fall into three categories:

- Threshold interrupts, which are caused by three different events. All of these interrupts are optional and enabled through control register CTRL1:
  - Zero crossing — occurs if the current result value has a sign change from the previous result as configured by the ZXCTRL* register.
  - Low limit exceeded error — occurs when the current result value is less than the low limit register value. The raw result value is compared to LOLIM*[LLMT] before the offset register value is subtracted.
  - High limit exceeded error — is asserted if the current result value is greater than the high limit register value. The raw result value is compared to HILIM*[HLMT] before the offset register value is subtracted.
- Conversion (or expansion AUX scan) complete interrupts, which are generated upon completion of any scan and convert sequence when CTRL1[EOSIE0] or CTRL2[EOSIE1]=1 (or EXPCFG[MUXAIE] or EXPCFG[MUXBIE] = 1). Additional bits may need to be set in the INTC module to enable the CPU to receive the interrupt signal.
- Scan interrupts, which are generated when a sample is converted. This allows processing of intermediate conversion data during a scan. The interrupt occurs when any sample has its SCHLTEN*[SCHLTEN] bit enabled, and the RDY*[RDY] bit for that sample is asserted. Use these registers to determine which sample triggered the interrupt, or when expansion AUX channel scan is completed.

22.4 Signal Descriptions

22.4.1 Signal Overview

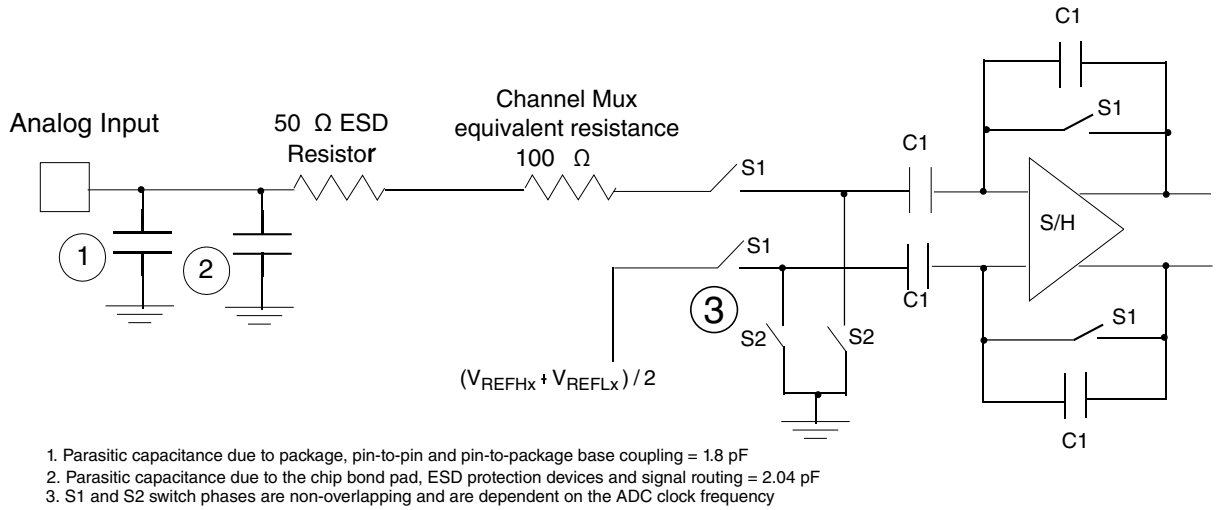
Table 22-3. Signal Properties

Name	I/O Type	Function	Reset State	Notes
VREFH	I	Voltage reference pin	n/a	Selectable between VDDA and ANA2
VREFL	I	Voltage reference pin	n/a	Selectable between VSSA and ANA3
VREFH	I	Voltage reference pin	n/a	Selectable between VDDA and ANB2
VREFL	I	Voltage reference pin	n/a	Selectable between VSSA and ANB3
VDDA	Supply	ADC power	n/a	—
VSSA	Supply	ADC ground	n/a	—
ANA0–ANA7	I	Analog input pins	n/a	—
ANB0–ANB7	I	Analog input pins	n/a	—

22.4.2 External Signal Descriptions

22.4.2.1 Analog Input Pins (ANA[0:7] and ANB[0:7])

The ADC module has sixteen analog input pins and four on-chip analog inputs, that are subdivided into two sets (ANA[0:7] plus two on-chip analog signal inputs, and ANB[0:7] plus two on-chip analog signal inputs), each with its own sample and hold (S/H) circuit and converter as well as temperature sensor. ANA[0:7] and ANB[0:7] connect to off-chip analog inputs, while on-chip analog inputs connect to on-chip generated signals, such as on-chip temperature sensors, OPAMPs. This configuration allows simultaneous sampling of two selected channels: one from each subgroup. Sequential scans have access to up to twenty analog inputs. During parallel scans, each ADC converter has access to its ten analog inputs. An equivalent circuit for an analog input is shown below.



1. Parasitic capacitance due to package, pin-to-pin and pin-to-package base coupling = 1.8 pF
2. Parasitic capacitance due to the chip bond pad, ESD protection devices and signal routing = 2.04 pF
3. S1 and S2 switch phases are non-overlapping and are dependent on the ADC clock frequency

Figure 22-9. Equivalent Analog Input Circuit

22.4.2.2 Voltage Reference Pins (V_{REFH} and V_{REFL})

The voltage difference between V_{REFH} and V_{REFL} provides the reference voltage against which all analog inputs are measured. V_{REFH} is nominally set to V_{DDA}, and V_{REFL} is nominally set to 0 V. Any external reference voltage should come from a low noise filtered source. The external reference source should provide up to 1 mA of reference current. The following figure illustrates the internal workings of the ADC voltage reference circuit. V_{REFH} must be noise filtered. A minimum configuration is shown in the figure.

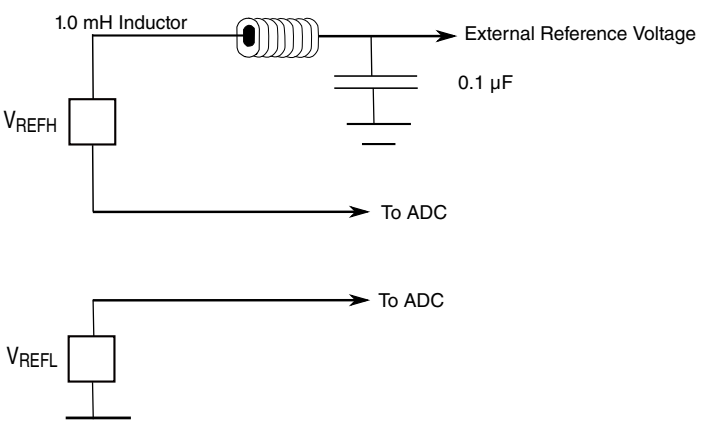


Figure 22-10. ADC Voltage Reference Circuit

When V_{DDA} is used as V_{REFH}, measurements are made with respect to the amplitude of V_{DDA}. Special precautions must be taken to assure that the voltage applied to V_{REFH} is as noise free as possible. Any noise residing on the V_{REFH} voltage is directly transferred to the digital result.

Dedicated power supply pins, V_{DDA} and V_{SSA}, are provided to reduce noise coupling and to improve accuracy. The power to these pins should come from a low noise filtered source. Uncoupling capacitors should be connected between V_{DDA} and V_{SSA}.

22.5 Initialization

See Startup in Different Power Modes for details on ADC startup process and the note.

22.6 Application information

22.6.1 Timing Specifications

The following figure shows a timing diagram for the ADC module. ADC is assumed to be in Once or Triggered mode, so the ADC clock is shown in the OFF state prior to the SYNC pulse or START bit write. The ADC clock restarts (switching high) within 1 to 2 IP bus clock cycles of that event. ADC_CLK is derived from the ROSC or PLL output. The frequency relationship is programmable. Conversions are pipelined. The second start command is ignored because the ADC is busy with the previous start request. The third start command is recognized and is synchronized to the positive edge of the ADC clock when the conversion process is restarted. ADC has two possible interrupts that are latched in the ADSTAT register:

- Conversion complete interrupt (End of Scan interrupt, EOSIx)
- Zero crossing or limit error interrupt (ZCI, LLMTI, and HLMTI)

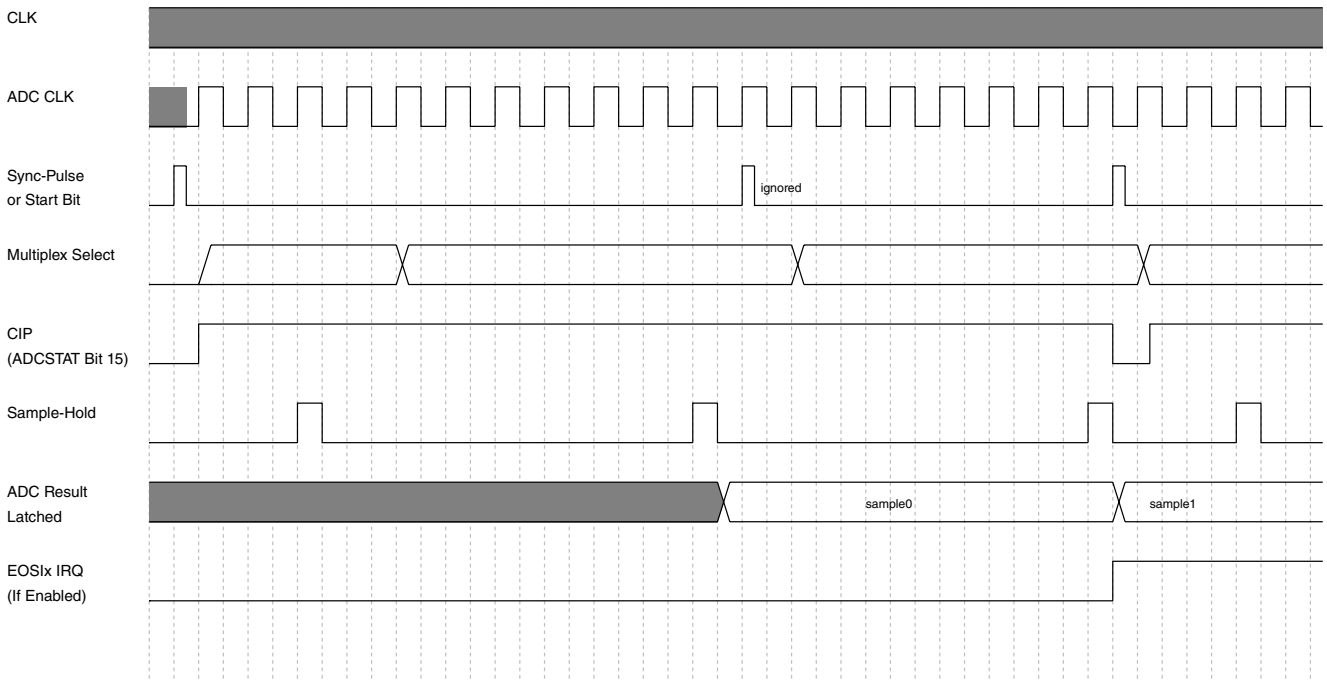


Figure 22-11. ADC Timing

As the figure shows, a conversion is initiated by (1) a SYNC pulse originating from the timer module or by (2) a write to CTRLn[STARTx]. In APD, a delay of PWR[PUDELAY] ADC clock cycles is imposed. The conversion is initiated in the next clock cycle. The ADC clock period is determined by the CTRL2[DIV0] or PWR2[DIV1] value and the OCCS clock configuration.

The first conversion takes 10 ADC clock cycles to be valid. Then, each additional sample takes only eight ADC clock cycles. The start conversion command is latched and the real conversion process is synchronized to the positive edge of the ADC clock.

Because the conversion is a pipeline process, after the last sample is in the S/H, ADC cannot be restarted until the pipeline is emptied. However, the conversion cycle can be aborted by issuing a STOP command.

The figure shown here illustrates the case in which PWR[APD] is not in use. When PWR[APD] is set, the SYNC pulse or CTRLn[STARTx] powers up the ADC, and it waits for a number of ADC clock cycles (determined by PWR[PUDELAY]) for the ADC circuitry to stabilize, and only then the conversion sequence begins.

22.7 Memory Map and Registers

22.7.1 ADC register descriptions

22.7.1.1 ADC memory map

ADC base address: E500h

Offset	Register	Width (In bits)	Access	Reset value
0h	ADC Control Register 1 (CTRL1)	16	RW	5005h
1h	ADC Control Register 2 (CTRL2)	16	RW	5044h
2h	ADC Zero Crossing Control 1 Register (ZXCTRL1)	16	RW	0000h
3h	ADC Zero Crossing Control 2 Register (ZXCTRL2)	16	RW	0000h
4h	ADC Channel List Register 1 (CLIST1)	16	RW	3210h
5h	ADC Channel List Register 2 (CLIST2)	16	RW	7654h
6h	ADC Channel List Register 3 (CLIST3)	16	RW	BA98h
7h	ADC Channel List Register 4 (CLIST4)	16	RW	FEDCh
8h	ADC Sample Disable Register (SDIS)	16	RW	F0F0h
9h	ADC Status Register (STAT)	16	W1C	0000h
Ah	ADC Ready Register (RDY)	16	RO	0000h
Bh	ADC Low Limit Status Register (LOLIMSTAT)	16	W1C	0000h
Ch	ADC High Limit Status Register (HILIMSTAT)	16	W1C	0000h
Dh	ADC Zero Crossing Status Register (ZXSTAT)	16	W1C	0000h
Eh	ADC Result Registers with sign extension (RSLT0)	16	RW	0000h
Fh	ADC Result Registers with sign extension (RSLT1)	16	RW	0000h
10h	ADC Result Registers with sign extension (RSLT2)	16	RW	0000h
11h	ADC Result Registers with sign extension (RSLT3)	16	RW	0000h
12h	ADC Result Registers with sign extension (RSLT4)	16	RW	0000h
13h	ADC Result Registers with sign extension (RSLT5)	16	RW	0000h
14h	ADC Result Registers with sign extension (RSLT6)	16	RW	0000h
15h	ADC Result Registers with sign extension (RSLT7)	16	RW	0000h
16h	ADC Result Registers with sign extension (RSLT8)	16	RW	0000h
17h	ADC Result Registers with sign extension (RSLT9)	16	RW	0000h
18h	ADC Result Registers with sign extension (RSLT10)	16	RW	0000h
19h	ADC Result Registers with sign extension (RSLT11)	16	RW	0000h
1Ah	ADC Result Registers with sign extension (RSLT12)	16	RW	0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
1Bh	ADC Result Registers with sign extension (RSLT13)	16	RW	0000h
1Ch	ADC Result Registers with sign extension (RSLT14)	16	RW	0000h
1Dh	ADC Result Registers with sign extension (RSLT15)	16	RW	0000h
1Eh	ADC Low Limit Registers (LOLIM0)	16	RW	0000h
1Fh	ADC Low Limit Registers (LOLIM1)	16	RW	0000h
20h	ADC Low Limit Registers (LOLIM2)	16	RW	0000h
21h	ADC Low Limit Registers (LOLIM3)	16	RW	0000h
22h	ADC Low Limit Registers (LOLIM4)	16	RW	0000h
23h	ADC Low Limit Registers (LOLIM5)	16	RW	0000h
24h	ADC Low Limit Registers (LOLIM6)	16	RW	0000h
25h	ADC Low Limit Registers (LOLIM7)	16	RW	0000h
26h	ADC Low Limit Registers (LOLIM8)	16	RW	0000h
27h	ADC Low Limit Registers (LOLIM9)	16	RW	0000h
28h	ADC Low Limit Registers (LOLIM10)	16	RW	0000h
29h	ADC Low Limit Registers (LOLIM11)	16	RW	0000h
2Ah	ADC Low Limit Registers (LOLIM12)	16	RW	0000h
2Bh	ADC Low Limit Registers (LOLIM13)	16	RW	0000h
2Ch	ADC Low Limit Registers (LOLIM14)	16	RW	0000h
2Dh	ADC Low Limit Registers (LOLIM15)	16	RW	0000h
2Eh	ADC High Limit Registers (HILIM0)	16	RW	7FF8h
2Fh	ADC High Limit Registers (HILIM1)	16	RW	7FF8h
30h	ADC High Limit Registers (HILIM2)	16	RW	7FF8h
31h	ADC High Limit Registers (HILIM3)	16	RW	7FF8h
32h	ADC High Limit Registers (HILIM4)	16	RW	7FF8h
33h	ADC High Limit Registers (HILIM5)	16	RW	7FF8h
34h	ADC High Limit Registers (HILIM6)	16	RW	7FF8h
35h	ADC High Limit Registers (HILIM7)	16	RW	7FF8h
36h	ADC High Limit Registers (HILIM8)	16	RW	7FF8h
37h	ADC High Limit Registers (HILIM9)	16	RW	7FF8h
38h	ADC High Limit Registers (HILIM10)	16	RW	7FF8h
39h	ADC High Limit Registers (HILIM11)	16	RW	7FF8h
3Ah	ADC High Limit Registers (HILIM12)	16	RW	7FF8h
3Bh	ADC High Limit Registers (HILIM13)	16	RW	7FF8h
3Ch	ADC High Limit Registers (HILIM14)	16	RW	7FF8h
3Dh	ADC High Limit Registers (HILIM15)	16	RW	7FF8h
3Eh	ADC Offset Registers (OFFST0)	16	RW	0000h
3Fh	ADC Offset Registers (OFFST1)	16	RW	0000h
40h	ADC Offset Registers (OFFST2)	16	RW	0000h
41h	ADC Offset Registers (OFFST3)	16	RW	0000h
42h	ADC Offset Registers (OFFST4)	16	RW	0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
43h	ADC Offset Registers (OFFST5)	16	RW	0000h
44h	ADC Offset Registers (OFFST6)	16	RW	0000h
45h	ADC Offset Registers (OFFST7)	16	RW	0000h
46h	ADC Offset Registers (OFFST8)	16	RW	0000h
47h	ADC Offset Registers (OFFST9)	16	RW	0000h
48h	ADC Offset Registers (OFFST10)	16	RW	0000h
49h	ADC Offset Registers (OFFST11)	16	RW	0000h
4Ah	ADC Offset Registers (OFFST12)	16	RW	0000h
4Bh	ADC Offset Registers (OFFST13)	16	RW	0000h
4Ch	ADC Offset Registers (OFFST14)	16	RW	0000h
4Dh	ADC Offset Registers (OFFST15)	16	RW	0000h
4Eh	ADC Power Control Register (PWR)	16	RW	1DA7h
4Fh	ADC Calibration Register (CAL)	16	RW	0000h
50h	Gain Control 1 Register (GC1)	16	RW	0000h
51h	Gain Control 2 Register (GC2)	16	RW	0000h
52h	ADC Scan Control Register (SCTRL)	16	RW	0000h
53h	ADC Power Control Register 2 (PWR2)	16	RW	0400h
54h	ADC Control Register 3 (CTRL3)	16	RW	0000h
55h	ADC Scan Interrupt Enable Register (SCHLTEN)	16	RW	0000h
58h	ADC Zero Crossing Control 3 Register (ZXCTRL3)	16	RW	0000h
59h	ADC Channel List Register 5 (CLIST5)	16	RW	00E4h
5Ah	ADC Sample Disable Register 2 (SDIS2)	16	RW	FFFFh
5Bh	ADC Ready Register 2 (RDY2)	16	RO	0000h
5Ch	ADC Low Limit Status Register 2 (LOLIMSTAT2)	16	W1C	0000h
5Dh	ADC High Limit Status Register 2 (HILIMSTAT2)	16	W1C	0000h
5Eh	ADC Zero Crossing Status Register 2 (ZXSTAT2)	16	W1C	0000h
5Fh	ADC Result Registers 2 with sign extension (RSLT216)	16	RW	0000h
60h	ADC Result Registers 2 with sign extension (RSLT217)	16	RW	0000h
61h	ADC Result Registers 2 with sign extension (RSLT218)	16	RW	0000h
62h	ADC Result Registers 2 with sign extension (RSLT219)	16	RW	0000h
63h	ADC Low Limit Registers 2 (LOLIM216)	16	RW	0000h
64h	ADC Low Limit Registers 2 (LOLIM217)	16	RW	0000h
65h	ADC Low Limit Registers 2 (LOLIM218)	16	RW	0000h
66h	ADC Low Limit Registers 2 (LOLIM219)	16	RW	0000h
67h	ADC High Limit Registers 2 (HILIM216)	16	RW	7FF8h
68h	ADC High Limit Registers 2 (HILIM217)	16	RW	7FF8h
69h	ADC High Limit Registers 2 (HILIM218)	16	RW	7FF8h
6Ah	ADC High Limit Registers 2 (HILIM219)	16	RW	7FF8h
6Bh	ADC Offset Registers 2 (OFFST216)	16	RW	0000h
6Ch	ADC Offset Registers 2 (OFFST217)	16	RW	0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
6Dh	ADC Offset Registers 2 (OFFST218)	16	RW	0000h
6Eh	ADC Offset Registers 2 (OFFST219)	16	RW	0000h
6Fh	Gain Control 3 Register (GC3)	16	RW	0000h
70h	ADC Scan Control Register 2 (SCTRL2)	16	RW	0000h
71h	ADC Scan Interrupt Enable Register 2 (SCHLTEN2)	16	RW	0000h
78h	Expansion AUX Status Register (EXPSTAT)	16	W1C	0000h
79h	Expansion AUX Config Register (EXPCFG)	16	RW	00F0h
7Ah	ANA4 Expansion Auxiliary Control Register (EXPAUX4A)	16	RW	0000h
7Bh	ANB4 Expansion Auxiliary Control Register (EXPAUX4B)	16	RW	0000h
7Ch	ANA4 Expansion MUX Control Register 0 (EXPMUX4A0)	16	RW	0000h
7Dh	ANA4 Expansion MUX Control Register 1 (EXPMUX4A1)	16	RW	0000h
7Eh	ANB4 Expansion MUX Control Register 0 (EXPMUX4B0)	16	RW	0000h
7Fh	ANB4 Expansion MUX Control Register 1 (EXPMUX4B1)	16	RW	0000h

22.7.1.2 ADC Control Register 1 (CTRL1)

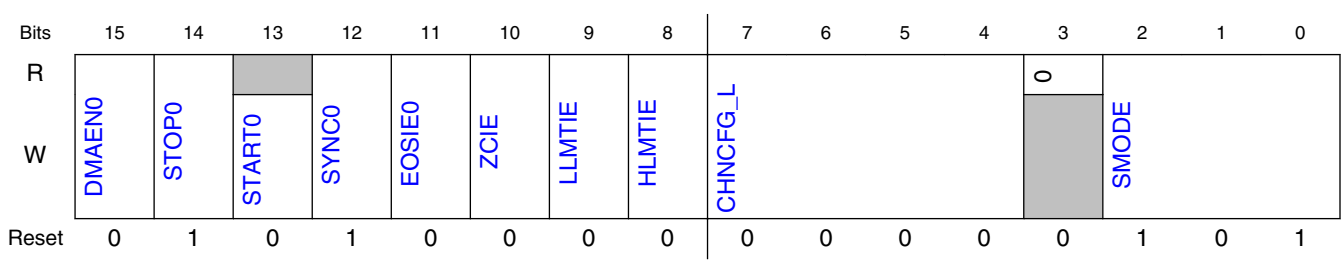
22.7.1.2.1 Offset

Register	Offset
CTRL1	0h

22.7.1.2.2 Function

Bits [14:11] in CTRL1 control all types of scans except parallel scans in the B converter when CTRL2[SIMULT]=0. Non-simultaneous parallel scan modes allow independent parallel scanning in the A and B converter. Bits [14:11] in CTRL2 are used to control B converter scans in non-simultaneous parallel scan modes.

22.7.1.2.3 Diagram



22.7.1.2.4 Fields

Field	Function
15 DMAEN0	DMA enable When this bit is asserted, the DMA source selected by CTRL3[DMASRC] causes the conversion results to be transferred by the DMA controller. Setting this bit blocks the generation of the EOSI0 end of scan interrupt. 0b - DMA is not enabled. 1b - DMA is enabled.
14 STOP0	Stop When this bit is asserted, the current scan is stopped and no further scans can start. Any further SYNC0 input pulses (see CTRL1[SYNC0] bit) or writes to the CTRL1[START0] bit are ignored until this bit has been cleared. After the ADC is in stop mode, the results registers can be modified by the processor. Any changes to the result registers in stop mode are treated as if the analog core supplied the data. Therefore, limit checking, zero crossing, and associated interrupts can occur when authorized. **This is not the same as the chip STOP mode.** 0b - Normal operation 1b - Stop mode
13 START0	START0 Conversion A scan is started by writing 1 to this bit. This is a write only bit. Writing 1 to it again while the scan remains in process, is ignored. The ADC must be in a stable power configuration prior to writing the start bit. Refer to the functional description of power modes for further details. 0b - No action 1b - Start command is issued
12 SYNC0	SYNC0 Enable A conversion may be initiated by asserting a positive edge on the SYNC0 input. Any subsequent SYNC0 input pulses while the scan remains in process are ignored unless the scan is awaiting further SYNC inputs due to the SCTRL[SCn] bits. CTRL1[SYNC0] is cleared in ONCE mode, CTRL1[SMODE=000 or 001], when the first SYNC input is detected. This prevents unintentionally starting a new scan after the first scan has completed. The ADC must be in a stable power mode prior to SYNC0 input assertion. Refer to the functional description of power modes for further details. In "once" scan modes, only a first SYNC0 input pulse is honored. CTRL1[SYNC0] is cleared in this mode when the first SYNC input is detected. This prevents unintentionally starting a new scan after the first scan has completed. The CTRL1[SYNC0] bit can be set again at any time including while the scan remains in process 0b - Scan is initiated by a write to CTRL1[START0] only 1b - Use a SYNC0 input pulse or CTRL1[START0] to initiate a scan
11 EOSIE0	End Of Scan Interrupt Enable This bit enables an EOSI0 interrupt to be generated upon completion of the scan. For looping scan modes, the interrupt will trigger after the completion of each iteration of the loop. 0b - Interrupt disabled 1b - Interrupt enabled
10 ZCIE	Zero Crossing Interrupt Enable This bit enables the zero crossing interrupt if the current result value has a sign change from the previous result as configured by the ZXCTRL1 and ZXCTRL2 registers.

Table continues on the next page...

Field	Function
	0b - Interrupt disabled 1b - Interrupt enabled
9 LLMTIE	Low Limit Interrupt Enable This bit enables the Low Limit exceeded interrupt when the current result value is less than the low limit register value. The raw result value is compared to LOLIM[LLMT] before the offset register value is subtracted. 0b - Interrupt disabled 1b - Interrupt enabled
8 HLMTIE	High Limit Interrupt Enable This bit enables the High Limit exceeded interrupt if the current result value is greater than the high limit register value. The raw result value is compared to HILIM[HLMT] before the offset register value is subtracted. 0b - Interrupt disabled 1b - Interrupt enabled
7-4 CHNCFG_L	CHCNF (Channel Configure Low) bits The bits configure the analog inputs for either single-ended or differential conversions. Differential conversions can be fully differential or unipolar based on the value of CTRL3[UPDEN_L]. Fully differential measurements return the max value ((2**12)-1) when the + input is V_{REFH} and the - input is V_{REFLO}, return 0 when the + input is at V_{REFLO} and the - input is at V_{REFH}, and scale linearly between based on the voltage difference between the two signals. Unipolar differential measurements are only positive and return the max value ((2**12)-1) when the + input is V_{REFH} and the - input is V_{REFLO}, return 0 when the + input and - input are the same voltage, and scale linearly between based on the positive voltage difference between the two signals. single-ended measurements return the max value when the input is at V_{REFH}, return 0 when the input is at V_{REFLO}, and scale linearly between based on the amount by which the input exceeds V_{REFLO}. 0xxxb - Inputs = ANB2-ANB3 : Both configured as single-ended inputs 1xxxb - Inputs = ANB2-ANB3 : Configured as differential pair (ANB2 is + and ANB3 is -) x0xxb - Inputs = ANB0-ANB1 : Both configured as single-ended inputs x1xxb - Inputs = ANB0-ANB1 : Configured as differential pair (ANB0 is + and ANB1 is -) xx0xb - Inputs = ANA2-ANA3 : Both configured as single-ended inputs xx1xb - Inputs = ANA2-ANA3 : Configured as differential pair (ANA2 is + and ANA3 is -) xxx0b - Inputs = ANA0-ANA1 : Both configured as single-ended inputs xxx1b - Inputs = ANA0-ANA1 : Configured as differential pair (ANA0 is + and ANA1 is -)
3 —	RESERVED
2-0 SMODE	ADC Scan Mode Control This field controls the ADC module's scan mode. All scan modes use 16 sample slots defined by the CLIST1-4 registers. A scan is the process of stepping through a subset of these sample slots, converting the input indicated by a slot, and storing the result. Unused slots may be disabled using the SDIS register. Input pairs ANA0-1, ANA2-3, ANA4-5, ANA6-7, ANB0-1, ANB2-3, ANB4-5, and ANB6-7 may be configured as differential pairs using the CHNCFG fields. When a slot refers to either member of a differential pair, a differential measurement on that pair is made; otherwise, a single-ended measurement is taken on that input. The CTRL*[CHNCFG] fields' descriptions detail differential and single-ended measurement. Optionally, up to 4 additional sample slots defined by CLIST5, can be appended to the end of the scan. These sample slots are used specifically for single-ended conversions of on-chip generated analog signals such as the temperature sensor output or the voltage regulator bandgap reference. Unused slots may be disabled using the SDIS2 register. Only single-ended measurements of these on-chip signals are supported. The SMODE field determines whether the slots are used to perform one long sequential scan or two shorter parallel scans, each performed by one of the two converters. SMODE controls how these scans are initiated and terminated. It also controls whether the scans are performed once or repetitively. For details, refer to Sequential versus Parallel Sampling and Scan Sequencing .

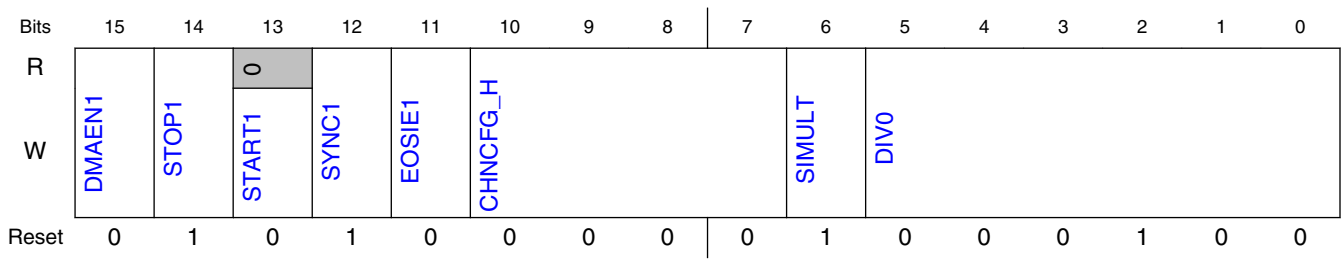
Field	Function
	Parallel scans may be simultaneous (CTRL2[SIMULT] is 1) or non-simultaneous. Simultaneous parallel scans perform the A and B converter scan in lock step using one set of shared controls. Non-simultaneous parallel scans operate the A and B converters independently, with each converter using its own set of controls. Refer to the CTRL2[SIMULT] bit's description for details. Setting any sequential mode overrides the setting of CTRL2[SIMULT]. 000b - **Once (single) sequential** - Upon start or an enabled sync signal, samples are taken one at a time starting with CLIST1[SAMPLE0], until the first disabled sample is encountered. If no disabled sample is encountered, conversion concludes after CLIST4[SAMPLE15]. If CLIST5[SAMPLE16] is enabled in SDIS2 then the scan will continue until the first disabled sample is encountered or when all 4 additional samples are completed. If the scan is initiated by a SYNC signal, only one scan is completed because the CTRL*[SYNC*] bit is cleared automatically by the initial SYNC detection. CTRL*[SYNC*] can be set again at any time during the scan. 001b - **Once parallel** - Upon start or an armed and enabled sync signal: In parallel, converter A converts SAMPLEs 0-7 , and converter B converts SAMPLEs 8-15 . When CTRL2[SIMULT] is 1 (default), scanning stops when either converter encounters a disabled sample or both converters complete all 8 samples. When CTRL2[SIMULT] is 0, a converter stops scanning when it encounters a disabled sample or completes all 8 samples. If additional samples are enabled in SDIS2 then the parallel scan will continue with converter A converting SAMPLEs 16-17 and convert B converting SAMPLEs 18-19, until the first disabled sample is encountered or when each converter completes 2 additional samples. If the scan is initiated by a SYNC signal, only one scan is completed because the CTRL*[SYNC*] bit is cleared automatically by the initial SYNC detection. CTRL*[SYNC*] can be set again at any time during the scan. If CTRL2[SIMULT] is 0, the B converter must be rearmed by writing the CTRL2[SYNC1] bit. 010b - **Loop sequential** - Upon an initial start or enabled sync pulse, up to 16 samples in the order SAMPLEs 0-15 are taken one at a time until a disabled sample is encountered. If additional samples are enabled in the SDIS2 register, the scan will continue with SAMPLEs 16-19 until a disabled sample is encountered. The process repeats perpetually until the CTRL1[STOP0] bit is set. While a loop mode is running, any additional start commands or sync pulses are ignored unless the scan is paused using the SCTRL[SC*] bits. If PWR[APD] is the selected power mode control, PWR[PUDELAY] is applied only on the first conversion. 011b - **Loop parallel** - Upon an initial start or enabled sync pulse, converter A converts SAMPLEs 0-7 , and converter B converts SAMPLEs 8-15 . If additional samples are enabled in SDIS2 then the parallel scan will continue with converter A converting SAMPLEs 16-17 and convert B converting SAMPLEs 18-19, until the first disabled sample is encountered or when each converter completes 2 additional samples. Each time a converter completes its current scan, it immediately restarts its scan sequence. This process continues until the CTRL*[STOP*] bit is asserted. While a loop is running, any additional start commands or sync pulses are ignored unless the scan is paused using the SCTRL[SC*] bits. When CTRL2[SIMULT] is 1 (default), scanning restarts when either converter encounters a disabled sample. When CTRL2[SIMULT] is 0, a converter restarts scanning when it encounters a disabled sample. If PWR[APD] is the selected power mode control, PWR[PUDELAY] is applied only on the first conversion. 100b - **Triggered sequential** - Upon start or an enabled sync signal, samples are taken one at a time starting with CLIST1[SAMPLE0], until the first disabled sample is encountered. If no disabled sample is encountered, conversion concludes after CLIST4[SAMPLE15]. If CLIST5[SAMPLE16] is enabled in SDIS2 then the scan will continue until the first disabled sample is encountered or when all 4 additional samples are completed. If external sync is enabled, new scans start for each SYNC pulse that does not overlap with a current scan in progress. 101b - **Triggered parallel (default)** - Upon start or an enabled sync signal: In parallel, converter A converts SAMPLEs 0-7 , and converter B converts SAMPLEs 8-15 . When CTRL2[SIMULT] is 1 (default), scanning stops when either converter encounters a disabled sample. When CTRL2[SIMULT] is 0, a converter stops scanning when it encounters a disabled sample. If additional samples are enabled in SDIS2 then the parallel scan will continue with converter A converting SAMPLEs 16-17 and convert B converting SAMPLEs 18-19, until the first disabled sample is encountered or when each converter completes 2 additional samples. If external sync is enabled, new scans start for each SYNC pulse that does not overlap with a current scan in progress. 11xb - Reserved

22.7.1.3 ADC Control Register 2 (CTRL2)

22.7.1.3.1 Offset

Register	Offset
CTRL2	1h

22.7.1.3.2 Diagram



22.7.1.3.3 Fields

Field	Function
15 DMAEN1	DMA enable During parallel scan modes when SIMULT=0, this bit enables DMA for converter B. When this bit is asserted, the DMA source selected by CTRL3[DMASRC] causes the conversion results to be transferred by the DMA controller. Setting this bit blocks the generation of the EOSI1 end of scan interrupt. 0b - DMA is not enabled. 1b - DMA is enabled.
14 STOP1	Stop During parallel scan modes when SIMULT = 0, this bit enables stop control of a B converter parallel scan. When this bit is asserted, the current scan is stopped and no further scans can start. Any further SYNC1 input pulses (see CTRL2[SYNC1] bit) or writes to the CTRL2[START1] bit are ignored until this bit has been cleared. After the ADC is in stop mode, the results registers can be modified by the processor. Any changes to the result registers in stop mode are treated as if the analog core supplied the data. Therefore, limit checking, zero crossing, and associated interrupts can occur when authorized. **This is not the same as the chip STOP mode.** 0b - Normal operation 1b - Stop mode
13 START1	START1 Conversion During parallel scan modes when SIMULT = 0, this bit enables start control of a B converter parallel scan.

Table continues on the next page...

Field	Function
	A scan is started by writing 1 to this bit. This is a write only bit. Writing 1 to it again while the scan remains in process, is ignored. The ADC must be in a stable power configuration prior to writing the start bit. Refer to the functional description of power modes for further details. 0b - No action 1b - Start command is issued
12 SYNC1	SYNC1 Enable During parallel scan modes when CTRL2[SIMULT]=0, setting this bit to 1 permits a B converter parallel scan to be initiated by asserting the SYNC1 input for at least one ADC clock cycle. CTRL2[SYNC1] is cleared in ONCE mode, CTRL1[SMODE=000 or 001], when the first SYNC input is detected. This prevents unintentionally starting a new scan after the first scan has completed. The ADC must be in a stable power mode prior to SYNC1 input assertion. Refer to the functional description of power modes for further details. In "once" scan modes, only a first SYNC1 input pulse is honored. CTRL2[SYNC1] is cleared in this mode when the first SYNC1 input is detected. This prevents unintentionally starting a new scan after the first scan has completed. The CTRL2[SYNC1] bit can be set again at any time including while the scan remains in process. 0b - B converter parallel scan is initiated by a write to CTRL2[START1] bit only 1b - Use a SYNC1 input pulse or CTRL2[START1] bit to initiate a B converter parallel scan
11 EOSIE1	End Of Scan Interrupt Enable During parallel scan modes when SIMULT = 0, this bit enables interrupt control for a B converter parallel scan. This bit enables an EOSI1 interrupt to be generated upon completion of the scan. For looping scan modes, the interrupt will trigger after the completion of each iteration of the loop. 0b - Interrupt disabled 1b - Interrupt enabled
10-7 CHNCFG_H	CHCNF (Channel Configure High) bits The bits configure the analog inputs for either single-ended or differential conversions. Differential conversions can be fully differential or unipolar based on the value of CTRL3[UPDEN_H]. Fully differential measurements return the max value ((2**12)-1) when the + input is V_{REFH} and the - input is V_{REFLO}, return 0 when the + input is at V_{REFLO} and the - input is at V_{REFH}, and scale linearly between based on the voltage difference between the two signals. Unipolar differential measurements are only positive and return the max value ((2**12)-1) when the + input is V_{REFH} and the - input is V_{REFLO}, return 0 when the + input and - input are the same voltage, and scale linearly between based on the positive voltage difference between the two signals. single-ended measurements return the max value when the input is at V_{REFH}, return 0 when the input is at V_{REFLO}, and scale linearly between based on the amount by which the input exceeds V_{REFLO}. 0xxxb - Inputs = ANB6-ANB7 : Both configured as single-ended inputs 1xxxb - Inputs = ANB6-ANB7 : Configured as differential pair (ANB6 is + and ANB7 is -) x0xxb - Inputs = ANB4-ANB5 : Both configured as single-ended inputs x1xxb - Inputs = ANB4-ANB5 : Configured as differential pair (ANB4 is + and ANB5 is -) xx0xb - Inputs = ANA6-ANA7 : Both configured as single-ended inputs xx1xb - Inputs = ANA6-ANA7 : Configured as differential pair (ANA6 is + and ANA7 is -) xxx0b - Inputs = ANA4-ANA5 : Both configured as single-ended inputs xxx1b - Inputs = ANA4-ANA5 : Configured as differential pair (ANA4 is + and ANA5 is -)
6 SIMULT	Simultaneous mode This bit only affects parallel scan modes. By default (CTRL2[SIMULT]=1) parallel scans operate in simultaneous mode. The scans in the A and B converter operate simultaneously and always result in pairs of simultaneous conversions in the A and B converter. CTRL1[STOP0, SYNC0, and START0] control bits and the SYNC0 input are used to start and stop scans in both converters simultaneously. A

Table continues on the next page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>scan ends in both converters when either converter encounters a disabled sample slot. When the parallel scan completes, the STAT[EOSI0] triggers if CTRL1[EOSIE0] is set. The STAT[CIP0] status bit indicates that a parallel scan is in process.

When CTRL2[SIMULT]=0, parallel scans in the A and B converters operate independently. The B converter has its own independent set of the above controls (with a 1 suffix) which control its operation and report its status. Each converter's scan continues until its sample list is exhausted (8 samples) or a disabled sample IN ITS LIST is encountered. For looping parallel scan mode, each converter starts its next iteration as soon as the previous iteration in that converter is complete and continues until the CTRL*[STOP*] bit for that converter is asserted.

0b - Parallel scans done independently
1b - Parallel scans done simultaneously (default)</td></tr>
<tr><td>5-0

DIV0</td><td>Clock Divisor Select

The divider circuit generates the ADC clock by dividing the system clock:
• When DIV0 is 0, the divisor is 2.
• For all other DIV0 values, the divisor is 1 more than the decimal value of DIV0: (DIV0) + 1d.

A DIV0 value must be chosen so the ADC clock does not exceed the maximum frequency.

This clock is used by ADCA during all scans and is used by ADCB during sequential scan modes and during parallel simultaneous scan modes.

The following table shows ADC clock frequency based on the value of DIV0 for these various OCCS configurations.

<table>
<tr><th>DIV0</th><th>Divisor</th><th>ROSC Normal 8 MHz</th><th>PLL (BUS_CLK = 50 MHz)</th><th>External CLK</th></tr>
<tr><td>00_0000</td><td>2</td><td>2.00M</td><td>25M</td><td>CLK/4</td></tr>
<tr><td>00_0001</td><td>2</td><td>2.00M</td><td>25M</td><td>CLK/4</td></tr>
<tr><td>00_0010</td><td>3</td><td>1.33M</td><td>16.67M</td><td>CLK/6</td></tr>
<tr><td>00_0011</td><td>4</td><td>1.00M</td><td>12.5M</td><td>CLK/8</td></tr>
<tr><td>00_0100</td><td>5</td><td>800K</td><td>10M</td><td>CLK/10</td></tr>
<tr><td>00_0101</td><td>6</td><td>667K</td><td>8.33M</td><td>CLK/12</td></tr>
<tr><td>-</td><td>-</td><td>-</td><td>-</td><td></td></tr>
<tr><td>-</td><td>-</td><td>-</td><td>-</td><td></td></tr>
<tr><td>11_1111</td><td>64</td><td>62.5K</td><td>781.25K</td><td>CLK/128</td></tr>
</table>
</td></tr>
</table>

22.7.1.4 ADC Zero Crossing Control 1 Register (ZXCTRL1)

22.7.1.4.1 Offset

Register	Offset
ZXCTRL1	2h

22.7.1.4.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ZCE7		ZCE6		ZCE5		ZCE4		ZCE3		ZCE2		ZCE1		ZCE0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22.7.1.4.3 Fields

Field	Function
15-14 ZCE7	Zero crossing enable 7 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change
13-12 ZCE6	Zero crossing enable 6 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change
11-10 ZCE5	Zero crossing enable 5 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change
9-8 ZCE4	Zero crossing enable 4 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change
7-6 ZCE3	Zero crossing enable 3 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change
5-4 ZCE2	Zero crossing enable 2 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change
3-2 ZCE1	Zero crossing enable 1 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change
1-0 ZCE0	Zero crossing enable 0 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change

22.7.1.5 ADC Zero Crossing Control 2 Register (ZXCTRL2)

22.7.1.5.1 Offset

Register	Offset
ZXCTRL2	3h

22.7.1.5.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ZCE15		ZCE14		ZCE13		ZCE12		ZCE11		ZCE10		ZCE9		ZCE8	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22.7.1.5.3 Fields

Field	Function
15-14 ZCE15	Zero crossing enable 15 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change
13-12 ZCE14	Zero crossing enable 14 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change
11-10 ZCE13	Zero crossing enable 13 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change
9-8 ZCE12	Zero crossing enable 12 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change
7-6 ZCE11	Zero crossing enable 11 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change
5-4	Zero crossing enable 10

Table continues on the next page...

Field	Function
ZCE10	00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change
3-2 ZCE9	Zero crossing enable 9 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change
1-0 ZCE8	Zero crossing enable 8 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change

22.7.1.6 ADC Channel List Register 1 (CLIST1)

22.7.1.6.1 Offset

Register	Offset
CLIST1	4h

22.7.1.6.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SAMPLE3				SAMPLE2				SAMPLE1				SAMPLE0			
Reset	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

22.7.1.6.3 Fields

Field	Function
15-12 SAMPLE3	Sample Field 3 This sample and all subsequent samples can be disabled by setting ADC_SDIS[3]. 0000b - single-ended: ANA0, Differential: ANA0+, ANA1- 0001b - single-ended: ANA1, Differential: ANA0+, ANA1- 0010b - single-ended: ANA2, Differential: ANA2+, ANA3- 0011b - single-ended: ANA3, Differential: ANA2+, ANA3- 0100b - single-ended: ANA4, Differential: ANA4+, ANA5- 0101b - single-ended: ANA5, Differential: ANA4+, ANA5- 0110b - single-ended: ANA6, Differential: ANA6+, ANA7-

Table continues on the next page...

Field	Function
	0111b - single-ended: ANA7, Differential: ANA6+, ANA7- 1000b - single-ended: ANB0, Differential: ANB0+, ANB1- 1001b - single-ended: ANB1, Differential: ANB0+, ANB1- 1010b - single-ended: ANB2, Differential: ANB2+, ANB3- 1011b - single-ended: ANB3, Differential: ANB2+, ANB3- 1100b - single-ended: ANB4, Differential: ANB4+, ANB5- 1101b - single-ended: ANB5, Differential: ANB4+, ANB5- 1110b - single-ended: ANB6, Differential: ANB6+, ANB7- 1111b - single-ended: ANB7, Differential: ANB6+, ANB7-
11-8 SAMPLE2	Sample Field 2 This sample and all subsequent samples can be disabled by setting ADC_SDIS[2]. 0000b - single-ended: ANA0, Differential: ANA0+, ANA1- 0001b - single-ended: ANA1, Differential: ANA0+, ANA1- 0010b - single-ended: ANA2, Differential: ANA2+, ANA3- 0011b - single-ended: ANA3, Differential: ANA2+, ANA3- 0100b - single-ended: ANA4, Differential: ANA4+, ANA5- 0101b - single-ended: ANA5, Differential: ANA4+, ANA5- 0110b - single-ended: ANA6, Differential: ANA6+, ANA7- 0111b - single-ended: ANA7, Differential: ANA6+, ANA7- 1000b - single-ended: ANB0, Differential: ANB0+, ANB1- 1001b - single-ended: ANB1, Differential: ANB0+, ANB1- 1010b - single-ended: ANB2, Differential: ANB2+, ANB3- 1011b - single-ended: ANB3, Differential: ANB2+, ANB3- 1100b - single-ended: ANB4, Differential: ANB4+, ANB5- 1101b - single-ended: ANB5, Differential: ANB4+, ANB5- 1110b - single-ended: ANB6, Differential: ANB6+, ANB7- 1111b - single-ended: ANB7, Differential: ANB6+, ANB7-
7-4 SAMPLE1	Sample Field 1 This sample and all subsequent samples can be disabled by setting ADC_SDIS[1]. 0000b - single-ended: ANA0, Differential: ANA0+, ANA1- 0001b - single-ended: ANA1, Differential: ANA0+, ANA1- 0010b - single-ended: ANA2, Differential: ANA2+, ANA3- 0011b - single-ended: ANA3, Differential: ANA2+, ANA3- 0100b - single-ended: ANA4, Differential: ANA4+, ANA5- 0101b - single-ended: ANA5, Differential: ANA4+, ANA5- 0110b - single-ended: ANA6, Differential: ANA6+, ANA7- 0111b - single-ended: ANA7, Differential: ANA6+, ANA7- 1000b - single-ended: ANB0, Differential: ANB0+, ANB1- 1001b - single-ended: ANB1, Differential: ANB0+, ANB1- 1010b - single-ended: ANB2, Differential: ANB2+, ANB3- 1011b - single-ended: ANB3, Differential: ANB2+, ANB3- 1100b - single-ended: ANB4, Differential: ANB4+, ANB5- 1101b - single-ended: ANB5, Differential: ANB4+, ANB5- 1110b - single-ended: ANB6, Differential: ANB6+, ANB7- 1111b - single-ended: ANB7, Differential: ANB6+, ANB7-
3-0 SAMPLE0	Sample Field 0 This sample and all subsequent samples can be disabled by setting ADC_SDIS[0]. 0000b - single-ended: ANA0, Differential: ANA0+, ANA1- 0001b - single-ended: ANA1, Differential: ANA0+, ANA1- 0010b - single-ended: ANA2, Differential: ANA2+, ANA3- 0011b - single-ended: ANA3, Differential: ANA2+, ANA3- 0100b - single-ended: ANA4, Differential: ANA4+, ANA5- 0101b - single-ended: ANA5, Differential: ANA4+, ANA5- 0110b - single-ended: ANA6, Differential: ANA6+, ANA7-

Field	Function
	0111b - single-ended: ANA7, Differential: ANA6+, ANA7- 1000b - single-ended: ANB0, Differential: ANB0+, ANB1- 1001b - single-ended: ANB1, Differential: ANB0+, ANB1- 1010b - single-ended: ANB2, Differential: ANB2+, ANB3- 1011b - single-ended: ANB3, Differential: ANB2+, ANB3- 1100b - single-ended: ANB4, Differential: ANB4+, ANB5- 1101b - single-ended: ANB5, Differential: ANB4+, ANB5- 1110b - single-ended: ANB6, Differential: ANB6+, ANB7- 1111b - single-ended: ANB7, Differential: ANB6+, ANB7-

22.7.1.7 ADC Channel List Register 2 (CLIST2)

22.7.1.7.1 Offset

Register	Offset
CLIST2	5h

22.7.1.7.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SAMPLE7				SAMPLE6				SAMPLE5				SAMPLE4			
Reset	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0

22.7.1.7.3 Fields

Field	Function
15-12 SAMPLE7	Sample Field 7 This sample and all subsequent samples can be disabled by setting ADC_SDIS[7]. 0000b - single-ended: ANA0, Differential: ANA0+, ANA1- 0001b - single-ended: ANA1, Differential: ANA0+, ANA1- 0010b - single-ended: ANA2, Differential: ANA2+, ANA3- 0011b - single-ended: ANA3, Differential: ANA2+, ANA3- 0100b - single-ended: ANA4, Differential: ANA4+, ANA5- 0101b - single-ended: ANA5, Differential: ANA4+, ANA5- 0110b - single-ended: ANA6, Differential: ANA6+, ANA7- 0111b - single-ended: ANA7, Differential: ANA6+, ANA7- 1000b - single-ended: ANB0, Differential: ANB0+, ANB1- 1001b - single-ended: ANB1, Differential: ANB0+, ANB1- 1010b - single-ended: ANB2, Differential: ANB2+, ANB3- 1011b - single-ended: ANB3, Differential: ANB2+, ANB3- 1100b - single-ended: ANB4, Differential: ANB4+, ANB5-

Table continues on the next page...

Field	Function
	1101b - single-ended: ANB5, Differential: ANB4+, ANB5- 1110b - single-ended: ANB6, Differential: ANB6+, ANB7- 1111b - single-ended: ANB7, Differential: ANB6+, ANB7-
11-8 SAMPLE6	Sample Field 6 This sample and all subsequent samples can be disabled by setting ADC_SDIS[6]. 0000b - single-ended: ANA0, Differential: ANA0+, ANA1- 0001b - single-ended: ANA1, Differential: ANA0+, ANA1- 0010b - single-ended: ANA2, Differential: ANA2+, ANA3- 0011b - single-ended: ANA3, Differential: ANA2+, ANA3- 0100b - single-ended: ANA4, Differential: ANA4+, ANA5- 0101b - single-ended: ANA5, Differential: ANA4+, ANA5- 0110b - single-ended: ANA6, Differential: ANA6+, ANA7- 0111b - single-ended: ANA7, Differential: ANA6+, ANA7- 1000b - single-ended: ANB0, Differential: ANB0+, ANB1- 1001b - single-ended: ANB1, Differential: ANB0+, ANB1- 1010b - single-ended: ANB2, Differential: ANB2+, ANB3- 1011b - single-ended: ANB3, Differential: ANB2+, ANB3- 1100b - single-ended: ANB4, Differential: ANB4+, ANB5- 1101b - single-ended: ANB5, Differential: ANB4+, ANB5- 1110b - single-ended: ANB6, Differential: ANB6+, ANB7- 1111b - single-ended: ANB7, Differential: ANB6+, ANB7-
7-4 SAMPLE5	Sample Field 5 This sample and all subsequent samples can be disabled by setting ADC_SDIS[5]. 0000b - single-ended: ANA0, Differential: ANA0+, ANA1- 0001b - single-ended: ANA1, Differential: ANA0+, ANA1- 0010b - single-ended: ANA2, Differential: ANA2+, ANA3- 0011b - single-ended: ANA3, Differential: ANA2+, ANA3- 0100b - single-ended: ANA4, Differential: ANA4+, ANA5- 0101b - single-ended: ANA5, Differential: ANA4+, ANA5- 0110b - single-ended: ANA6, Differential: ANA6+, ANA7- 0111b - single-ended: ANA7, Differential: ANA6+, ANA7- 1000b - single-ended: ANB0, Differential: ANB0+, ANB1- 1001b - single-ended: ANB1, Differential: ANB0+, ANB1- 1010b - single-ended: ANB2, Differential: ANB2+, ANB3- 1011b - single-ended: ANB3, Differential: ANB2+, ANB3- 1100b - single-ended: ANB4, Differential: ANB4+, ANB5- 1101b - single-ended: ANB5, Differential: ANB4+, ANB5- 1110b - single-ended: ANB6, Differential: ANB6+, ANB7- 1111b - single-ended: ANB7, Differential: ANB6+, ANB7-
3-0 SAMPLE4	Sample Field 4 This sample and all subsequent samples can be disabled by setting ADC_SDIS[4]. 0000b - single-ended: ANA0, Differential: ANA0+, ANA1- 0001b - single-ended: ANA1, Differential: ANA0+, ANA1- 0010b - single-ended: ANA2, Differential: ANA2+, ANA3- 0011b - single-ended: ANA3, Differential: ANA2+, ANA3- 0100b - single-ended: ANA4, Differential: ANA4+, ANA5- 0101b - single-ended: ANA5, Differential: ANA4+, ANA5- 0110b - single-ended: ANA6, Differential: ANA6+, ANA7- 0111b - single-ended: ANA7, Differential: ANA6+, ANA7- 1000b - single-ended: ANB0, Differential: ANB0+, ANB1- 1001b - single-ended: ANB1, Differential: ANB0+, ANB1- 1010b - single-ended: ANB2, Differential: ANB2+, ANB3- 1011b - single-ended: ANB3, Differential: ANB2+, ANB3- 1100b - single-ended: ANB4, Differential: ANB4+, ANB5-

Field	Function
	1101b - single-ended: ANB5, Differential: ANB4+, ANB5- 1110b - single-ended: ANB6, Differential: ANB6+, ANB7- 1111b - single-ended: ANB7, Differential: ANB6+, ANB7-

22.7.1.8 ADC Channel List Register 3 (CLIST3)

22.7.1.8.1 Offset

Register	Offset
CLIST3	6h

22.7.1.8.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SAMPLE11				SAMPLE10				SAMPLE9				SAMPLE8			
Reset	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0

22.7.1.8.3 Fields

Field	Function
15-12 SAMPLE11	Sample Field 11 This sample and all subsequent samples can be disabled by setting ADC_SDIS[11]. 0000b - single-ended: ANA0, Differential: ANA0+, ANA1- 0001b - single-ended: ANA1, Differential: ANA0+, ANA1- 0010b - single-ended: ANA2, Differential: ANA2+, ANA3- 0011b - single-ended: ANA3, Differential: ANA2+, ANA3- 0100b - single-ended: ANA4, Differential: ANA4+, ANA5- 0101b - single-ended: ANA5, Differential: ANA4+, ANA5- 0110b - single-ended: ANA6, Differential: ANA6+, ANA7- 0111b - single-ended: ANA7, Differential: ANA6+, ANA7- 1000b - single-ended: ANB0, Differential: ANB0+, ANB1- 1001b - single-ended: ANB1, Differential: ANB0+, ANB1- 1010b - single-ended: ANB2, Differential: ANB2+, ANB3- 1011b - single-ended: ANB3, Differential: ANB2+, ANB3- 1100b - single-ended: ANB4, Differential: ANB4+, ANB5- 1101b - single-ended: ANB5, Differential: ANB4+, ANB5- 1110b - single-ended: ANB6, Differential: ANB6+, ANB7- 1111b - single-ended: ANB7, Differential: ANB6+, ANB7-
11-8	Sample Field 10

Table continues on the next page...

Field	Function
SAMPLE10	This sample and all subsequent samples can be disabled by setting ADC_SDIS[10]. 0000b - single-ended: ANA0, Differential: ANA0+, ANA1- 0001b - single-ended: ANA1, Differential: ANA0+, ANA1- 0010b - single-ended: ANA2, Differential: ANA2+, ANA3- 0011b - single-ended: ANA3, Differential: ANA2+, ANA3- 0100b - single-ended: ANA4, Differential: ANA4+, ANA5- 0101b - single-ended: ANA5, Differential: ANA4+, ANA5- 0110b - single-ended: ANA6, Differential: ANA6+, ANA7- 0111b - single-ended: ANA7, Differential: ANA6+, ANA7- 1000b - single-ended: ANB0, Differential: ANB0+, ANB1- 1001b - single-ended: ANB1, Differential: ANB0+, ANB1- 1010b - single-ended: ANB2, Differential: ANB2+, ANB3- 1011b - single-ended: ANB3, Differential: ANB2+, ANB3- 1100b - single-ended: ANB4, Differential: ANB4+, ANB5- 1101b - single-ended: ANB5, Differential: ANB4+, ANB5- 1110b - single-ended: ANB6, Differential: ANB6+, ANB7- 1111b - single-ended: ANB7, Differential: ANB6+, ANB7-
7-4 SAMPLE9	Sample Field 9 This sample and all subsequent samples can be disabled by setting ADC_SDIS[9]. 0000b - single-ended: ANA0, Differential: ANA0+, ANA1- 0001b - single-ended: ANA1, Differential: ANA0+, ANA1- 0010b - single-ended: ANA2, Differential: ANA2+, ANA3- 0011b - single-ended: ANA3, Differential: ANA2+, ANA3- 0100b - single-ended: ANA4, Differential: ANA4+, ANA5- 0101b - single-ended: ANA5, Differential: ANA4+, ANA5- 0110b - single-ended: ANA6, Differential: ANA6+, ANA7- 0111b - single-ended: ANA7, Differential: ANA6+, ANA7- 1000b - single-ended: ANB0, Differential: ANB0+, ANB1- 1001b - single-ended: ANB1, Differential: ANB0+, ANB1- 1010b - single-ended: ANB2, Differential: ANB2+, ANB3- 1011b - single-ended: ANB3, Differential: ANB2+, ANB3- 1100b - single-ended: ANB4, Differential: ANB4+, ANB5- 1101b - single-ended: ANB5, Differential: ANB4+, ANB5- 1110b - single-ended: ANB6, Differential: ANB6+, ANB7- 1111b - single-ended: ANB7, Differential: ANB6+, ANB7-
3-0 SAMPLE8	Sample Field 8 This sample and all subsequent samples can be disabled by setting ADC_SDIS[8]. 0000b - single-ended: ANA0, Differential: ANA0+, ANA1- 0001b - single-ended: ANA1, Differential: ANA0+, ANA1- 0010b - single-ended: ANA2, Differential: ANA2+, ANA3- 0011b - single-ended: ANA3, Differential: ANA2+, ANA3- 0100b - single-ended: ANA4, Differential: ANA4+, ANA5- 0101b - single-ended: ANA5, Differential: ANA4+, ANA5- 0110b - single-ended: ANA6, Differential: ANA6+, ANA7- 0111b - single-ended: ANA7, Differential: ANA6+, ANA7- 1000b - single-ended: ANB0, Differential: ANB0+, ANB1- 1001b - single-ended: ANB1, Differential: ANB0+, ANB1- 1010b - single-ended: ANB2, Differential: ANB2+, ANB3- 1011b - single-ended: ANB3, Differential: ANB2+, ANB3- 1100b - single-ended: ANB4, Differential: ANB4+, ANB5- 1101b - single-ended: ANB5, Differential: ANB4+, ANB5- 1110b - single-ended: ANB6, Differential: ANB6+, ANB7- 1111b - single-ended: ANB7, Differential: ANB6+, ANB7-

22.7.1.9 ADC Channel List Register 4 (CLIST4)

22.7.1.9.1 Offset

Register	Offset
CLIST4	7h

22.7.1.9.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SAMPLE15				SAMPLE14				SAMPLE13				SAMPLE12			
Reset	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0

22.7.1.9.3 Fields

Field	Function
15-12 SAMPLE15	Sample Field 15 This sample and all subsequent samples can be disabled by setting ADC_SDIS[15]. 0000b - single-ended: ANA0, Differential: ANA0+, ANA1- 0001b - single-ended: ANA1, Differential: ANA0+, ANA1- 0010b - single-ended: ANA2, Differential: ANA2+, ANA3- 0011b - single-ended: ANA3, Differential: ANA2+, ANA3- 0100b - single-ended: ANA4, Differential: ANA4+, ANA5- 0101b - single-ended: ANA5, Differential: ANA4+, ANA5- 0110b - single-ended: ANA6, Differential: ANA6+, ANA7- 0111b - single-ended: ANA7, Differential: ANA6+, ANA7- 1000b - single-ended: ANB0, Differential: ANB0+, ANB1- 1001b - single-ended: ANB1, Differential: ANB0+, ANB1- 1010b - single-ended: ANB2, Differential: ANB2+, ANB3- 1011b - single-ended: ANB3, Differential: ANB2+, ANB3- 1100b - single-ended: ANB4, Differential: ANB4+, ANB5- 1101b - single-ended: ANB5, Differential: ANB4+, ANB5- 1110b - single-ended: ANB6, Differential: ANB6+, ANB7- 1111b - single-ended: ANB7, Differential: ANB6+, ANB7-
11-8 SAMPLE14	Sample Field 14 This sample and all subsequent samples can be disabled by setting ADC_SDIS[14]. 0000b - single-ended: ANA0, Differential: ANA0+, ANA1- 0001b - single-ended: ANA1, Differential: ANA0+, ANA1- 0010b - single-ended: ANA2, Differential: ANA2+, ANA3- 0011b - single-ended: ANA3, Differential: ANA2+, ANA3- 0100b - single-ended: ANA4, Differential: ANA4+, ANA5- 0101b - single-ended: ANA5, Differential: ANA4+, ANA5-

Table continues on the next page...

Field	Function
	0110b - single-ended: ANA6, Differential: ANA6+, ANA7- 0111b - single-ended: ANA7, Differential: ANA6+, ANA7- 1000b - single-ended: ANB0, Differential: ANB0+, ANB1- 1001b - single-ended: ANB1, Differential: ANB0+, ANB1- 1010b - single-ended: ANB2, Differential: ANB2+, ANB3- 1011b - single-ended: ANB3, Differential: ANB2+, ANB3- 1100b - single-ended: ANB4, Differential: ANB4+, ANB5- 1101b - single-ended: ANB5, Differential: ANB4+, ANB5- 1110b - single-ended: ANB6, Differential: ANB6+, ANB7- 1111b - single-ended: ANB7, Differential: ANB6+, ANB7-
7-4 SAMPLE13	Sample Field 13 This sample and all subsequent samples can be disabled by setting ADC_SDIS[13]. 0000b - single-ended: ANA0, Differential: ANA0+, ANA1- 0001b - single-ended: ANA1, Differential: ANA0+, ANA1- 0010b - single-ended: ANA2, Differential: ANA2+, ANA3- 0011b - single-ended: ANA3, Differential: ANA2+, ANA3- 0100b - single-ended: ANA4, Differential: ANA4+, ANA5- 0101b - single-ended: ANA5, Differential: ANA4+, ANA5- 0110b - single-ended: ANA6, Differential: ANA6+, ANA7- 0111b - single-ended: ANA7, Differential: ANA6+, ANA7- 1000b - single-ended: ANB0, Differential: ANB0+, ANB1- 1001b - single-ended: ANB1, Differential: ANB0+, ANB1- 1010b - single-ended: ANB2, Differential: ANB2+, ANB3- 1011b - single-ended: ANB3, Differential: ANB2+, ANB3- 1100b - single-ended: ANB4, Differential: ANB4+, ANB5- 1101b - single-ended: ANB5, Differential: ANB4+, ANB5- 1110b - single-ended: ANB6, Differential: ANB6+, ANB7- 1111b - single-ended: ANB7, Differential: ANB6+, ANB7-
3-0 SAMPLE12	Sample Field 12 This sample and all subsequent samples can be disabled by setting ADC_SDIS[12]. 0000b - single-ended: ANA0, Differential: ANA0+, ANA1- 0001b - single-ended: ANA1, Differential: ANA0+, ANA1- 0010b - single-ended: ANA2, Differential: ANA2+, ANA3- 0011b - single-ended: ANA3, Differential: ANA2+, ANA3- 0100b - single-ended: ANA4, Differential: ANA4+, ANA5- 0101b - single-ended: ANA5, Differential: ANA4+, ANA5- 0110b - single-ended: ANA6, Differential: ANA6+, ANA7- 0111b - single-ended: ANA7, Differential: ANA6+, ANA7- 1000b - single-ended: ANB0, Differential: ANB0+, ANB1- 1001b - single-ended: ANB1, Differential: ANB0+, ANB1- 1010b - single-ended: ANB2, Differential: ANB2+, ANB3- 1011b - single-ended: ANB3, Differential: ANB2+, ANB3- 1100b - single-ended: ANB4, Differential: ANB4+, ANB5- 1101b - single-ended: ANB5, Differential: ANB4+, ANB5- 1110b - single-ended: ANB6, Differential: ANB6+, ANB7- 1111b - single-ended: ANB7, Differential: ANB6+, ANB7-

22.7.1.10 ADC Sample Disable Register (SDIS)

22.7.1.10.1 Offset

Register	Offset
SDIS	8h

22.7.1.10.2 Function

Each bit of SDIS corresponds to a SAMPLEx field in one or multiple CLISTn registers. For example, bit 0 of SDIS will enable/disable SAMPLE0, while bit 15 of SDIS will enable/disable SAMPLE15.

If an SDIS bit is clear, then the corresponding SAMPLEx field will be enabled in an ADC scan. If the SDIS bit is set, then the corresponding SAMPLEx field will be disabled and the ADC scan will be halted and the subsequent ADC acquisition will also not occur. Be noted that the ADC will sequentially scan in the order SAMPLE0, SAMPLE1, SAMPLE2, ... etc. Thus if SDIS=FFF8h, then an ADC scan will acquire SAMPLE0 then SAMPLE1 and SAMPLE2, but all other acquisitions will not occur. The 4-bit wide SAMPLEx fields are found in the CLISTn registers which select a particular ADC channel and whether it will be a differential or single-ended conversion.

22.7.1.10.3 Diagram

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

22.7.1.10.4 Fields

Field	Function
15-0 DS	Disable Sample Bits 0000_0000_0000_0000b - Bit value 0: enables CLISTn[SAMPLEx]. 0000_0000_0000_0001b - Bit value 1: disables CLISTn[SAMPLEx] and all subsequent samples. Which samples are actually disabled will depend on the conversion mode, sequential/parallel, and the value of CTRL2[SIMULT].

22.7.1.11 ADC Status Register (STAT)

22.7.1.11.1 Offset

Register	Offset
STAT	9h

22.7.1.11.2 Function

This register provides the current status of the ADC module. STAT[HLMTI and LLMTI] bits are cleared by writing 1s to all asserted bits in the limit status register s , LIMSTAT * . Likewise, the STAT[ZCI] bit, is cleared by writing 1s to all asserted bits in the zero crossing status register s , ZXSTAT * . The STAT[EOSIx] bits are cleared by writing a one to them.

Except for STAT[CIP0 and CIP1] this register's bits are sticky. Once set to a one state, they require some specific action to clear them. They are not cleared automatically on the next scan sequence.

22.7.1.11.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CIP0	CIP1	0	EOSI1	EOSI0	ZCI	LLMTI	HLMTI	Reserved							
W				W1C	W1C											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22.7.1.11.4 Fields

Field	Function
15 CIP0	Conversion in Progress This bit indicates whether a scan is in progress. This refers to any scan except a B converter scan in non-simultaneous parallel scan modes. 0b - Idle state 1b - A scan cycle is in progress. The ADC will ignore all sync pulses or start commands
14 CIP1	Conversion in Progress This bit indicates whether a scan is in progress. This refers only to a B converter scan in non-simultaneous parallel scan modes. 0b - Idle state 1b - A scan cycle is in progress. The ADC will ignore all sync pulses or start commands

Table continues on the next page...

Field	Function
13 —	RESERVED
12 EOSI1	End of Scan Interrupt This bit indicates whether a scan of analog inputs have been completed since the last read of the status register or since a reset. This bit is cleared by writing a one to it. This bit cannot be set by software. In looping scan modes, this interrupt is triggered at the completion of each iteration of the loop. This interrupt is triggered only by the completion of a B converter scan in non-simultaneous parallel scan modes. 0b - A scan cycle has not been completed, no end of scan IRQ pending 1b - A scan cycle has been completed, end of scan IRQ pending
11 EOSI0	End of Scan Interrupt This bit indicates whether a scan of analog inputs have been completed since the last read of the status register or since a reset. This bit is cleared by writing a one to it. This bit cannot be set by software. STAT[EOSI0] is the preferred bit to poll for scan completion if interrupts are not enabled. In looping scan modes, this interrupt is triggered at the completion of each iteration of the loop. This interrupt is triggered upon the completion of any scan except for the completion of a B converter scan in non-simultaneous parallel scan modes. 0b - A scan cycle has not been completed, no end of scan IRQ pending 1b - A scan cycle has been completed, end of scan IRQ pending
10 ZCI	Zero Crossing Interrupt If the respective offset register is configured by having a value greater than 0000h, zero crossing checking is enabled. If the offset register is programmed with 7FF8h, the result will always be less than or equal to zero. On the other hand, if 0000h is programmed into the offset register, the result will always be greater than or equal to zero, and no zero crossing can occur because the sign of the result will not change. This interrupt asserts at the completion of an individual conversion which may or may not be the end of a scan. This bit is cleared by writing a "1" to all active ZXSTAT * [ZCS] bits. 0b - No zero crossing interrupt request 1b - Zero crossing encountered, IRQ pending if CTRL1[ZCIE] is set
9 LLMTI	Low Limit Interrupt If the respective low limit register is enabled by having a value other than 0000h, low limit checking is enabled. This interrupt asserts at the completion of an individual conversion which may or may not be the end of a scan. This bit is cleared by writing a "1" to all active LIMSTAT * [LLS] bits. 0b - No low limit interrupt request 1b - Low limit exceeded, IRQ pending if CTRL1[LLMTIE] is set
8 HLMTI	High Limit Interrupt If the respective high limit register is enabled by having a value other than 7FF8h, high limit checking is enabled. This interrupt asserts at the completion of an individual conversion which may or may not be the end of a scan. This bit is cleared by writing a "1" to all active LIMSTAT * [HLS] bits. 0b - No high limit interrupt request 1b - High limit exceeded, IRQ pending if CTRL1[HLMTIE] is set
7-0 —	UNDEFINED This read-only bitfield is undefined and will always contain random data.

22.7.1.12 ADC Ready Register (RDY)

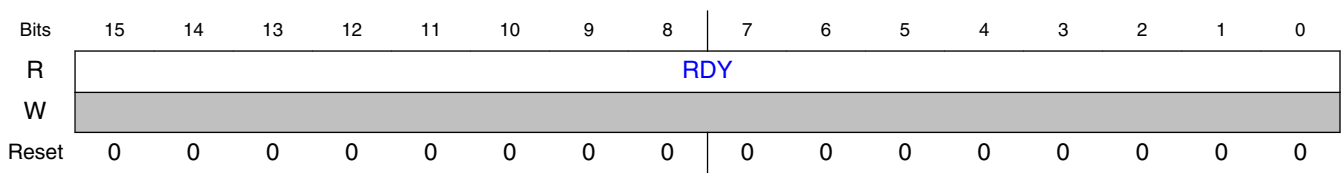
22.7.1.12.1 Offset

Register	Offset
RDY	Ah

22.7.1.12.2 Function

This register provides the current status of the ADC conversions. RDY[RDYx] bits are cleared by reading their corresponding result registers (RSLTx).

22.7.1.12.3 Diagram



22.7.1.12.4 Fields

Field	Function
15-0 RDY	Ready Sample These bits indicate samples fifteen through zero are ready to be read. These bits are cleared after a read from the respective results register. The RDY[RDYn] bits are set as the individual channel conversions are completed. Polling the RDY[RDYn] bits can determine if a particular sample is ready to be read. 0000_0000_0000_0000b - Bit value 0: sample not ready or has been read 0000_0000_0000_0001b - Bit value 1: sample ready to be read

22.7.1.13 ADC Low Limit Status Register (LOLIMSTAT)

22.7.1.13.1 Offset

Register	Offset
LOLIMSTAT	Bh

22.7.1.13.2 Function

The low limit status register latches in the result of the comparison between the result of the sample and the respective low limit register, LOLIM0-15. Here is an example: If the result for the channel programmed in CLIST1[SAMPLE0] is lower than the value programmed into the Low Limit Register zero, then the LIMSTAT[LLS0] bit is set to one. An interrupt is generated if the CTRL1[LLMTIE] bit is set. These bits are sticky. They are not cleared automatically by subsequent conversions. Each bit is cleared only by writing a value of one to that specific bit.

22.7.1.13.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LLS															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22.7.1.13.4 Fields

Field	Function
15-0 LLS	Low Limit Status Bits

22.7.1.14 ADC High Limit Status Register (HILIMSTAT)

22.7.1.14.1 Offset

Register	Offset
HILIMSTAT	Ch

22.7.1.14.2 Function

The high limit status register latches in the result of the comparison between the result of the sample and the respective high limit register, HILIM0-15. Here is an example: If the result for the channel programmed in CLIST1[SAMPLE0] is greater than the value

programmed into the High Limit Register zero, then the LIMSTAT[HLS0] bit is set to one. An interrupt is generated if the CTRL1[HLMTIE] bit is set. These bits are sticky. They are not cleared automatically by subsequent conversions. Each bit is cleared only by writing a value of one to that specific bit.

22.7.1.14.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HLS															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22.7.1.14.4 Fields

Field	Function
15-0 HLS	High Limit Status Bits

22.7.1.15 ADC Zero Crossing Status Register (ZXSTAT)

22.7.1.15.1 Offset

Register	Offset
ZXSTAT	Dh

22.7.1.15.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ZCS															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22.7.1.15.3 Fields

Field	Function
15-0 ZCS	Zero Crossing Status The zero crossing condition is determined by examining the ADC value after it has been adjusted by the offset for the result register. Each bit of the register is cleared by writing a one to that register bit. 0000_0000_0000_0000b - Either: A sign change did not occur in a comparison between the current channel[x] result and the previous channel[x] result, or Zero crossing control is disabled for channel[x] in the zero crossing control register (ZXCTRL) 0000_0000_0000_0001b - In a comparison between the current channel[x] result and the previous channel[x] result, a sign change condition occurred as defined in the zero crossing control register (ZXCTRL)

22.7.1.16 ADC Result Registers with sign extension (RSLT0 - RSLT15)

22.7.1.16.1 Offset

For a = 0 to 15:

Register	Offset
RSLTa	Eh + (a × 1h)

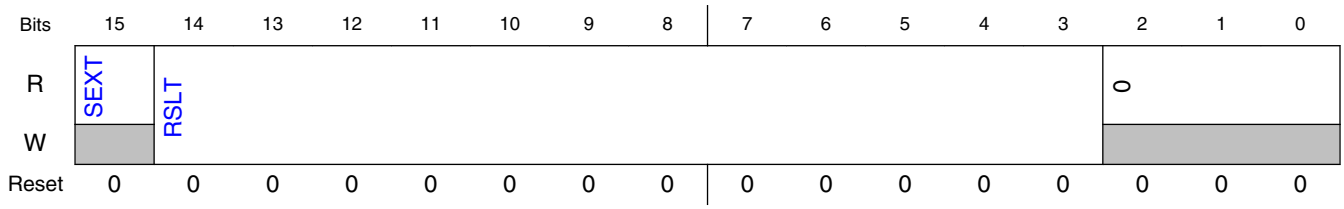
22.7.1.16.2 Function

The result registers contain the converted results from a scan. The CLIST1[SAMPLE0] result is loaded into RSLT0, CLIST1[SAMPLE1] result in RSLT1, and so on. In a parallel scan mode, the first channel pair designated by CLIST1[SAMPLE0] and CLIST3[SAMPLE8] are stored in RSLT0 and RSLT8, respectively.

Note

When writing to this register, only the RSLT portion of the value written is used. This value is modified and the result of the subtraction is stored. The SEXT bit is only set as a result of this subtraction and is not directly determined by the value written.

22.7.1.16.3 Diagram



22.7.1.16.4 Fields

Field	Function
15 SEXT	Sign Extend This is the sign-extend bit of the result. RSLT*[SEXT] set to one implies a negative result. RSLT*[SEXT] set to zero implies a positive result. If unsigned results are required, then the respective offset register must be set to a value of zero.
14-3 RSLT	Digital Result of the Conversion RSLT can be interpreted as either a signed integer or a signed fractional number. As a signed fractional number, the RSLT can be used directly. As a signed integer, it is an option to right shift with sign extend (ASR) three places and interpret the number, or accept the number as presented, knowing there are missing codes. The lower three bits are always going to be zero. Negative results, RSLT*[SEXT] = 1, are always presented in two's complement format. If it is a requirement of your application that the result registers always be positive, the offset registers must always be set to zero. The interpretation of the numbers programmed into the limit and offset registers, LOLIM, HILIM, and OFFST should match your interpretation of the result register.
2-0 —	RESERVED

22.7.1.17 ADC Low Limit Registers (LOLIM0 - LOLIM15)

22.7.1.17.1 Offset

For a = 0 to 15:

Register	Offset
LOLIMa	1Eh + (a × 1h)

22.7.1.17.2 Function

Each ADC sample is compared against the values in the limit registers. The comparison is based upon the raw conversion value with no offset correction applied. The limit register used corresponds to the result register the value will be written to. The high limit register is used for the comparison of Result > High Limit. The low limit register is used for the comparison of Result < Low Limit. The limit checking can be disabled by programming the respective limit register with 7FF8h for the high limit and 0000h for the low limit. At reset, limit checking is disabled.

22.7.1.17.3 Diagram

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

22.7.1.17.4 Fields

Field	Function
15 —	RESERVED
14-3 LLMT	Low Limit Bits
2-0 —	RESERVED

22.7.1.18 ADC High Limit Registers (HILIM0 - HILIM15)

22.7.1.18.1 Offset

For a = 0 to 15:

Register	Offset
HILIMa	2Eh + (a × 1h)

22.7.1.18.2 Function

Each ADC sample is compared against the values in the limit registers. The comparison is based upon the raw conversion value with no offset correction applied. The limit register used corresponds to the result register the value will be written to. The high limit register is used for the comparison of Result > High Limit. The low limit register is used for the comparison of Result < Low Limit. The limit checking can be disabled by programming the respective limit register with 7FF8h for the high limit and 0000h for the low limit. At reset, limit checking is disabled.

22.7.1.18.3 Diagram

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

22.7.1.18.4 Fields

Field	Function
15 —	RESERVED
14-3 HLMT	High Limit Bits
2-0 —	RESERVED

22.7.1.19 ADC Offset Registers (OFFST0 - OFFST15)

22.7.1.19.1 Offset

For a = 0 to 15:

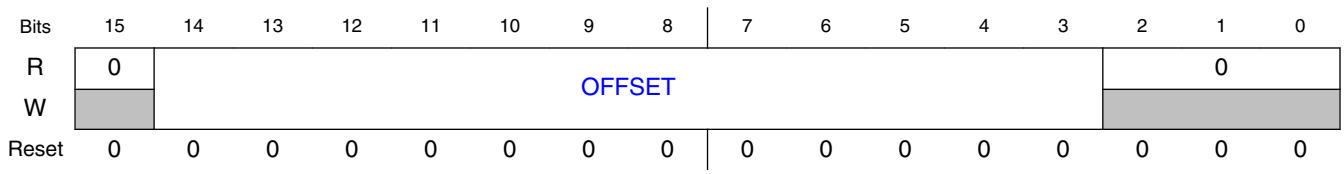
Register	Offset
OFFSTa	3Eh + (a × 1h)

22.7.1.19.2 Function

The value of the offset register is used to correct the ADC result before it is stored in the RSLT registers.

The offset value is subtracted from the ADC result. To obtain unsigned results, program the respective offset register with a value of $0000, thus giving a result range of $0000 to $7FF8.

22.7.1.19.3 Diagram



22.7.1.19.4 Fields

Field	Function
15 —	RESERVED
14-3 OFFSET	ADC Offset Bits
2-0 —	RESERVED

22.7.1.20 ADC Power Control Register (PWR)

22.7.1.20.1 Offset

Register	Offset
PWR	4Eh

22.7.1.20.2 Function

This register controls the power management features of the ADC module. There are individual manual power down controls for the two ADC converters and the voltage reference generators. There are also five distinct power modes. The following terms are used to describe power modes and their related controls.

Power down state	Each converter and voltage reference generator can individually be put into a power down state. When powered down, the unit consumes no power. Results of scans referencing a powered down converter are undefined. At least one converter must be powered up to use the ADC module.
Manual power down controls	Each converter and voltage reference generator have a manual power control bit capable of putting that component into the power down state. Converters have other mechanisms that can automatically put them into the power down state.
Idle state	The ADC module is idle when neither of the two converters has a scan in process.
Active state	The ADC module is active when at least one of the two converters has a scan in process.
Current Mode	Both converters share a common current mode. Normal current mode is used to power the converters at clock rates above 600kHz. Current mode does not affect the number of ADC clock cycles required to do a conversion or the accuracy of a conversion. The ADC module may change the current mode when idle as part of the power saving strategy.
Startup delay	Auto-powerdown power mode causes a startup delay when the ADC module goes between the idle and active states to allow time to switch clocks or power configurations.

22.7.1.20.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	0		1	PSTS1	PSTS0	PUDELAY						APD	1	PD1	PD0
W																
Reset	0	0	0	1	1	1	0	1	1	0	1	0	0	1	1	1

22.7.1.20.4 Fields

Field	Function
15 —	RESERVED

Table continues on the next page...

Field	Function
14-13 —	RESERVED
12 —	RESERVED
11 PSTS1	ADC Converter B Power Status This bit is asserted immediately following a write of "1" to PWR[PD1]. It is de-asserted PWR[PUDELAY] ADC clock cycles after a write of "0" to PWR[PD1] if PWR[APD] is "0". This bit can be read as a status bit to determine when the ADC is ready for operation. During auto-powerdown mode, this bit indicates the current powered state of converter B. **NOTE:** In auto-powerdown mode, this bit is de-asserted right after a scan is started (which means it does not wait PWR[PUDELAY] ADC clock cycles). 0b - ADC Converter B is currently powered up 1b - ADC Converter B is currently powered down
10 PSTS0	ADC Converter A Power Status This bit is asserted immediately following a write of "1" to PWR[PD0]. It is de-asserted PWR[PUDELAY] ADC clock cycles after a write of "0" to PWR[PD0] if PWR[APD] is "0". This bit can be read as a status bit to determine when the ADC is ready for operation. During auto-powerdown mode, this bit indicates the current powered state of converter A. **NOTE:** In auto-powerdown mode, this bit is de-asserted right after a scan is started (which means it does not wait PWR[PUDELAY] ADC clock cycles). 0b - ADC Converter A is currently powered up 1b - ADC Converter A is currently powered down
9-4 PUDELAY	Power Up Delay This 6-bit field determines the number of ADC clocks provided to power up an ADC converter (after setting PWR[PD0 or PD1] to 0) before allowing a scan to start. It also determines the number of ADC clocks of delay provided in auto-powerdown (APD) mode between when the ADC goes from the idle to active state and when the scan is allowed to start. The default value is 13 ADC clocks. Accuracy of the initial conversions in a scan will be degraded if PWR[PUDELAY] is set to a value too small. **NOTE:** PWR[PUDELAY] defaults to a value that is typically sufficient for any power mode. The latency of a scan can be reduced by reducing PWR[PUDELAY] to the lowest value for which accuracy is not degraded. Refer to the data sheet for further details.
3 APD	Auto Powerdown Auto-powerdown mode powers down converters when not in use for a scan. When a scan is started in PWR[APD] mode, a delay of PWR[PUDELAY] ADC clock cycles is imposed during which the needed converter(s), if idle, are powered up. The ADC will then initiate a scan equivalent to that done when PWR[APD] is not active. When the scan is completed, the converter(s) are powered down again. **NOTE:** If PWR[APD] is asserted while a scan is in progress, that scan is unaffected and the ADC will wait to enter its low power state until after all conversions are complete and both ADCs are idle. PWR[APD] are not useful in looping modes. The continuous nature of scanning means that the low power state can never be entered. 0b - Auto Powerdown Mode is not active 1b - Auto Powerdown Mode is active
2 —	RESERVED
1 PD1	Manual Power Down for Converter B This bit forces ADC converter B to power down.

Table continues on the next page...

Field	Function
	Asserting this bit powers down converter B immediately. The results of a scan using converter B will be invalid while PWR[PD1] is asserted. When PWR[PD1] is cleared, converter B is either continuously powered up (PWR[APD] = 0) or automatically powered up when needed (PWR[APD]=1). When clearing this bit in any power mode except auto-powerdown (PWR[APD]=1), wait PWR[PUDELAY] ADC clock cycles before initiating a scan to stabilize power levels within the converter. The PWR[PSTS1] bit can be polled to determine when the PWR[PUDELAY] time has elapsed. Failure to follow this procedure can result in loss of accuracy of the first two samples. 0b - Power Up ADC converter B 1b - Power Down ADC converter B
0 PD0	Manual Power Down for Converter A This bit forces ADC converter A to power down. Asserting this bit powers down converter A immediately. The results of a scan using converter A will be invalid while PWR[PD0] is asserted. When PWR[PD0] is cleared, converter A is either continuously powered up (PWR[APD] = 0) or automatically powered up when needed (PWR[APD]=1). When clearing this bit in any power mode except auto-powerdown (PWR[APD]=1), wait PWR[PUDELAY] ADC clock cycles before initiating a scan to stabilize power levels within the converter. The PWR[PSTS0] bit can be polled to determine when the PWR[PUDELAY] time has elapsed. Failure to follow this procedure can result in loss of accuracy of the first two samples. 0b - Power Up ADC converter A 1b - Power Down ADC converter A

22.7.1.21 ADC Calibration Register (CAL)

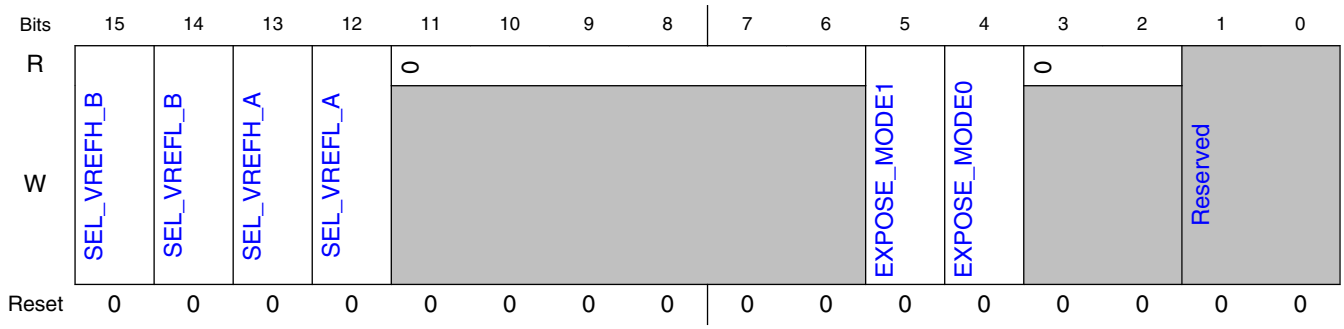
22.7.1.21.1 Offset

Register	Offset
CAL	4Fh

22.7.1.21.2 Function

The ADC provides for off-chip references that can be used for ADC conversions.

22.7.1.21.3 Diagram



22.7.1.21.4 Fields

Field	Function
15 SEL_VREFH_B	Select V REFH Source This bit selects the source of the V $_{REFH}$ reference for all conversions in converter 1. 0b - Internal VDDA 1b - ANB2
14 SEL_VREFL_B	Select V REFLO Source This bit selects the source of the V $_{REFLO}$ reference for all conversions in converter 1. 0b - Internal VSSA 1b - ANB3
13 SEL_VREFH_A	Select V REFH Source This bit selects the source of the V $_{REFH}$ reference for all conversions in converter 0. 0b - Internal VDDA 1b - ANA2
12 SEL_VREFL_A	Select V REFLO Source This bit selects the source of the V $_{REFLO}$ reference for all conversions in converter 0. 0b - Internal VSSA 1b - ANA3
11-6 —	RESERVED
5 EXPOSE_MODE1	Expose chip internal signals to user using ADCB's channel 4,5,6,7. See chip specific content for the description of the internal signals and the channel used in ADCB. 0b - ADCB expose mode is disabled. 1b - ADCB expose mode is enabled.
4 EXPOSE_MODE0	Expose chip internal signals to user using ADCA's channel 4,5,6,7. See chip specific content for the description of the internal signals and the channel used in ADCA. 0b - ADCA expose mode is disabled. 1b - ADCA expose mode is enabled.
3-2 —	RESERVED

Table continues on the next page...

Field	Function
1-0 —	RESERVED

22.7.1.22 Gain Control 1 Register (GC1)

22.7.1.22.1 Offset

Register	Offset
GC1	50h

22.7.1.22.2 Function

The gain control registers are used to control amplification of each of the 16 input channels. GAIN0-GAIN7 control the amplification of inputs ANA0-ANA7 while GAIN8-GAIN15 control the amplification of inputs ANB0-ANB7.

22.7.1.22.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GAIN7		GAIN6		GAIN5		GAIN4		GAIN3		GAIN2		GAIN1		GAIN0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22.7.1.22.4 Fields

Field	Function
15-14 GAIN7	Gain Control Bit 7 GAIN 7 controls ANA7 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved
13-12 GAIN6	Gain Control Bit 6 GAIN 6 controls ANA6 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification

Table continues on the next page...

Field	Function
	11b - reserved
11-10 GAIN5	Gain Control Bit 5 GAIN 5 controls ANA5 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved
9-8 GAIN4	Gain Control Bit 4 GAIN 4 controls ANA4 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved
7-6 GAIN3	Gain Control Bit 3 GAIN 3 controls ANA3 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved
5-4 GAIN2	Gain Control Bit 2 GAIN 2 controls ANA2 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved
3-2 GAIN1	Gain Control Bit 1 GAIN 1 controls ANA1 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved
1-0 GAIN0	Gain Control Bit 0 GAIN 0 controls ANA0 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved

22.7.1.23 Gain Control 2 Register (GC2)

22.7.1.23.1 Offset

Register	Offset
GC2	51h

22.7.1.23.2 Function

The gain control registers are used to control amplification of each of the 16 input channels. GAIN0-GAIN7 control the amplification of inputs ANA0-ANA7 while GAIN8-GAIN15 control the amplification of inputs ANB0-ANB7.

22.7.1.23.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GAIN15		GAIN14		GAIN13		GAIN12		GAIN11		GAIN10		GAIN9		GAIN8	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22.7.1.23.4 Fields

Field	Function
15-14 GAIN15	Gain Control Bit 15 GAIN 15 controls ANB7 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved
13-12 GAIN14	Gain Control Bit 14 GAIN 14 controls ANB6 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved
11-10 GAIN13	Gain Control Bit 13 GAIN 13 controls ANB5 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved
9-8 GAIN12	Gain Control Bit 12 GAIN 12 controls ANB4 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved
7-6 GAIN11	Gain Control Bit 11 GAIN 11 controls ANB3 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification

Table continues on the next page...

Field	Function
	11b - reserved
5-4 GAIN10	Gain Control Bit 10 GAIN 10 controls ANB2 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved
3-2 GAIN9	Gain Control Bit 9 GAIN 9 controls ANB1 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved
1-0 GAIN8	Gain Control Bit 8 GAIN 8 controls ANB0 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved

22.7.1.24 ADC Scan Control Register (SCTRL)

22.7.1.24.1 Offset

Register	Offset
SCTRL	52h

22.7.1.24.2 Function

This register is an extension to the CLIST1-4 registers, providing the ability to pause and await a new sync while processing samples programmed in the CLISTn[SAMPLE0–SAMPLE15] fields.

These 16 control bits are used to determine whether a sample in a scan occurs immediately or if the sample waits for an enabled sync input to occur. The sync input must occur after the conversion of the current sample completes. During sequential mode scans, the SCTRL[SC] bits are used in order from SC0 to SC15. During simultaneous parallel scan modes, the bits are used in order from SC0 to SC3 and SC8 to SC11. In non-simultaneous parallel scans, ADCA uses the bits in order from SC0 to SC3 followed by SC8 to SC11. ADCB will use bits SC4 to SC7 followed by SC12 to SC15 in non-simultaneous parallel scans.

When setting SCTRL[SC0], do not set CTRL1[START0] or CTRL2[START1]. Just clear CTRL1[STOP0] or CTRL2[STOP1] and the first enabled sync input will start the scan.

Setting SC0 delays sample 0 until a sync pulse occurs. Setting SC1 delays sample 1 until a sync pulse occurs after completing sample 0.

NOTE

For parallel scan modes setting, SIM_MISC0[3] will cause ADCA to use bits SC0 to SC7 for simultaneous and non-simultaneous scans and ADCB to use bits SC8 to SC15 for non-simultaneous scans.

22.7.1.24.3 Diagram

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

22.7.1.24.4 Fields

Field	Function
15-0 SC	Scan Control Bits 0000_0000_0000_0000b - Bit value 0: performs sample immediately after the completion of the current sample. 0000_0000_0000_0001b - Bit value 1: delays sample until a new sync input occurs.

22.7.1.25 ADC Power Control Register 2 (PWR2)

22.7.1.25.1 Offset

Register	Offset
PWR2	53h

22.7.1.25.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		DIV1						0				0		0	
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

22.7.1.25.3 Fields

Field	Function
15-14 —	RESERVED
13-8 DIV1	Clock Divisor Select The divider circuit operates in the same manner as the CTRL2[DIV0] field but is used to generate the clock used by ADCB during parallel non-simultaneous scan modes.
7-4 —	RESERVED
3-2 —	RESERVED
1-0 —	RESERVED

22.7.1.26 ADC Control Register 3 (CTRL3)

22.7.1.26.1 Offset

Register	Offset
CTRL3	54h

22.7.1.26.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UPDEN_H				UPDEN_L				0	DMASRC	0			0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22.7.1.26.3 Fields

Field	Function
15-12 UPDEN_H	Unipolar Differential Enable High bits The bits configure differential conversions for either unipolar or fully differential mode. These bits only apply to analog input pairs configured for differential measurements using the CTRL2[CHNCFG_H] bits. Refer to the CTRL2[CHNCFG_H] bit's description for details. 0xxxb - Inputs = ANB6-ANB7 : Fully differential mode enabled on ANB6-ANB7 1xxxb - Inputs = ANB6-ANB7 : Unipolar differential mode enabled on ANB6-ANB7 x0xxb - Inputs = ANB4-ANB5 : Fully differential mode enabled on ANB4-ANB5 x1xxb - Inputs = ANB4-ANB5 : Unipolar differential mode enabled on ANB4-ANB5 xx0xb - Inputs = ANA6-ANA7 : Fully differential mode enabled on ANA6-ANA7 xx1xb - Inputs = ANA6-ANA7 : Unipolar differential mode enabled on ANA6-ANA7 xxx0b - Inputs = ANA4-ANA5 : Fully differential mode enabled on ANA4-ANA5 xxx1b - Inputs = ANA4-ANA5 : Unipolar differential mode enabled on ANA4-ANA5
11-8 UPDEN_L	Unipolar Differential Enable Low bits The bits configure differential conversions for either unipolar or fully differential mode. These bits only apply to analog input pairs configured for differential measurements using the CTRL2[CHNCFG_H] bits. Refer to the CTRL1[CHNCFG_L] bit's description for details. 0xxxb - Inputs = ANB2-ANB3 : Fully differential mode enabled on ANB2-ANB3 1xxxb - Inputs = ANB2-ANB3 : Unipolar differential mode enabled on ANB2-ANB3 x0xxb - Inputs = ANB0-ANB1 : Fully differential mode enabled on ANB0-ANB1 x1xxb - Inputs = ANB0-ANB1 : Unipolar differential mode enabled on ANB0-ANB1 xx0xb - Inputs = ANA2-ANA3 : Fully differential mode enabled on ANA2-ANA3 xx1xb - Inputs = ANA2-ANA3 : Unipolar differential mode enabled on ANA2-ANA3 xxx0b - Inputs = ANA0-ANA1 : Fully differential mode enabled on ANA0-ANA1 xxx1b - Inputs = ANA0-ANA1 : Unipolar differential mode enabled on ANA0-ANA1
7 —	RESERVED
6 DMASRC	DMA Trigger Source During sequential and simultaneous parallel scan modes CTRL3[DMASRC] selects between EOSI0 and RDY bits as the DMA source. During non-simultaneous parallel scan mode CTRL3[DMASRC] selects between EOSI0/EOSI1 for converters A and B, and the RDY bits as the DMA source. 0b - DMA trigger source is end of scan interrupt 1b - DMA trigger source is RDY bits
5-3 —	RESERVED
2-0 —	RESERVED

22.7.1.27 ADC Scan Interrupt Enable Register (SCHLTEN)

22.7.1.27.1 Offset

Register	Offset
SCHLTEN	55h

22.7.1.27.2 Function

This register is used with ready register (RDY) to select the samples that will generate a scan interrupt .

22.7.1.27.3 Diagram

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

22.7.1.27.4 Fields

Field	Function
15-0 SCHLTEN	SCHLTEN Scan Interrupt Enable 0000_0000_0000_0000b - Bit value 0: scan interrupt is not enabled for this sample. 0000_0000_0000_0001b - Bit value 1: scan interrupt is enabled for this sample.

22.7.1.28 ADC Zero Crossing Control 3 Register (ZXCTRL3)

22.7.1.28.1 Offset

Register	Offset
ZXCTRL3	58h

22.7.1.28.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								ZCE19		ZCE18		ZCE17		ZCE16	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22.7.1.28.3 Fields

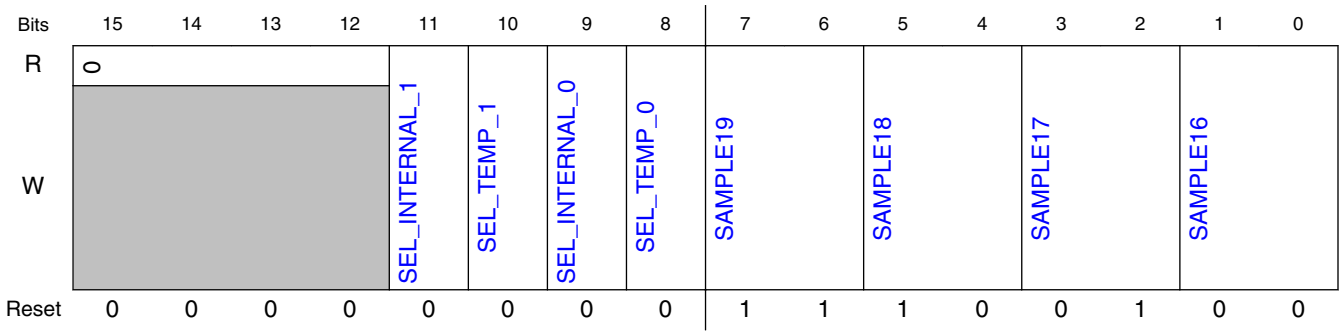
Field	Function
15-8 —	RESERVED
7-6 ZCE19	Zero crossing enable 19 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change
5-4 ZCE18	Zero crossing enable 18 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change
3-2 ZCE17	Zero crossing enable 17 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change
1-0 ZCE16	Zero crossing enable 16 00b - Zero Crossing disabled 01b - Zero Crossing enabled for positive to negative sign change 10b - Zero Crossing enabled for negative to positive sign change 11b - Zero Crossing enabled for any sign change

22.7.1.29 ADC Channel List Register 5 (CLIST5)

22.7.1.29.1 Offset

Register	Offset
CLIST5	59h

22.7.1.29.2 Diagram



22.7.1.29.3 Fields

Field	Function
15-12 —	RESERVED
11 SEL_INTERNAL_1	Select On-Chip Analog Input Alternate Source This bit selects the source of the ANB7 input as being either the input pin ANB7 or ADCB on-chip analog signal. 0b - Normal operation (ANB7) 1b - ANB7 input is replaced with ADCB on-chip analog signal
10 SEL_TEMP_1	Select Temperature Sensor Alternate Source This bit selects the source of the ADCB6 input as being either the input pin ADCB6 or ADCB temperature sensors. 0b - Normal Operation (ADCB6) 1b - ADCB6 input is replaced with ADCB temperature sensor
9 SEL_INTERNAL_0	Select On-Chip Analog Input Alternate Source This bit selects the source of the ANA7 input as being either the input pin ANA7 or ADCA on-chip analog signal. 0b - Normal Operation (ANA7) 1b - ANA7 input is replaced with ADCA on-chip analog signal
8 SEL_TEMP_0	Select Temperature Sensor Alternate Source This bit selects the source of the ADCA6 input as being either the input pin ADCA6 or ADCA temperature sensors. 0b - Normal Operation (ADCA6) 1b - ADCA6 input is replaced with ADCA temperature sensor
7-6 SAMPLE19	Sample Field 19 This sample and all subsequent samples can be disabled by setting ADC_SDIS2[3]. 00b - single-ended: ADCA temperature sensor 01b - single-ended: ADCA analog input for on-chip generated signals 10b - single-ended: ADCB temperature sensor 11b - single-ended: ADCB analog input for on-chip generated signals
5-4 SAMPLE18	Sample Field 18 This sample and all subsequent samples can be disabled by setting ADC_SDIS2[2].

Table continues on the next page...

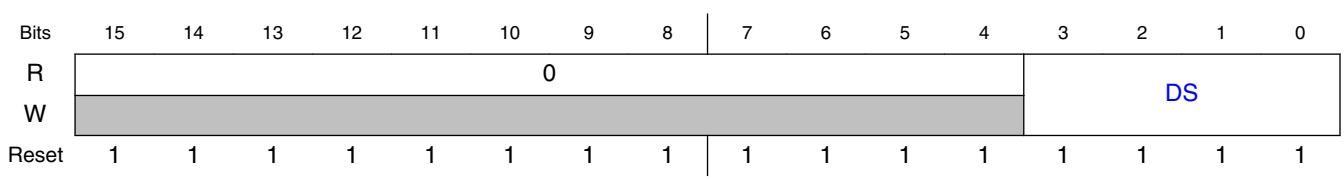
Field	Function
	00b - single-ended: ADCA temperature sensor 01b - single-ended: ADCA analog input for on-chip generated signals 10b - single-ended: ADCB temperature sensor 11b - single-ended: ADC B analog input for on-chip generated signals
3-2 SAMPLE17	Sample Field 17 This sample and all subsequent samples can be disabled by setting ADC_SDIS2[1]. 00b - single-ended: ADCA temperature sensor 01b - single-ended: ADCA analog input for on-chip generated signals 10b - single-ended: ADCB temperature sensor 11b - single-ended: ADCB analog input for on-chip generated signals
1-0 SAMPLE16	Sample Field 16 This sample and all subsequent samples can be disabled by setting ADC_SDIS2[0]. 00b - single-ended: ADCA temperature sensor 01b - single-ended: ADCA analog input for on-chip generated signals 10b - single-ended: ADCB temperature sensor 11b - single-ended: ADCB analog input for on-chip generated signals

22.7.1.30 ADC Sample Disable Register 2 (SDIS2)

22.7.1.30.1 Offset

Register	Offset
SDIS2	5Ah

22.7.1.30.2 Diagram



22.7.1.30.3 Fields

Field	Function
15-4 —	RESERVED
3-0 DS	Disable Sample Bits 0000b - SAMPLEx channel is enabled for ADC scan.

Field	Function
	0001b - SAMPLEx channel is disabled for ADC scan and corresponding channels after SAMPLEx will also not occur in an ADC scan. Enabling the four extra sample slots by themselves (ADC_SDIS=FFFF, and samples enabled in ADC_SDIS2) is supported only in once sequential mode. Sequential loop, and parallel (both sequential and loop) modes are not supported. It is suggested to poll the ADC_RDY2 register to check for conversion completion for this case.

22.7.1.31 ADC Ready Register 2 (RDY2)

22.7.1.31.1 Offset

Register	Offset
RDY2	5Bh

22.7.1.31.2 Function

This register provides the current status of the ADC conversions. RDY2[RDYx] bits are cleared by reading their corresponding result registers (RSLTx).

22.7.1.31.3 Diagram

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

22.7.1.31.4 Fields

Field	Function
15-4 —	RESERVED
3-0 RDY	Ready Sample These bits indicate samples nineteen through sixteen are ready to be read. These bits are cleared after a read from the respective results register. The RDY2[RDYn] bits are set as the individual channel conversions are completed. Polling the RDY2[RDYn] bits can determine if a particular sample is ready to be read. 0000b - Sample not ready or has been read 0001b - Sample ready to be read

22.7.1.32 ADC Low Limit Status Register 2 (LOLIMSTAT2)

22.7.1.32.1 Offset

Register	Offset
LOLIMSTAT2	5Ch

22.7.1.32.2 Function

The low limit status register 2 provides the same function as the low limit status register (LOLIMSTAT2) for SAMPLEs 16-19.

22.7.1.32.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												LLS			
W													W1C			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22.7.1.32.4 Fields

Field	Function
15-4 —	RESERVED
3-0 LLS	Low Limit Status Bits

22.7.1.33 ADC High Limit Status Register 2 (HILIMSTAT2)

22.7.1.33.1 Offset

Register	Offset
HILIMSTAT2	5Dh

22.7.1.33.2 Function

The high limit status register 2 provides the same function as the high limit status register (HILIMSTAT) for SAMPLEs 16-19.

22.7.1.33.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												HLS			
W													W1C			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22.7.1.33.4 Fields

Field	Function
15-4 —	RESERVED
3-0 HLS	High Limit Status Bits

22.7.1.34 ADC Zero Crossing Status Register 2 (ZXSTAT2)

22.7.1.34.1 Offset

Register	Offset
ZXSTAT2	5Eh

22.7.1.34.2 Function

The zero crossing status register 2 provides the same function as the zero crossing status register (ZXSTAT) for SAMPLEs 16-19.

22.7.1.34.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												ZCS			
W													W1C			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22.7.1.34.4 Fields

Field	Function
15-4 —	RESERVED
3-0 ZCS	Zero Crossing Status The zero crossing condition is determined by examining the ADC value after it has been adjusted by the offset for the result register. Each bit of the register is cleared by writing a one to that register bit. 0000b - Either: A sign change did not occur in a comparison between the current channel[x] result and the previous channel[x] result, or Zero crossing control is disabled for channel[x] in the zero crossing control register (ZXCTRL3) 0001b - In a comparison between the current channel[x] result and the previous channel[x] result, a sign change condition occurred as defined in the zero crossing control register (ZXCTRL3)

22.7.1.35 ADC Result Registers 2 with sign extension (RSLT216 - RSLT219)

22.7.1.35.1 Offset

Register	Offset
RSLT216	5Fh
RSLT217	60h
RSLT218	61h
RSLT219	62h

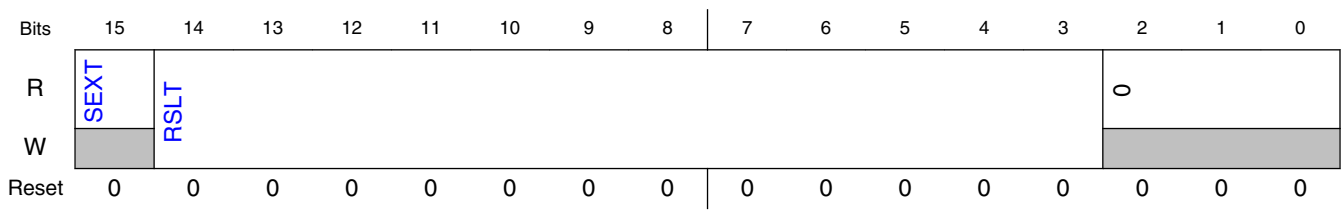
22.7.1.35.2 Function

The result registers contain the converted results from a scan. The CLIST5[SAMPLE16] result is loaded into RSLT16, CLIST5[SAMPLE17] result in RSLT17, and so on. In a parallel scan mode, the first channel pair designated by CLIST5[SAMPLE16] and CLIST5[SAMPLE18] are stored in RSLT16 and RSLT18, respectively.

Note

When writing to this register, only the RSLT portion of the value written is used. This value is modified and the result of the subtraction is stored. The SEXT bit is only set as a result of this subtraction and is not directly determined by the value written.

22.7.1.35.3 Diagram



22.7.1.35.4 Fields

Field	Function
15 SEXT	Sign Extend This is the sign-extend bit of the result. RSLT*[SEXT] set to one implies a negative result. RSLT*[SEXT] set to zero implies a positive result. If positive results are required, then the respective offset register must be set to a value of zero.
14-3 RSLT	Digital Result of the Conversion RSLT can be interpreted as either a signed integer or a signed fractional number. As a signed fractional number, the RSLT can be used directly. As a signed integer, it is an option to right shift with sign extend (ASR) three places and interpret the number, or accept the number as presented, knowing there are missing codes. The lower three bits are always going to be zero. Negative results, RSLT*[SEXT] = 1, are always presented in two's complement format. If it is a requirement of your application that the result registers always be positive, the offset registers must always be set to zero. The interpretation of the numbers programmed into the limit and offset registers, LOLIM2, HILIM2, and OFFST2 should match your interpretation of the result register.
2-0 —	RESERVED

22.7.1.36 ADC Low Limit Registers 2 (LOLIM216 - LOLIM219)

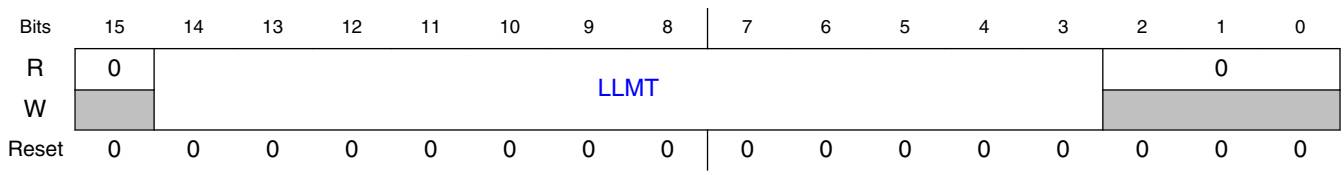
22.7.1.36.1 Offset

Register	Offset
LOLIM216	63h
LOLIM217	64h
LOLIM218	65h
LOLIM219	66h

22.7.1.36.2 Function

The low limit registers 2 provide the same function as the low limit registers (LOLIM) for SAMPLEs 16-19.

22.7.1.36.3 Diagram



22.7.1.36.4 Fields

Field	Function
15 —	RESERVED
14-3 LLMT	Low Limit Bits
2-0 —	RESERVED

22.7.1.37 ADC High Limit Registers 2 (HILIM216 - HILIM219)

22.7.1.37.1 Offset

Register	Offset
HILIM216	67h

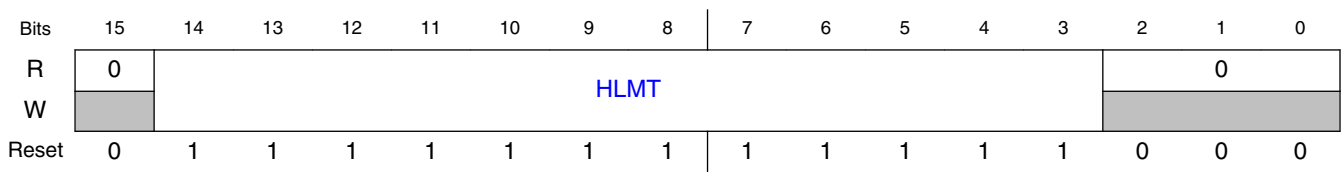
Table continues on the next page...

Register	Offset
HILIM217	68h
HILIM218	69h
HILIM219	6Ah

22.7.1.37.2 Function

The hight limit registers 2 provide the same function as the high limit registers (HILIM) for SAMPLEs 16-19.

22.7.1.37.3 Diagram



22.7.1.37.4 Fields

Field	Function
15 —	RESERVED
14-3 HLMT	High Limit Bits
2-0 —	RESERVED

22.7.1.38 ADC Offset Registers 2 (OFFST216 - OFFST219)

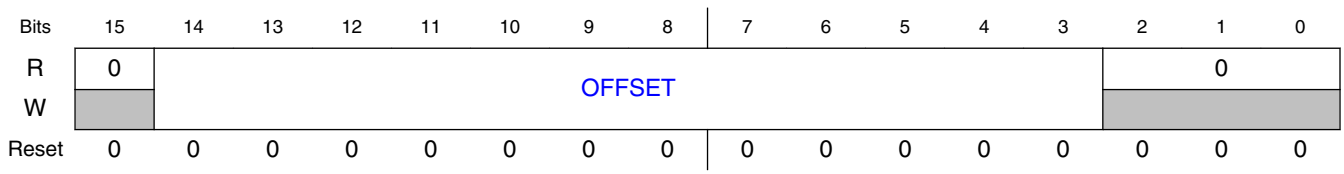
22.7.1.38.1 Offset

Register	Offset
OFFST216	6Bh
OFFST217	6Ch
OFFST218	6Dh
OFFST219	6Eh

22.7.1.38.2 Function

The offset registers 2 provide the same function as the offset registers (OFFST) for SAMPLEs 16-19.

22.7.1.38.3 Diagram



22.7.1.38.4 Fields

Field	Function
15 —	RESERVED
14-3 OFFSET	ADC Offset Bits
2-0 —	RESERVED

22.7.1.39 Gain Control 3 Register (GC3)

22.7.1.39.1 Offset

Register	Offset
GC3	6Fh

22.7.1.39.2 Function

The gain control registers are used to control amplification of each of the 4 analog inputs for on-chip generated signals. GAIN16-GAIN17 control the amplification of the ADCA temperature sensor and analog input for on-chip analog generated signals, while GAIN18-GAIN19 control the amplification of the ADCB temperature sensor and analog input for on-chip generated analog signals.

22.7.1.39.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								GAIN19		GAIN18		GAIN17		GAIN16	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22.7.1.39.4 Fields

Field	Function
15-8 —	RESERVED
7-6 GAIN19	Gain Control Bit 19 GAIN 19 controls ADCB analog input for on-chip generated signals 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved
5-4 GAIN18	Gain Control Bit 18 GAIN 18 ADCB temperature sensor 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved
3-2 GAIN17	Gain Control Bit 17 GAIN 17 controls ADCA analog input for on-chip generated signals 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved
1-0 GAIN16	Gain Control Bit 16 GAIN 16 controls ADCA temperature sensor 00b - x1 amplification 01b - x2 amplification 10b - x4 amplification 11b - reserved

22.7.1.40 ADC Scan Control Register 2 (SCTRL2)

22.7.1.40.1 Offset

Register	Offset
SCTRL2	70h

22.7.1.40.2 Function

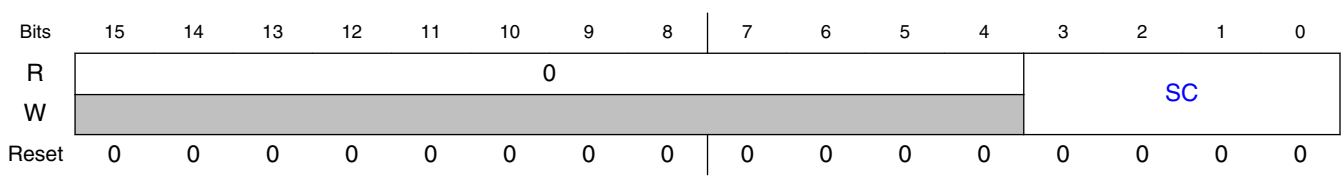
This register is an extension to the CLIST5 register, providing the ability to pause and await a new sync while processing samples programmed in the CLIST5[SAMPLE16–SAMPLE19] fields.

These 4 control bits are used to determine whether a sample in a scan occurs immediately or if the sample waits for an enabled sync input to occur. The sync input must occur after the conversion of the current sample completes. During sequential mode scans, the SCTRL2[SC] bits are used in order from SC16 to SC19. In parallel scan modes, the bits are used in order from SC16 to SC17 and SC18-SC19.

When setting SCTRL2[SC16], do not set CTRL1[START0] or CTRL2[START1]. Just clear CTRL1[STOP0] or CTRL2[STOP1] and the first enabled sync input will start the scan.

Setting SC16 delays sample 16 until a sync pulse occurs. Setting SC17 delays sample 17 until a sync pulse occurs after completing sample 16.

22.7.1.40.3 Diagram



22.7.1.40.4 Fields

Field	Function
15-4 —	RESERVED
3-0 SC	Scan Control Bits 0000b - Bit value 0: performs sample immediately after the completion of the current sample. 0001b - Bit value 1: delays sample until a new sync input occurs.

22.7.1.41 ADC Scan Interrupt Enable Register 2 (SCHLTEN2)

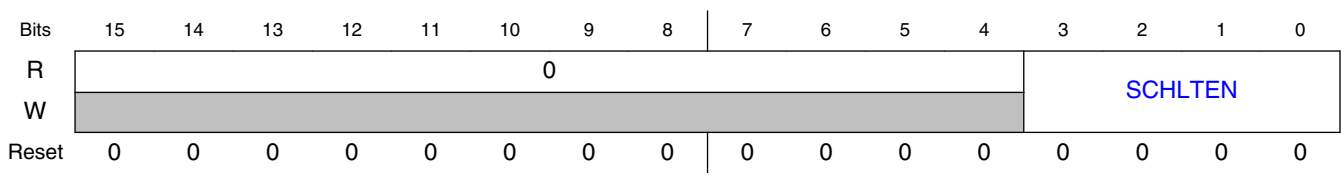
22.7.1.41.1 Offset

Register	Offset
SCHLTEN2	71h

22.7.1.41.2 Function

This register is used with the ready register 2 (RDY2) to select the samples that will generate a scan interrupt .

22.7.1.41.3 Diagram



22.7.1.41.4 Fields

Field	Function
15-4 —	RESERVED
3-0 SCHLTEN	SCHLTEN Scan Interrupt Enable 0000b - Bit value 0: scan interrupt is not enabled for this sample. 0001b - Bit value 1: scan interrupt is enabled for this sample.

22.7.1.42 Expansion AUX Status Register (EXPSTAT)

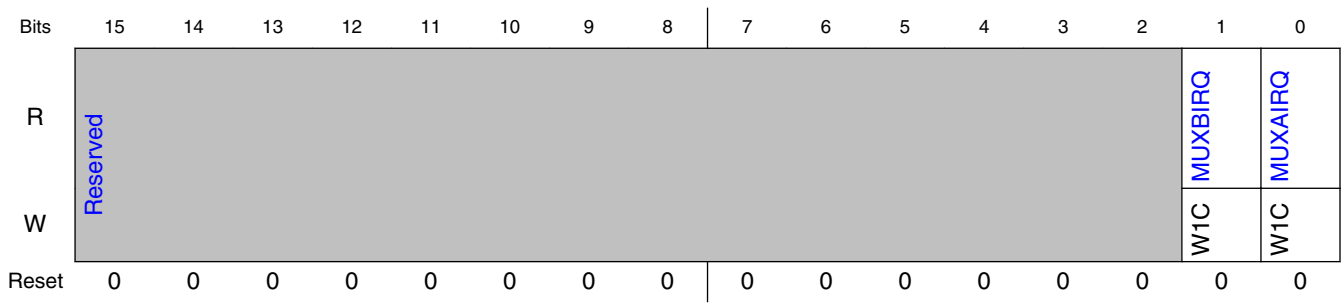
22.7.1.42.1 Offset

Register	Offset
EXPSTAT	78h

22.7.1.42.2 Function

This register provides the current status whether an expansion AUX scan is completed and an interrupt request (IRQ) is pending. If the subsequent expansion AUX scan is terminated when the AUX Scan Disable in EXPMUX4x0 or EXPMUX4x1 registers is encountered, or the last AUX channel in sequential scan determined by EXPAUX4x[AUXSEL7] is served, the corresponding bit is asserted.

22.7.1.42.3 Diagram



22.7.1.42.4 Fields

Field	Function
15-2 —	Reserved
1 MUXBIRQ	ANB4 Expansion AUX Channel Scan Complete Interrupt Request This bit indicates whether the channel scan of ANB4 Expansion AUX is terminated or completed. This bit is cleared by writing a logic 1 to it, and cannot be set by software. 0b - A scan cycle is not completed, and no IRQ is pending. 1b - A scan cycle is completed, and an IRQ is pending.
0 MUXAIRQ	ANA4 Expansion AUX Channel Scan Complete Interrupt Request This bit indicates whether the channel scan of ANA4 Expansion AUX is terminated or completed. This bit is cleared by writing a logic 1 to it, and cannot be set by software. 0b - A scan cycle is not completed, and no IRQ is pending. 1b - A scan cycle is completed, and an IRQ is pending.

22.7.1.43 Expansion AUX Config Register (EXPCFG)

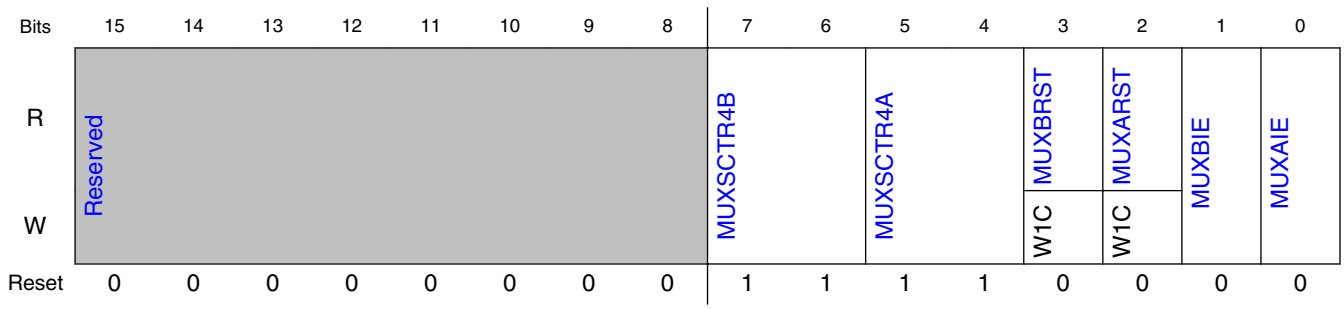
22.7.1.43.1 Offset

Register	Offset
EXPCFG	79h

22.7.1.43.2 Function

This register configures the operations of Auxiliary Controls (AUXCTRL).

22.7.1.43.3 Diagram



22.7.1.43.4 Fields

Field	Function
15-8 —	Reserved
7-6 MUXSCTR4B	ANB4 Expansion AUX Channel Scan Control This field determines the source of ADCB channels that are used to switch the channel of ANB4 Expansion AUX to subsequent channel after current channel is sampled by ADC. 00b - Manual select: The EXPAUX4B[AUXSEL0] bits control output signals AUXB_SEL0 and AUXB_SEL1. 01b - Scan mode 0: The sample completion of ANB4 enables subsequent selected channel. 10b - Scan mode 1: The sample completion of ANB7 enables subsequent selected channel. 11b - Scan mode 2: The sample completion of either ANB4 or ANB7 enables subsequent selected channel.
5-4 MUXSCTR4A	ANA4 Expansion AUX Channel Scan Control This field determines the source of ADCA channels that are used to switch the channel of ANA4 Expansion AUX to subsequent channel after current channel is sampled by ADC. 00b - Manual select: The EXPAUX4A[AUXSEL0] bits control output signals AUXA_SEL0 and AUXA_SEL1. 01b - Scan mode 0: The sample completion of ANA4 enables subsequent selected channel. 10b - Scan mode 1: The sample completion of ANA7 enables subsequent selected channel. 11b - Scan mode 2: The sample completion of either ANA4 or ANA7 enables subsequent selected channel.
3	ANB4 Expansion AUX Scan Reset

Table continues on the next page...

Field	Function
MUXBRST	Writing a logic 1 to this bit resets the Auxiliary control to EXPAUX4B[AUXSEL0]. 0b - Does not reset. 1b - Resets.
2 MUXARST	ANA4 Expansion AUX Scan Reset Writing a logic 1 to this bit resets the Auxiliary control to EXPAUX4A[AUXSEL0]. 0b - Does not reset. 1b - Resets.
1 MUXBIE	ANB4 Expansion AUX Channel Scan Complete Interrupt Enable This bit enables the Expansion AUX Channel Scan Complete Interrupt. 0b - Interrupt disabled 1b - Interrupt enabled.
0 MUXAIE	ANA4 Expansion AUX Channel Scan Complete Interrupt Enable This bit enables the Expansion AUX Channel Scan Complete Interrupt. 0b - Interrupt disabled. 1b - Interrupt enabled.

22.7.1.44 ANA4 Expansion Auxiliary Control Register (EXPAUX4A)

22.7.1.44.1 Offset

Register	Offset
EXPAUX4A	7Ah

22.7.1.44.2 Function

AUXSELx field manages output signals AUXA_SEL0 and AUXA_SEL1 (see the figure "Expansion MUX") to other modules, such as configuration settings of the OPAMP module or the Comparator input AUX or the off-chip external AUX, etc.

22.7.1.44.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	AUXSEL7		AUXSEL6		AUXSEL5		AUXSEL4		AUXSEL3		AUXSEL2		AUXSEL1		AUXSEL0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22.7.1.44.4 Fields

Field	Function
15-14 AUXSEL7	Auxiliary Select 7 controls AUXA_SEL0 and AUXA_SEL1 00b - AUXA_SEL1 = 0; AUXA_SEL0 = 0. 01b - AUXA_SEL1 = 0; AUXA_SEL0 = 1. 10b - AUXA_SEL1 = 1; AUXA_SEL0 = 0. 11b - AUXA_SEL1 = 1; AUXA_SEL0 = 1.
13-12 AUXSEL6	Auxiliary Select 6 controls AUXA_SEL0 and AUXA_SEL1 00b - AUXA_SEL1 = 0; AUXA_SEL0 = 0. 01b - AUXA_SEL1 = 0; AUXA_SEL0 = 1. 10b - AUXA_SEL1 = 1; AUXA_SEL0 = 0. 11b - AUXA_SEL1 = 1; AUXA_SEL0 = 1.
11-10 AUXSEL5	Auxiliary Select 5 controls AUXA_SEL0 and AUXA_SEL1 00b - AUXA_SEL1 = 0; AUXA_SEL0 = 0. 01b - AUXA_SEL1 = 0; AUXA_SEL0 = 1. 10b - AUXA_SEL1 = 1; AUXA_SEL0 = 0. 11b - AUXA_SEL1 = 1; AUXA_SEL0 = 1.
9-8 AUXSEL4	Auxiliary Select 4 controls AUXA_SEL0 and AUXA_SEL1 00b - AUXA_SEL1 = 0; AUXA_SEL0 = 0. 01b - AUXA_SEL1 = 0; AUXA_SEL0 = 1. 10b - AUXA_SEL1 = 1; AUXA_SEL0 = 0. 11b - AUXA_SEL1 = 1; AUXA_SEL0 = 1.
7-6 AUXSEL3	Auxiliary Select 3 controls AUXA_SEL0 and AUXA_SEL1 00b - AUXA_SEL1 = 0; AUXA_SEL0 = 0. 01b - AUXA_SEL1 = 0; AUXA_SEL0 = 1. 10b - AUXA_SEL1 = 1; AUXA_SEL0 = 0. 11b - AUXA_SEL1 = 1; AUXA_SEL0 = 1.
5-4 AUXSEL2	Auxiliary Select 2 controls AUXA_SEL0 and AUXA_SEL1 00b - AUXA_SEL1 = 0; AUXA_SEL0 = 0. 01b - AUXA_SEL1 = 0; AUXA_SEL0 = 1. 10b - AUXA_SEL1 = 1; AUXA_SEL0 = 0. 11b - AUXA_SEL1 = 1; AUXA_SEL0 = 1.
3-2 AUXSEL1	Auxiliary Select 1 controls AUXA_SEL0 and AUXA_SEL1 00b - AUXA_SEL1 = 0; AUXA_SEL0 = 0. 01b - AUXA_SEL1 = 0; AUXA_SEL0 = 1. 10b - AUXA_SEL1 = 1; AUXA_SEL0 = 0. 11b - AUXA_SEL1 = 1; AUXA_SEL0 = 1.
1-0 AUXSEL0	Auxiliary Select 0 controls AUXA_SEL0 and AUXA_SEL1 00b - AUXA_SEL1 = 0; AUXA_SEL0 = 0. 01b - AUXA_SEL1 = 0; AUXA_SEL0 = 1. 10b - AUXA_SEL1 = 1; AUXA_SEL0 = 0. 11b - AUXA_SEL1 = 1; AUXA_SEL0 = 1.

22.7.1.45 ANB4 Expansion Auxiliary Control Register (EXPAUX4B)

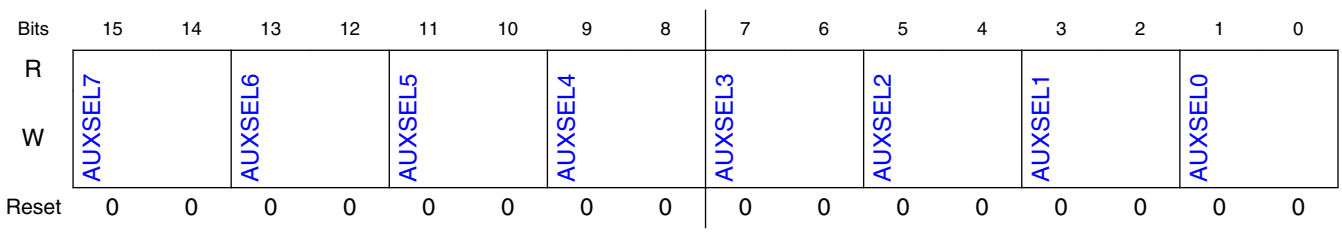
22.7.1.45.1 Offset

Register	Offset
EXPAUX4B	7Bh

22.7.1.45.2 Function

AUXSELx field manages output signals AUXB_SEL0 and AUXB_SEL1 (see the figure "Expansion MUX") to other modules, such as configuration settings of the OPAMP module or the Comparator input AUX or the off-chip external AUX, etc.

22.7.1.45.3 Diagram



22.7.1.45.4 Fields

Field	Function
15-14 AUXSEL7	Auxiliary Select 7 controls AUXB_SEL0 and AUXB_SEL1 00b - AUXB_SEL1 = 0; AUXB_SEL0 = 0. 01b - AUXB_SEL1 = 0; AUXB_SEL0 = 1. 10b - AUXB_SEL1 = 1; AUXB_SEL0 = 0. 11b - AUXB_SEL1 = 1; AUXB_SEL0 = 1.
13-12 AUXSEL6	Auxiliary Select 6 controls AUXB_SEL0 and AUXB_SEL1 00b - AUXB_SEL1 = 0; AUXB_SEL0 = 0. 01b - AUXB_SEL1 = 0; AUXB_SEL0 = 1. 10b - AUXB_SEL1 = 1; AUXB_SEL0 = 0. 11b - AUXB_SEL1 = 1; AUXB_SEL0 = 1.
11-10 AUXSEL5	Auxiliary Select 5 controls AUXB_SEL0 and AUXB_SEL1 00b - AUXB_SEL1 = 0; AUXB_SEL0 = 0. 01b - AUXB_SEL1 = 0; AUXB_SEL0 = 1. 10b - AUXB_SEL1 = 1; AUXB_SEL0 = 0. 11b - AUXB_SEL1 = 1; AUXB_SEL0 = 1.
9-8 AUXSEL4	Auxiliary Select 4 controls AUXB_SEL0 and AUXB_SEL1 00b - AUXB_SEL1 = 0; AUXB_SEL0 = 0. 01b - AUXB_SEL1 = 0; AUXB_SEL0 = 1. 10b - AUXB_SEL1 = 1; AUXB_SEL0 = 0. 11b - AUXB_SEL1 = 1; AUXB_SEL0 = 1.
7-6	Auxiliary Select 3 controls AUXB_SEL0 and AUXB_SEL1 00b - AUXB_SEL1 = 0; AUXB_SEL0 = 0.

Table continues on the next page...

Field	Function
AUXSEL3	01b - AUXB_SEL1 = 0; AUXB_SEL0 = 1. 10b - AUXB_SEL1 = 1; AUXB_SEL0 = 0. 11b - AUXB_SEL1 = 1; AUXB_SEL0 = 1.
5-4 AUXSEL2	Auxiliary Select 2 controls AUXB_SEL0 and AUXB_SEL1 00b - AUXB_SEL1 = 0; AUXB_SEL0 = 0. 01b - AUXB_SEL1 = 0; AUXB_SEL0 = 1. 10b - AUXB_SEL1 = 1; AUXB_SEL0 = 0. 11b - AUXB_SEL1 = 1; AUXB_SEL0 = 1.
3-2 AUXSEL1	Auxiliary Select 1 controls AUXB_SEL0 and AUXB_SEL1 00b - AUXB_SEL1 = 0; AUXB_SEL0 = 0. 01b - AUXB_SEL1 = 0; AUXB_SEL0 = 1. 10b - AUXB_SEL1 = 1; AUXB_SEL0 = 0. 11b - AUXB_SEL1 = 1; AUXB_SEL0 = 1.
1-0 AUXSEL0	Auxiliary Select 0 controls AUXB_SEL0 and AUXB_SEL1 00b - AUXB_SEL1 = 0; AUXB_SEL0 = 0. 01b - AUXB_SEL1 = 0; AUXB_SEL0 = 1. 10b - AUXB_SEL1 = 1; AUXB_SEL0 = 0. 11b - AUXB_SEL1 = 1; AUXB_SEL0 = 1.

22.7.1.46 ANA4 Expansion MUX Control Register 0 (EXPMUX4A0)

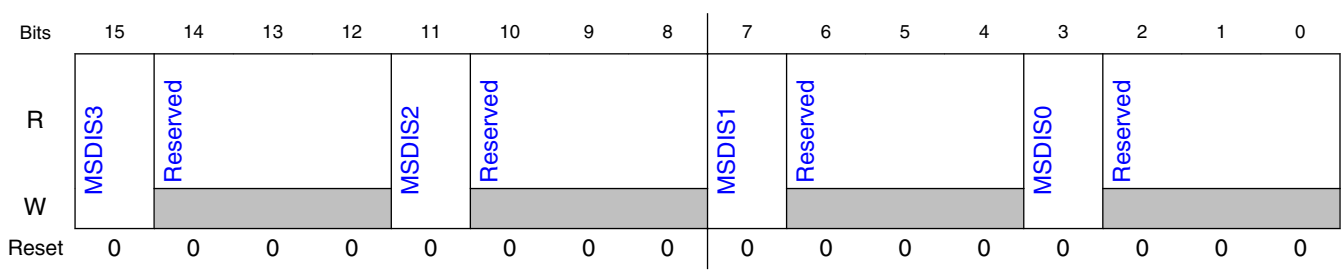
22.7.1.46.1 Offset

Register	Offset
EXPMUX4A0	7Ch

22.7.1.46.2 Function

The MUX Select Disable (MSDIS) bit is used to stop the scan. When a scan is stopped, the scan sequence is reset to EXPAUX4A[AUXSEL0].

22.7.1.46.3 Diagram



22.7.1.46.4 Fields

Field	Function
15 MSDIS3	AUX Select Disable 0b - Expansion AUX scan continues. 1b - Expansion AUX scan stops AUXSEL3 are served, and the scan sequence resets to EXPAUX4A[AUXSEL0].
14-12 —	Reserved
11 MSDIS2	AUX Select Disable 0b - Expansion AUX scan continues. 1b - Expansion AUX scan stops after AUXSEL2 are served, and the scan sequence resets to EXPAUX4A[AUXSEL0].
10-8 —	Reserved
7 MSDIS1	AUX Select Disable 0b - Expansion AUX scan continues. 1b - Expansion AUX scan stops after AUXSEL1 are served, and the scan sequence resets to EXPAUX4A[AUXSEL0].
6-4 —	Reserved
3 MSDIS0	AUX Select Disable 0b - Expansion AUX scan continues. 1b - Expansion AUX scan stops after AUXSEL0 are served, and the scan sequence resets to EXPAUX4A[AUXSEL0].
2-0 —	Reserved

22.7.1.47 ANA4 Expansion MUX Control Register 1 (EXPMUX4A1)

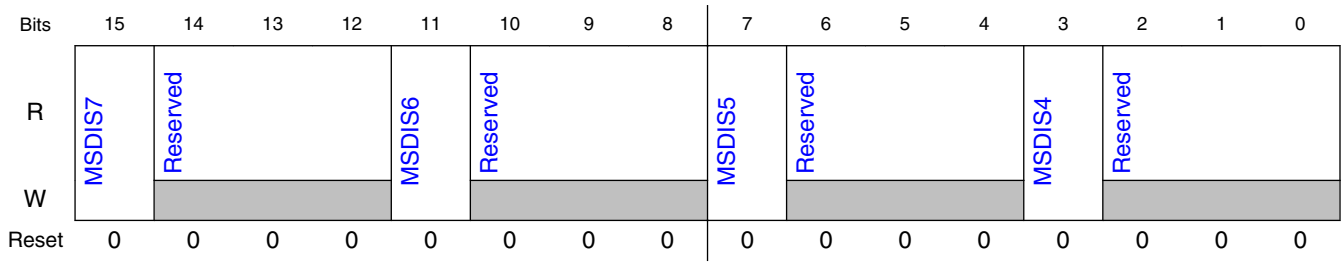
22.7.1.47.1 Offset

Register	Offset
EXPMUX4A1	7Dh

22.7.1.47.2 Function

The AUX Select Disable (MSDIS) bit is used to stop the scan. When a scan is stopped, the scan sequence is reset to EXPAUX4A[AUXSEL0].

22.7.1.47.3 Diagram



22.7.1.47.4 Fields

Field	Function
15 MSDIS7	AUX Select Disable No matter the bit value is 0 or 1, Expansion AUX scan stops after AUXSEL7 is served, and the scan sequence resets to EXPAUX4A[AUXSEL0].
14-12 —	Reserved
11 MSDIS6	AUX Select Disable 0b - Expansion AUX scan continues. 1b - Expansion AUX scan stops after AUXSEL6 are served, and the scan sequence resets to EXPAUX4A[AUXSEL0].
10-8 —	Reserved
7 MSDIS5	AUX Select Disable 0b - Expansion AUX scan continues. 1b - Expansion AUX scan stops after AUXSEL5 are served, and the scan sequence resets to EXPAUX4A[AUXSEL0].
6-4 —	Reserved
3 MSDIS4	AUX Select Disable 0b - Expansion AUX scan continues. 1b - Expansion AUX scan stops after AUXSEL4 are served, and the scan sequence resets to EXPAUX4A[AUXSEL0].
2-0 —	Reserved

22.7.1.48 ANB4 Expansion MUX Control Register 0 (EXPMUX4B0)

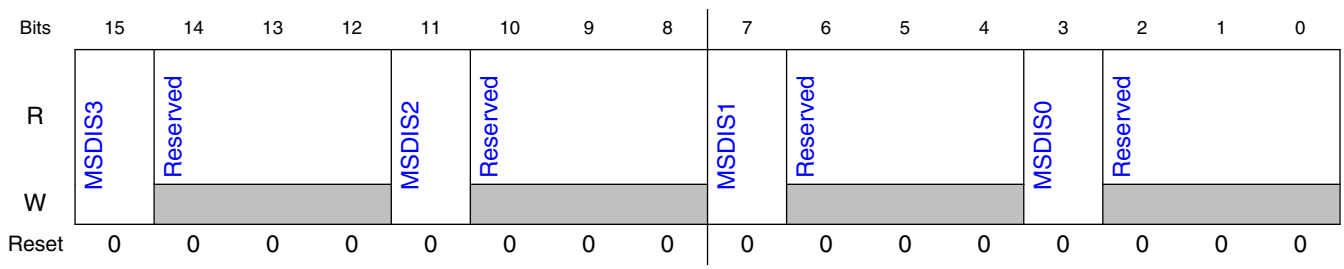
22.7.1.48.1 Offset

Register	Offset
EXPMUX4B0	7Eh

22.7.1.48.2 Function

The AUX Select Disable (MSDIS) bit is used to stop the scan. When a scan is stopped, the scan sequence is reset to EXPAUX4B[AUXSEL0].

22.7.1.48.3 Diagram



22.7.1.48.4 Fields

Field	Function
15 MSDIS3	AUX Select Disable 0b - Expansion AUX scan continues. 1b - Expansion AUX scan stops after AUXSEL3 are served, and the scan sequence resets to EXPAUX4B[AUXSEL0].
14-12 —	Reserved
11 MSDIS2	AUX Select Disable 0b - Expansion AUX scan continues. 1b - Expansion AUX scan stops after AUXSEL2 are served, and the scan sequence resets and EXPAUX4B[AUXSEL0].
10-8 —	Reserved
7 MSDIS1	AUX Select Disable 0b - Expansion AUX scan continues. 1b - Expansion AUX scan stops after AUXSEL1 are served, and the scan sequence resets to EXPAUX4B[AUXSEL0].
6-4 —	Reserved
3	AUX Select Disable

Table continues on the next page...

Field	Function
MSDIS0	0b - Expansion AUX scan continues. 1b - Expansion AUX scan stops after AUXSEL0 are served, and the scan sequence resets to EXPAUX4B[AUXSEL0].
2-0 —	Reserved

22.7.1.49 ANB4 Expansion MUX Control Register 1 (EXPMUX4B1)

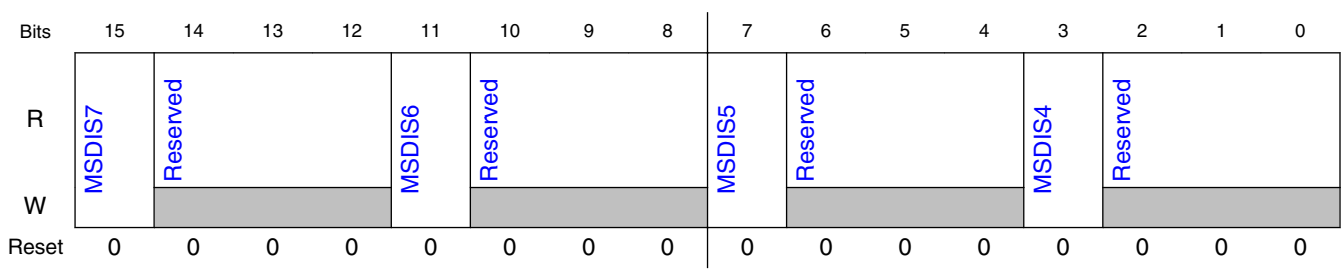
22.7.1.49.1 Offset

Register	Offset
EXPMUX4B1	7Fh

22.7.1.49.2 Function

The AUX Select Disable (MSDIS) bit is used to stop the scan. When a scan is stopped, the scan sequence is reset to EXPAUX4B[AUXSEL0].

22.7.1.49.3 Diagram



22.7.1.49.4 Fields

Field	Function
15 MSDIS7	AUX Select Disable No matter the bit value is 0 or 1, Expansion AUX scan stops after AUXSEL7 is served, and the scan sequence resets to EXPAUX4B[AUXSEL0].
14-12 —	Reserved

Table continues on the next page...

Field	Function
11 MSDIS6	AUX Select Disable 0b - Expansion AUX scan continues. 1b - Expansion AUX scan stops after AUXSEL6 are served, and the scan sequence resets to EXPAUX4B[AUXSEL0].
10-8 —	Reserved
7 MSDIS5	AUX Select Disable 0b - Expansion AUX scan continues. 1b - Expansion AUX scan stops after AUXSEL5 are served, and the scan sequence resets to EXPAUX4B[AUXSEL0].
6-4 —	Reserved
3 MSDIS4	AUX Select Disable 0b - Expansion AUX scan continues. 1b - Expansion AUX scan stops after AUXSEL4 are served, and the scan sequence resets to EXPAUX4B[AUXSEL0].
2-0 —	Reserved

Chapter 23
Comparator (CMP)

23.1 Chip-specific information for this module

23.1.1 Comparator Channel Assignments

Table 23-1. Comparator Channel Assignments

Module	Comparator/Mux Channel	Source
CMPA	Channel 7	8-bit DAC (in CMPA module)
	Channel 6	OPAMPA_OUT
	Channel 5	GPIOC6 (CMP_REF)
	Channel 4	12-bit DACA
	Channel 3	GPIOA0 (CMPA_IN3)
	Channel 2	GPIOA3 (CMPA_IN2)
	Channel 1	GPIOA2 (CMPA_IN1)
	Channel 0	GPIOA1 (CMPA_IN0)
CMPB	Channel 7	8-bit DAC (in CMPB module)
	Channel 6	OPAMPB_OUT
	Channel 5	GPIOC6 (CMP_REF)
	Channel 4	12-bit DACA
	Channel 3	GPIOB0 (CMPB_IN3)
	Channel 2	GPIOB7 (CMPB_IN2)
	Channel 1	GPIOB6 (CMPB_IN1)
	Channel 0	GPIOB1 (CMPB_IN0)
CMPC	Channel 7	8-bit DAC (in CMPC module)
	Channel 6	OPAMPA_OUT
	Channel 5	GPIOC6 (CMP_REF)
	Channel 4	12-bit DACA
	Channel 3	GPIOB2 (CMPC_IN3)
	Channel 2	GPIOB5 (CMPC_IN2)

Table continues on the next page...

Table 23-1. Comparator Channel Assignments (continued)

Module	Comparator/Mux Channel	Source
	Channel 1	GPIOB4 (CMPC_IN1)
	Channel 0	GPIOB3 (CMPC_IN0)
CMPD	Channel 7	8-bit DAC (in CMPD module)
	Channel 6	OPAMPB_OUT
	Channel 5	GPIOC6 (CMP_REF)
	Channel 4	12-bit DACA
	Channel 3	GPIOA7 (CMPD_IN3)
	Channel 2	GPIOA6 (CMPD_IN2)
	Channel 1	GPIOA5 (CMPD_IN1)
	Channel0	GPIOA4 (CMPD_IN0)

23.1.2 Comparator Voltage References

The 8-bit DAC sub-block supports the selection of two references. For this device, both the Vin1 input and the Vin2 input are connected to Vdd .

23.2 Introduction

The comparator (CMP) module provides a circuit for comparing two analog input voltages. The comparator circuit is designed to operate across the full range of the supply voltage, known as rail-to-rail operation.

The Analog MUX (ANMUX) provides a circuit for selecting an analog input signal from eight channels. One signal is provided by the 8-bit digital-to-analog converter (DAC). The mux circuit is designed to operate across the full range of the supply voltage.

The 8-bit DAC is 256-tap resistor ladder network which provides a selectable voltage reference for applications where voltage reference is needed. The 256-tap resistor ladder network divides the supply reference V_{in} into 256 voltage levels. A 8-bit digital signal input selects the output voltage level, which varies from V_{in} to $V_{in}/256$. V_{in} can be selected from two voltage sources, V_{in1} and V_{in2}. The 8-bit DAC from a comparator is available as an on-chip internal signal only and is not available externally to a pin.

23.2.1 CMP features

The CMP has the following features:

- Operational over the entire supply range
- Inputs may range from rail to rail
- Programmable hysteresis control
- Selectable interrupt on rising-edge, falling-edge, or both rising or falling edges of the comparator output
- Selectable inversion on comparator output
- Capability to produce a wide range of outputs such as:
  - Sampled
  - Windowed, which is ideal for certain PWM zero-crossing-detection applications
  - Digitally filtered:
    - Filter can be bypassed
    - Can be clocked via external SAMPLE signal or scaled bus clock
- External hysteresis can be used at the same time that the output filter is used for internal functions
- Two software selectable performance levels:
  - Shorter propagation delay at the expense of higher power
  - Low power, with longer propagation delay
- DMA transfer support
  - A comparison event can be selected to trigger a DMA transfer
- Functional in all modes of operation
- The window and filter functions are not available in the following modes:
  - Stop
  - VLPS
  - LPS

23.2.2 8-bit DAC key features

The 8-bit DAC has the following features:

- 8-bit resolution
- Selectable supply reference source
- Power Down mode to conserve power when not in use
- Option to route the output to internal comparator input

23.2.3 ANMUX key features

The ANMUX has the following features:

- Two 8-to-1 channel mux
- Operational over the entire supply range

23.2.4 CMP, DAC and ANMUX diagram

The following figure shows the block diagram for the High-Speed Comparator, DAC, and ANMUX modules.

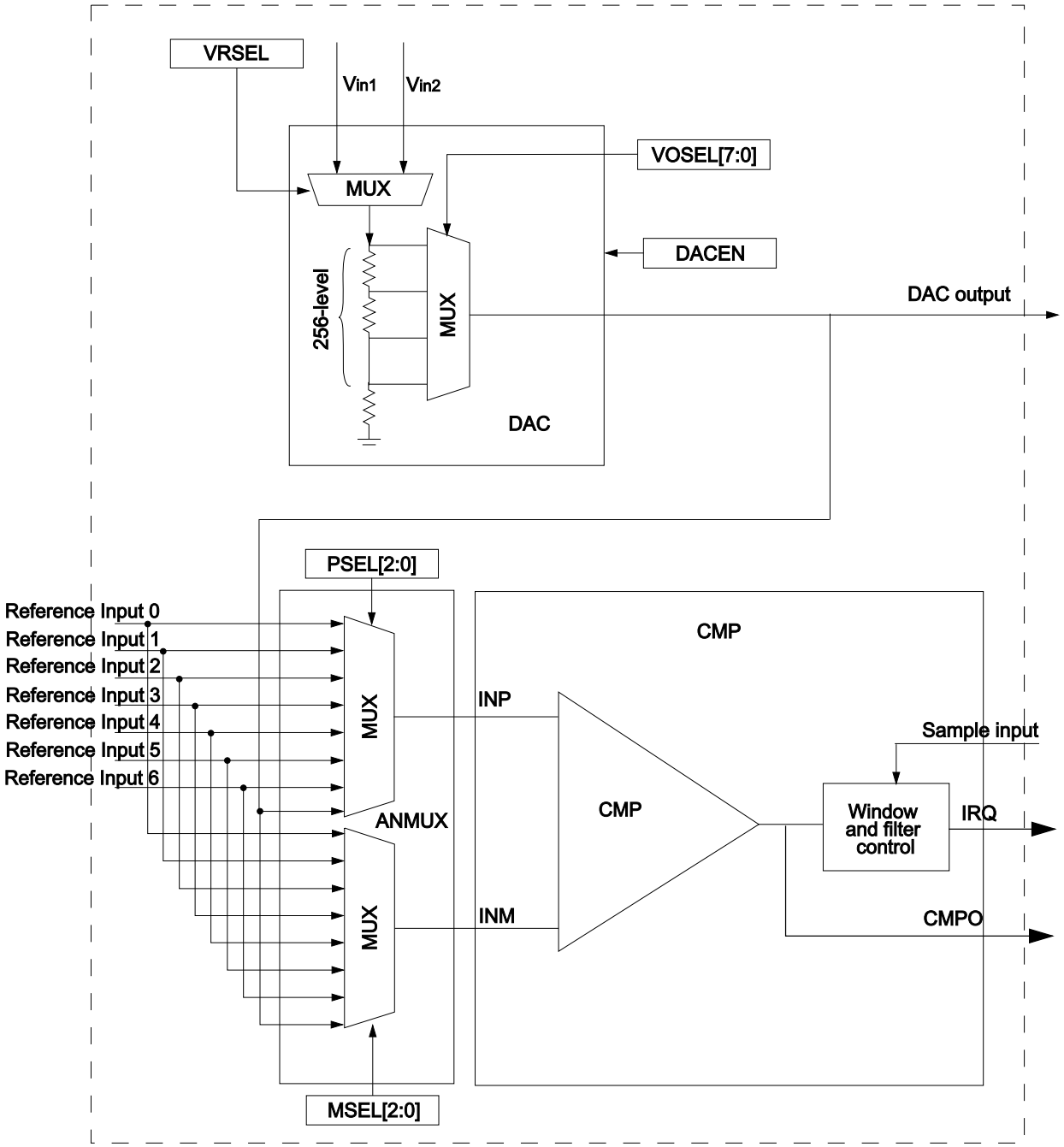


Figure 23-1. CMP, DAC and ANMUX block diagram

23.2.5 CMP block diagram

The following figure shows the block diagram for the CMP module.

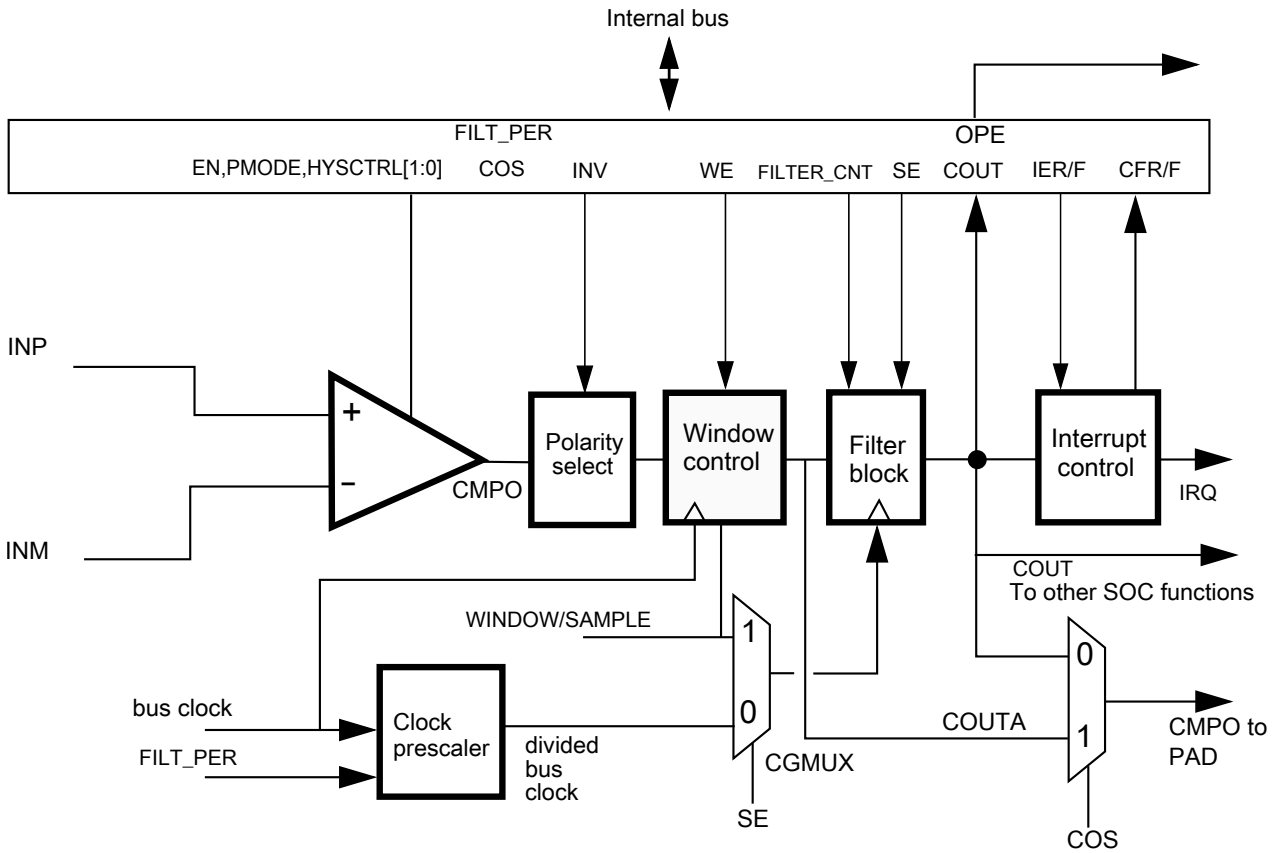


Figure 23-2. Comparator module block diagram

In the CMP block diagram:

- The Window Control block is bypassed when CR1[WE] = 0
- If CR1[WE] = 1, the comparator output will be sampled on every bus clock when WINDOW=1 to generate COUTA. Sampling does NOT occur when WINDOW = 0.
- The Filter block is bypassed when not in use.
- The Filter block acts as a simple sampler if the filter is bypassed and CR0[FILTER_CNT] is set to 0x01.
- The Filter block filters based on multiple samples when the filter is bypassed and CR0[FILTER_CNT] is set greater than 0x01.
  - If CR1[SE] = 1, the external SAMPLE input is used as sampling clock
  - If CR1[SE] = 0, the divided bus clock is used as sampling clock

- If enabled, the Filter block will incur up to one bus clock additional latency penalty on COUT due to the fact that COUT, which is crossing clock domain boundaries, must be resynchronized to the bus clock.
- CR1[WE] and CR1[SE] are mutually exclusive.

23.3 Memory map/register definitions

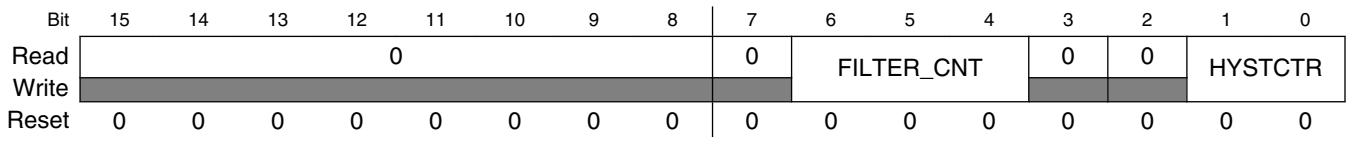
Address offsets are in terms of 16-bit words for DSC architectures. Each 8-bit register occupies bits 0-7 of the 16-bit width. The other 8 bits are read-only and always read 0.

CMP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E020	CMP Control Register 0 (CMPA_CR0)	16	R/W	0000h	23.3.1/630
E021	CMP Control Register 1 (CMPA_CR1)	16	R/W	0000h	23.3.2/631
E022	CMP Filter Period Register (CMPA_FPR)	16	R/W	0000h	23.3.3/632
E023	CMP Status and Control Register (CMPA_SCR)	16	R/W	0000h	23.3.4/633
E024	DAC Control Register (CMPA_DACCR)	16	R/W	0000h	23.3.5/634
E025	MUX Control Register (CMPA_MUXCR)	16	R/W	0000h	23.3.6/634
E028	CMP Control Register 0 (CMPB_CR0)	16	R/W	0000h	23.3.1/630
E029	CMP Control Register 1 (CMPB_CR1)	16	R/W	0000h	23.3.2/631
E02A	CMP Filter Period Register (CMPB_FPR)	16	R/W	0000h	23.3.3/632
E02B	CMP Status and Control Register (CMPB_SCR)	16	R/W	0000h	23.3.4/633
E02C	DAC Control Register (CMPB_DACCR)	16	R/W	0000h	23.3.5/634
E02D	MUX Control Register (CMPB_MUXCR)	16	R/W	0000h	23.3.6/634
E030	CMP Control Register 0 (CMPC_CR0)	16	R/W	0000h	23.3.1/630
E031	CMP Control Register 1 (CMPC_CR1)	16	R/W	0000h	23.3.2/631
E032	CMP Filter Period Register (CMPC_FPR)	16	R/W	0000h	23.3.3/632
E033	CMP Status and Control Register (CMPC_SCR)	16	R/W	0000h	23.3.4/633
E034	DAC Control Register (CMPC_DACCR)	16	R/W	0000h	23.3.5/634
E035	MUX Control Register (CMPC_MUXCR)	16	R/W	0000h	23.3.6/634
E038	CMP Control Register 0 (CMPD_CR0)	16	R/W	0000h	23.3.1/630
E039	CMP Control Register 1 (CMPD_CR1)	16	R/W	0000h	23.3.2/631
E03A	CMP Filter Period Register (CMPD_FPR)	16	R/W	0000h	23.3.3/632
E03B	CMP Status and Control Register (CMPD_SCR)	16	R/W	0000h	23.3.4/633
E03C	DAC Control Register (CMPD_DACCR)	16	R/W	0000h	23.3.5/634
E03D	MUX Control Register (CMPD_MUXCR)	16	R/W	0000h	23.3.6/634

23.3.1 CMP Control Register 0 (CMPx_CR0)

Address: Base address + 0h offset



CMPx_CR0 field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–4 FILTER_CNT	Filter Sample Count Represents the number of consecutive samples that must agree prior to the comparator ouput filter accepting a new output state. For information regarding filter programming and latency, see the Functional description. 000 Filter is disabled. If SE = 1, then COUT is a logic 0. This is not a legal state, and is not recommended. If SE = 0, COUT = COUTA. 001 One sample must agree. The comparator output is simply sampled. 010 2 consecutive samples must agree. 011 3 consecutive samples must agree. 100 4 consecutive samples must agree. 101 5 consecutive samples must agree. 110 6 consecutive samples must agree. 111 7 consecutive samples must agree.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
HYSTCTR	Comparator hard block hysteresis control Defines the programmable hysteresis level. The hysteresis values associated with each level are device-specific. See the Data Sheet of the device for the exact values. 00 Level 0 01 Level 1 10 Level 2 11 Level 3

23.3.2 CMP Control Register 1 (CMPx_CR1)

Address: Base address + 1h offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	SE	WE	COWZ	PMODE	INV	COS	OPE	EN
Reset	0	0	0	0	0	0	0	0

CMPx_CR1 field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SE	Sample Enable At any given time, either SE or WE can be set. If a write to this register attempts to set both, then SE is set and WE is cleared. However, avoid writing 1s to both field locations because this "11" case is reserved and may change in future implementations. 0 Sampling mode is not selected. 1 Sampling mode is selected.
6 WE	Windowing Enable At any given time, either SE or WE can be set. If a write to this register attempts to set both, then SE is set and WE is cleared. However, avoid writing 1s to both field locations because this "11" case is reserved and may change in future implementations. 0 Windowing mode is not selected. 1 Windowing mode is selected.
5 COWZ	COUTA out of window is zero enable. 0 In windowing mode, when WINDOW signal changes from 1 to 0, COUTA holds the last latched value before WINDOW signal falls to 0. 1 In windowing mode, when WINDOW signal changes from 1 to 0, COUTA is forced to 0.
4 PMODE	Power Mode Select See the electrical specifications table in the device Data Sheet for details. 0 Low-Speed (LS) Comparison mode selected. In this mode, CMP has slower output propagation delay and lower current consumption. 1 High-Speed (HS) Comparison mode selected. In this mode, CMP has faster output propagation delay and higher current consumption.
3 INV	Comparator INVERT Allows selection of the polarity of the analog comparator function. It is also driven to the COUT output, on both the device pin and as SCR[COUT], when OPE=0.

Table continues on the next page...

CMPx_CR1 field descriptions (continued)

Field	Description
	0 Does not invert the comparator output. 1 Inverts the comparator output.
2 COS	Comparator Output Select 0 Set the filtered comparator output (CMPO) to equal COUT. 1 Set the unfiltered comparator output (CMPO) to equal COUTA.
1 OPE	Comparator Output Pin Enable 0 CMPO is not available on the associated CMPO output pin. If the comparator does not own the pin, this field has no effect. 1 CMPO is available on the associated CMPO output pin. The comparator output (CMPO) is driven out on the associated CMPO output pin if the comparator owns the pin. If the comparator does not own the field, this bit has no effect.
0 EN	Comparator Module Enable Enables the Analog Comparator module. When the module is not enabled, it remains in the off state, and consumes no power. When the user selects the same input from analog mux to the positive and negative port, the comparator is disabled automatically. 0 Analog Comparator is disabled. 1 Analog Comparator is enabled.

23.3.3 CMP Filter Period Register (CMPx_FPR)

Address: Base address + 2h offset

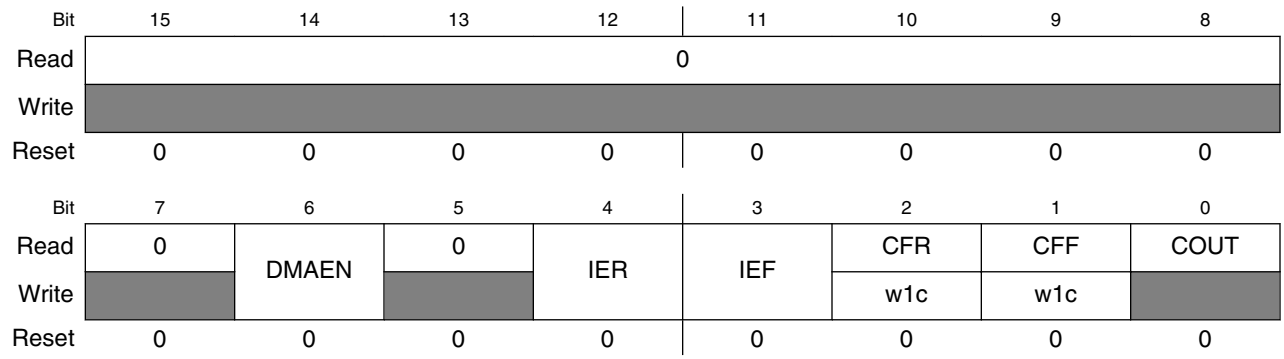
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								FILT_PER							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CMPx_FPR field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
FILT_PER	Filter Sample Period Specifies the sampling period, in bus clock cycles, of the comparator output filter, when CR1[SE]=0. Setting FILT_PER to 0x0 disables the filter. Filter programming and latency details appear in the Functional description. This field has no effect when CR1[SE]=1. In that case, the external SAMPLE signal is used to determine the sampling period.

23.3.4 CMP Status and Control Register (CMPx_SCR)

Address: Base address + 3h offset



CMPx_SCR field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 DMAEN	DMA Enable Control Enables the DMA transfer triggered from the CMP module. See the "DMA support" section for more details. • When this field and IER are set, a DMA request is asserted when CFR is set. • When this field and IEF are set, a DMA request is asserted when CFF is set. 0 DMA is disabled. 1 DMA is enabled.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 IER	Comparator Interrupt Enable Rising Enables the CFR interrupt from the CMP. When this field is set and DMAEN is not set, an interrupt will be asserted when CFR is set. 0 Interrupt is disabled. 1 Interrupt is enabled.
3 IEF	Comparator Interrupt Enable Falling Enables the CFF interrupt from the CMP. When this field is set and DMAEN is not set, an interrupt will be asserted when CFF is set. 0 Interrupt is disabled. 1 Interrupt is enabled.
2 CFR	Analog Comparator Flag Rising Detects a rising-edge on COUT, when set, during normal operation. CFR is cleared by writing 1 to it. During Stop modes, CFR is level sensitive .

Table continues on the next page...

CMPx_SCR field descriptions (continued)

Field	Description
	0 Rising-edge on COUT has not been detected. 1 Rising-edge on COUT has occurred.
1 CFF	Analog Comparator Flag Falling Detects a falling-edge on COUT, when set, during normal operation. CFF is cleared by writing 1 to it. During Stop modes, CFF is level sensitive . 0 Falling-edge on COUT has not been detected. 1 Falling-edge on COUT has occurred.
0 COUT	Analog Comparator Output Returns the current value of the Analog Comparator output, when read. The field is reset to 0 and will read as CR1[INV] when the Analog Comparator module is disabled, that is, when CR1[EN] = 0. Writes to this field are ignored.

23.3.5 DAC Control Register (CMPx_DACCR)

Address: Base address + 4h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								VOSEL							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CMPx_DACCR field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
VOSEL	DAC Output Voltage Select Selects an output voltage from one of 256 distinct levels. `DACO = (V in /256) * (VOSEL[7:0] + 1)` , so the DACO range is from $V_{in}/256$ to V_{in} .

23.3.6 MUX Control Register (CMPx_MUXCR)

Address: Base address + 5h offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	DACEN	VRSEL	PSEL				MSEL	
Reset	0	0	0	0	0	0	0	0

CMPx_MUXCR field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 DACEN	DAC Enable Enables the DAC. When the DAC is disabled, it is powered down to conserve power. 0 DAC is disabled. 1 DAC is enabled.
6 VRSEL	Supply Voltage Reference Source Select 0 V_{in1} is selected as resistor ladder network supply reference. 1 V_{in2} is selected as resistor ladder network supply reference.
5–3 PSEL	Plus Input Mux Control Determines which input is selected for the plus input of the comparator. For INx inputs, see CMP, DAC, and ANMUX block diagrams. **NOTE:** When an inappropriate operation selects the same input for both muxes, the comparator automatically shuts down to prevent itself from becoming a noise generator. 000 IN0 001 IN1 010 IN2 011 IN3 100 IN4 101 IN5 110 IN6 111 IN7
MSEL	Minus Input Mux Control Determines which input is selected for the minus input of the comparator. For INx inputs, see CMP, DAC, and ANMUX block diagrams. **NOTE:** When an inappropriate operation selects the same input for both muxes, the comparator automatically shuts down to prevent itself from becoming a noise generator. 000 IN0 001 IN1 010 IN2 011 IN3 100 IN4 101 IN5 110 IN6 111 IN7

23.4 Functional description

The CMP module can be used to compare two analog input voltages applied to INP and INM.

CMPO is high when the non-inverting input is greater than the inverting input, and is low when the non-inverting input is less than the inverting input. This signal can be selectively inverted by setting CR1[INV] = 1.

SCR[IER] and SCR[IEF] are used to select the condition which will cause the CMP module to assert an interrupt to the processor. SCR[CFF] is set on a falling-edge and SCR[CFR] is set on rising-edge of the comparator output. The optionally filtered CMPO can be read directly through SCR[COUT].

23.4.1 CMP functional modes

There are the following main sub-blocks to the CMP module:

- The comparator itself
- The window function
- The filter function

The filter, CR0[FILTER_CNT], can be clocked from an internal or external clock source. The filter is programmable with respect to the number of samples that must agree before a change in the output is registered. In the simplest case, only one sample must agree. In this case, the filter acts as a simple sampler.

The external sample input is enabled using CR1[SE]. When set, the output of the comparator is sampled only on rising edges of the sample input.

The "windowing mode" is enabled by setting CR1[WE]. When set, the comparator output is sampled only when WINDOW=1. This feature can be used to ignore the comparator output during time periods in which the input voltages are not valid. This is especially useful when implementing zero-crossing-detection for certain PWM applications.

The comparator filter and sampling features can be combined as shown in the following table. Individual modes are discussed below.

Table 23-2. Comparator sample/filter controls

Mode #	CR1[EN]	CR1[WE]	CR1[SE]	CR0[FILTER_CNT]	FPR[FILT_PER]	Operation
1	0	X	X	X	X	**Disabled** See the Disabled mode (# 1).
2A	1	0	0	0x00	X	**Continuous Mode** See the Continuous mode (#s 2A & 2B).
2B	1	0	0	X	0x00	
3A	1	0	1	0x01	X	**Sampled, Non-Filtered mode**
3B	1	0	0	0x01	> 0x00	

Table continues on the next page...

Table 23-2. Comparator sample/filter controls (continued)

Mode #	CR1[EN]	CR1[WE]	CR1[SE]	CR0[FILTER_CNT]	FPR[FILT_PER]	Operation
						See the Sampled, Non-Filtered mode (#s 3A & 3B).
4A	1	0	1	> 0x01	X	**Sampled, Filtered mode**
4B	1	0	0	> 0x01	> 0x00	See the Sampled, Filtered mode (#s 4A & 4B).
5A	1	1	0	0x00	X	**Windowed mode**
5B	1	1	0	X	0x00	Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA. See the Windowed mode (#s 5A & 5B).
6	1	1	0	0x01	0x01–0xFF	**Windowed/Resampled mode** Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA, which is then resampled on an interval determined by FILT_PER to generate COUT. See the Windowed/Resampled mode (# 6).
7	1	1	0	> 0x01	0x01–0xFF	**Windowed/Filtered mode** Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA, which is then resampled and filtered to generate COUT. See the Windowed/Filtered mode (#7).
All other combinations of CR1[EN], CR1[WE], CR1[SE], CR0[FILTER_CNT], and FPR[FILT_PER] are illegal.						

For cases where a comparator is used to drive a fault input, for example, for a motor-control module such as FTM, it must be configured to operate in Continuous mode so that an external fault can immediately pass through the comparator to the target fault circuitry.

Note

Filtering and sampling settings must be changed only after setting CR1[SE]=0 and CR0[FILTER_CNT]=0x00. This resets the filter to a known state.

23.4.1.1 Disabled mode (# 1)

In Disabled mode, the analog comparator is non-functional and consumes no power. CMPO is 0 in this mode.

23.4.1.2 Continuous mode (#s 2A & 2B)

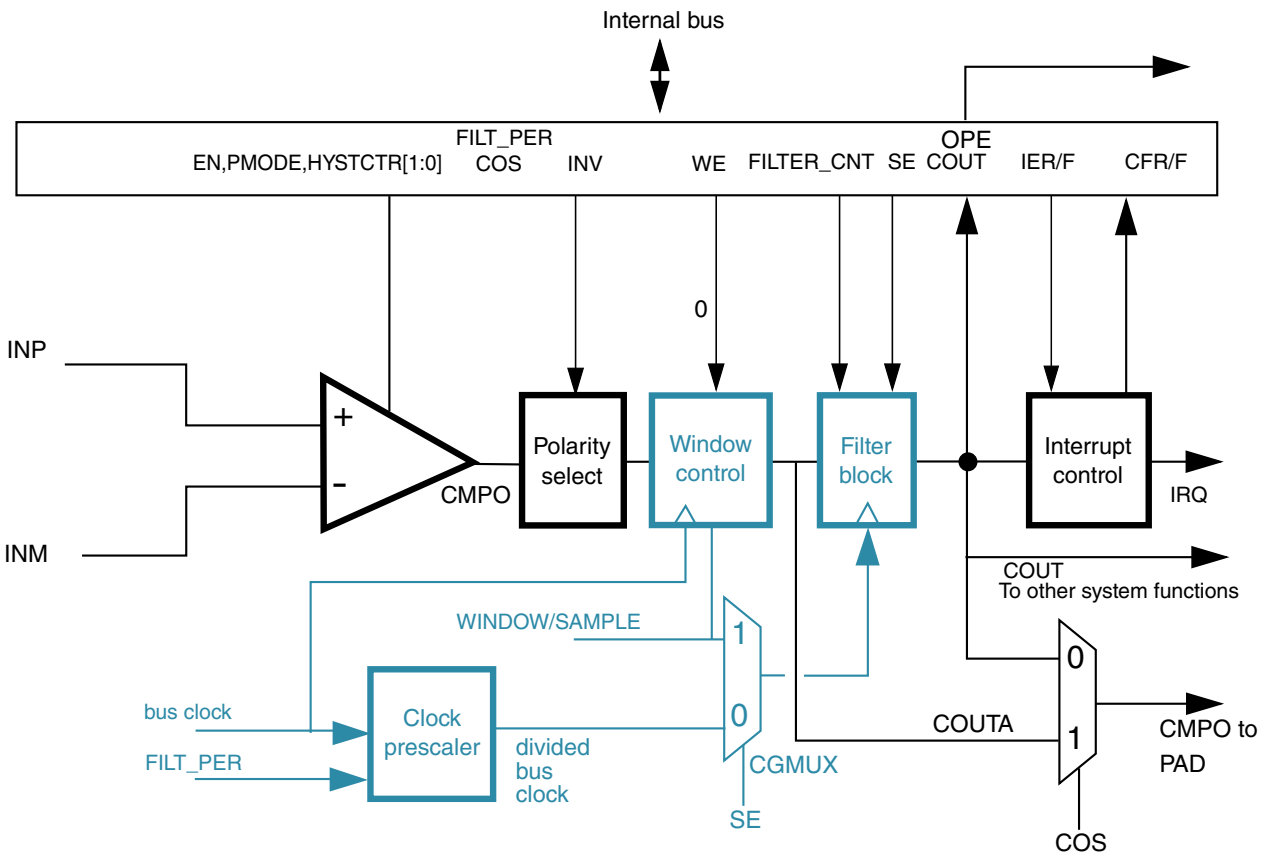


Figure 23-3. Comparator operation in Continuous mode

The analog comparator block is powered and active. CMPO may be optionally inverted, but is not subject to external sampling or filtering. Both window control and filter blocks are completely bypassed. SCR[COUT] is updated continuously. The path from comparator input pins to output pin is operating in combinational unclocked mode. COUT and COUTA are identical.

For control configurations which result in disabling the filter block, see the Filter Block Bypass Logic diagram.

23.4.1.3 Sampled, Non-Filtered mode (#s 3A & 3B)

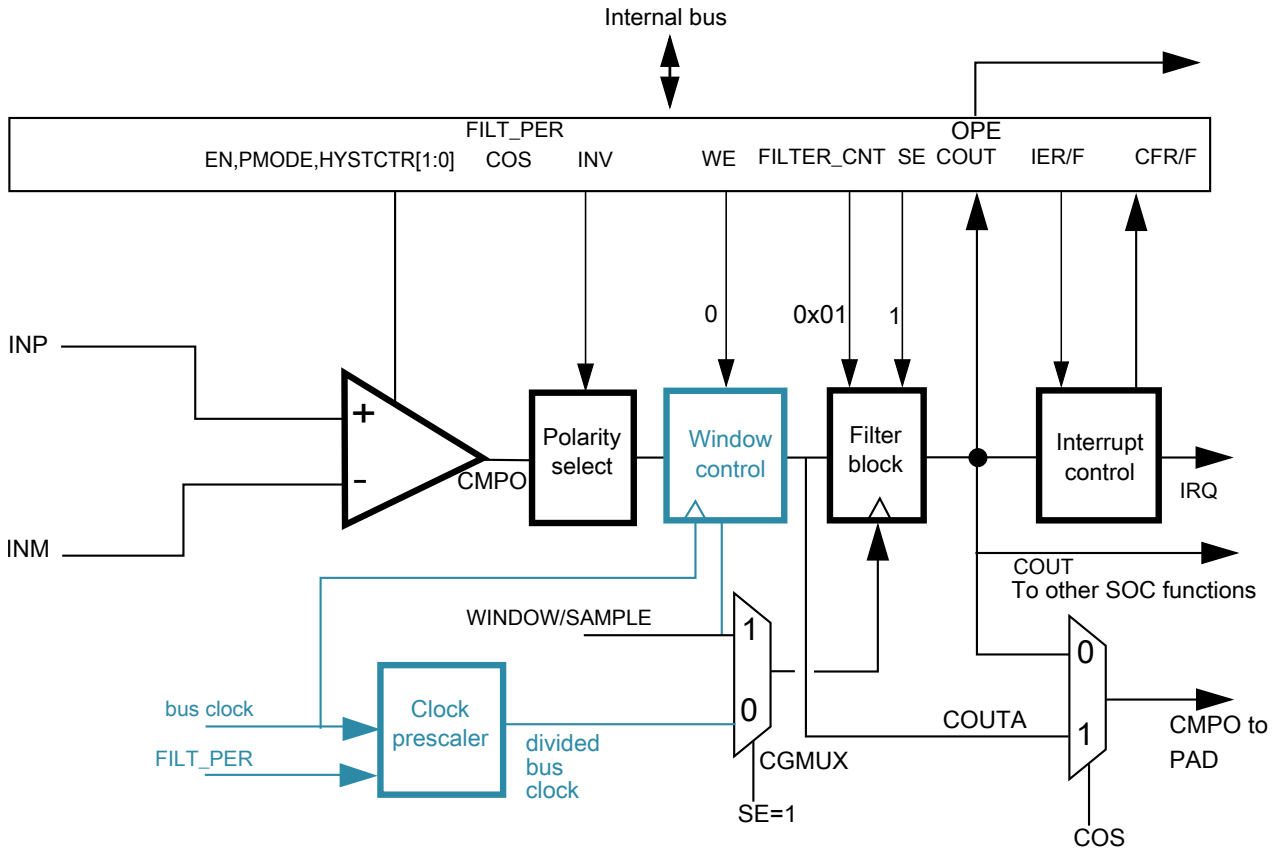


Figure 23-4. Sampled, Non-Filtered (# 3A): sampling point externally driven

In Sampled, Non-Filtered mode, the analog comparator block is powered and active. The path from analog inputs to COUTA is combinational unclocked. Windowing control is completely bypassed. COUTA is sampled whenever a rising-edge is detected on the filter block clock input.

The only difference in operation between Sampled, Non-Filtered (# 3A) and Sampled, Non-Filtered (# 3B) is in how the clock to the filter block is derived. In #3A, the clock to filter block is externally derived while in #3B, the clock to filter block is internally derived.

The comparator filter has no other function than sample/hold of the comparator output in this mode (# 3B).

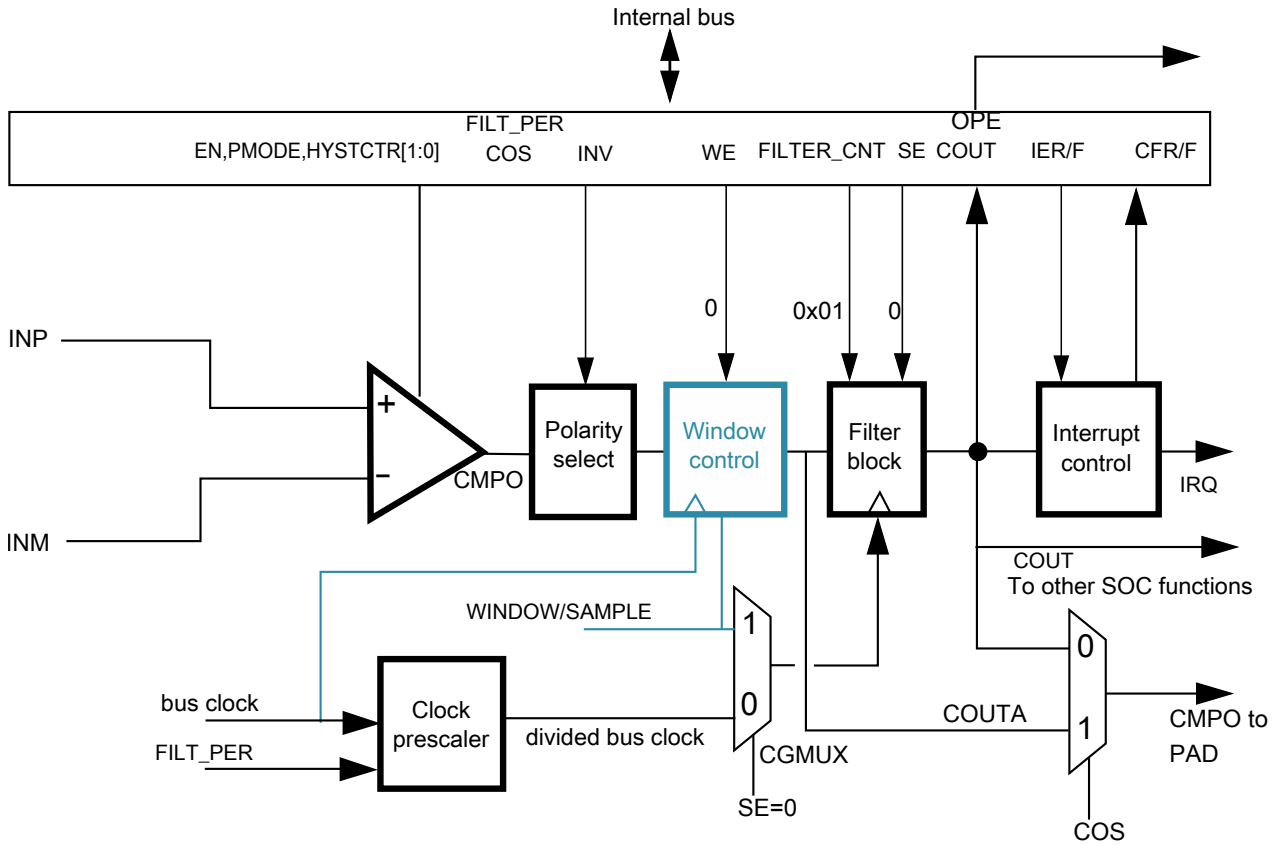


Figure 23-5. Sampled, Non-Filtered (# 3B): sampling interval internally derived

23.4.1.4 Sampled, Filtered mode (#s 4A & 4B)

In Sampled, Filtered mode, the analog comparator block is powered and active. The path from analog inputs to COUTA is combinational unclocked. Windowing control is completely bypassed. COUTA is sampled whenever a rising edge is detected on the filter block clock input.

The only difference in operation between Sampled, Non-Filtered (# 3A) and Sampled, Filtered (# 4A) is that, now, CR0[FILTER_CNT]>1, which activates filter operation.

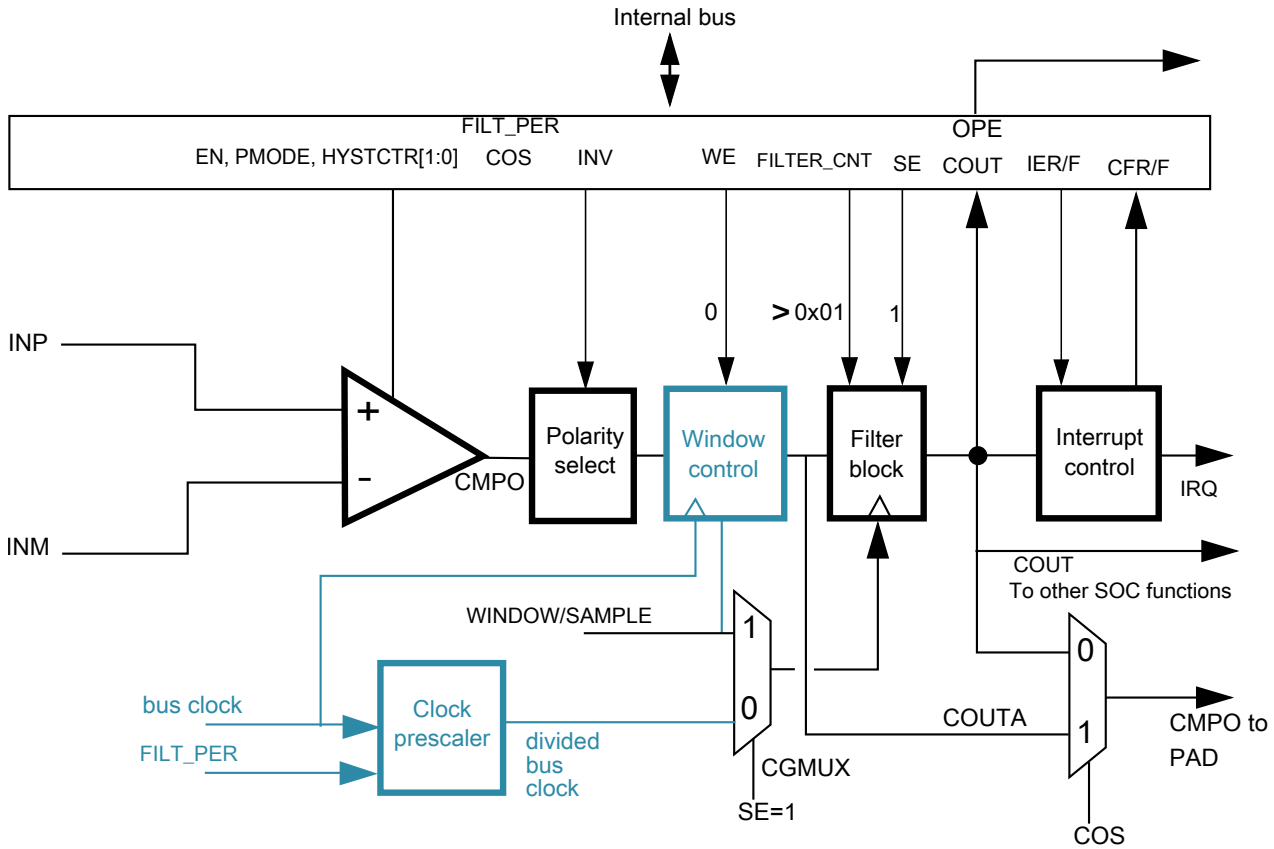


Figure 23-6. Sampled, Filtered (# 4A): sampling point externally driven

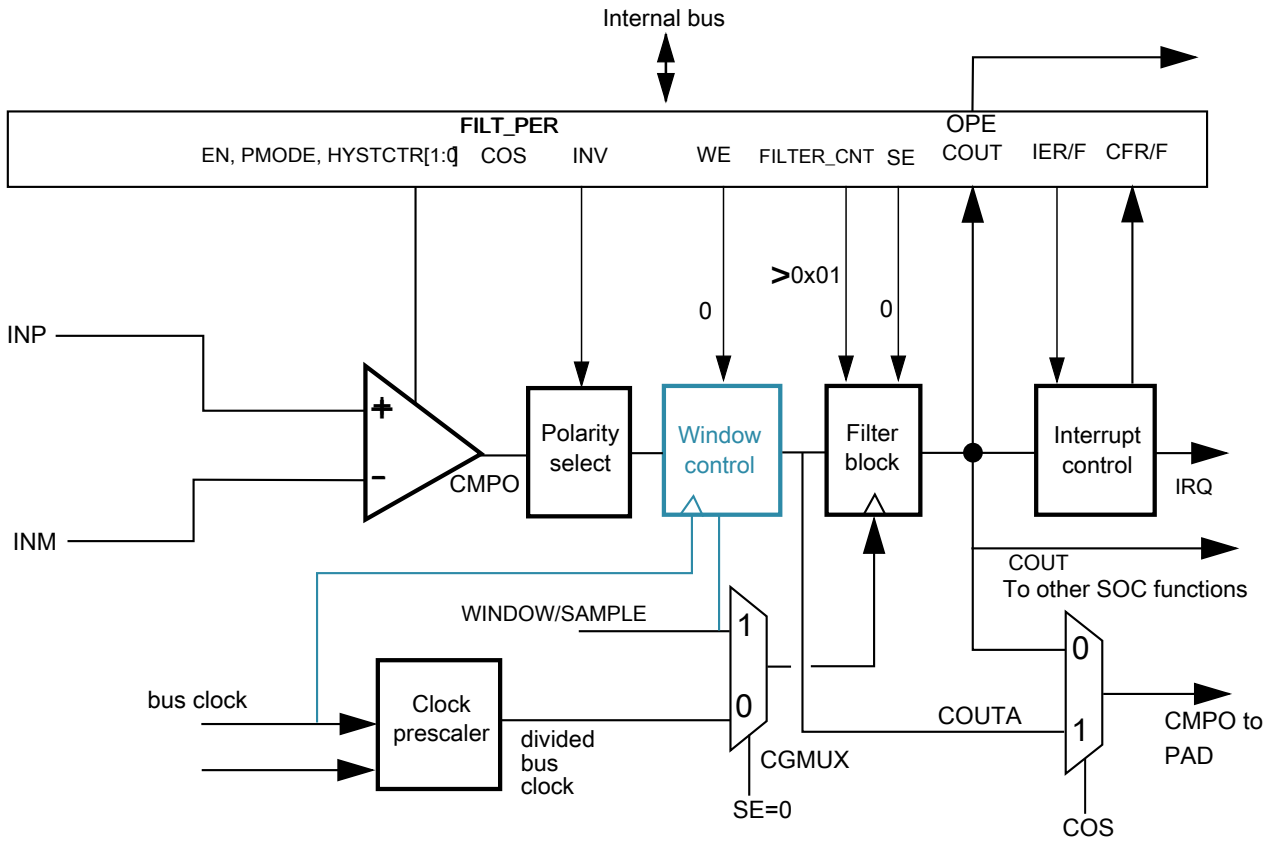


Figure 23-7. Sampled, Filtered (# 4B): sampling point internally derived

The only difference in operation between Sampled, Non-Filtered (# 3B) and Sampled, Filtered (# 4B) is that now, CR0[FILTER_CNT]>1, which activates filter operation.

23.4.1.5 Windowed mode (#s 5A & 5B)

The following figure illustrates comparator operation in the Windowed mode, ignoring latency of the analog comparator, polarity select, and window control block. It also assumes that the polarity select is set to non-inverting state.

NOTE

The analog comparator output is passed to COUTA only when the WINDOW signal is high.

In actual operation, COUTA may lag the analog inputs by up to one bus clock cycle plus the combinational path delay through the comparator and polarity select logic.

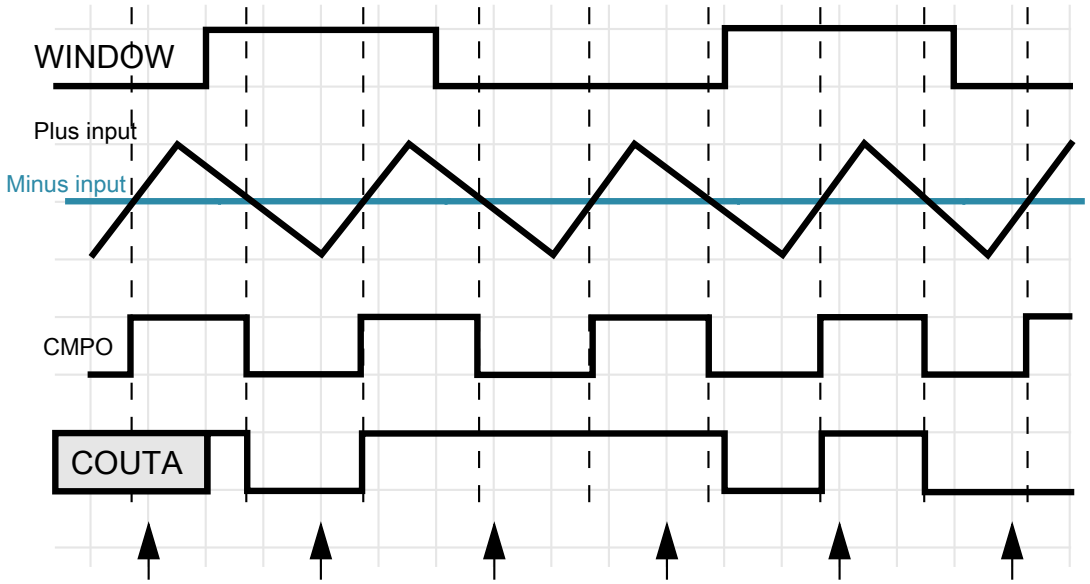


Figure 23-8. Windowed mode operation

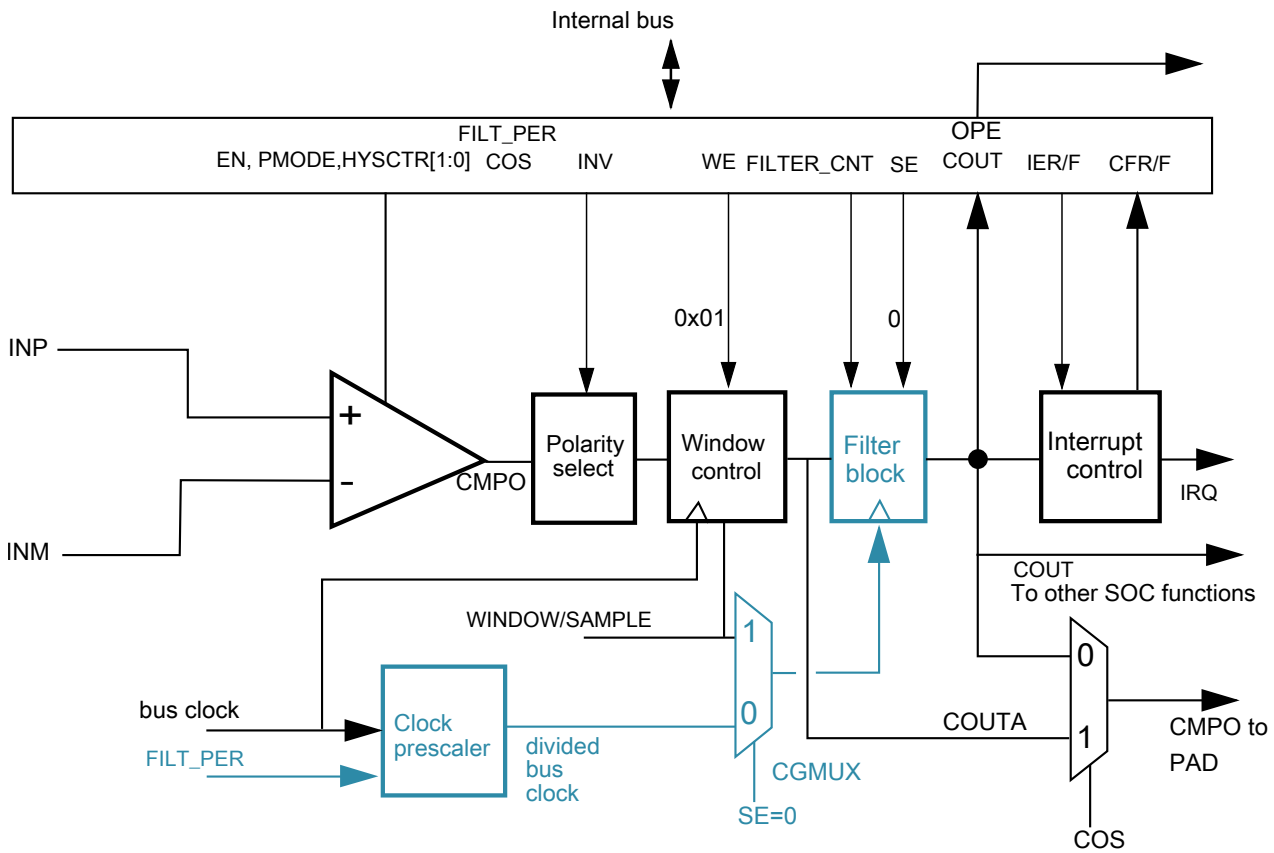


Figure 23-9. Windowed mode

For control configurations which result in disabling the filter block, see Filter Block Bypass Logic diagram.

When any windowed mode is active, COUTA is clocked by the bus clock whenever WINDOW = 1. The last latched value is held when WINDOW = 0.

When register bit COWZ is enabled, and when WINDOW=0, COUTA is forced to zero, as shown in the following waveform.

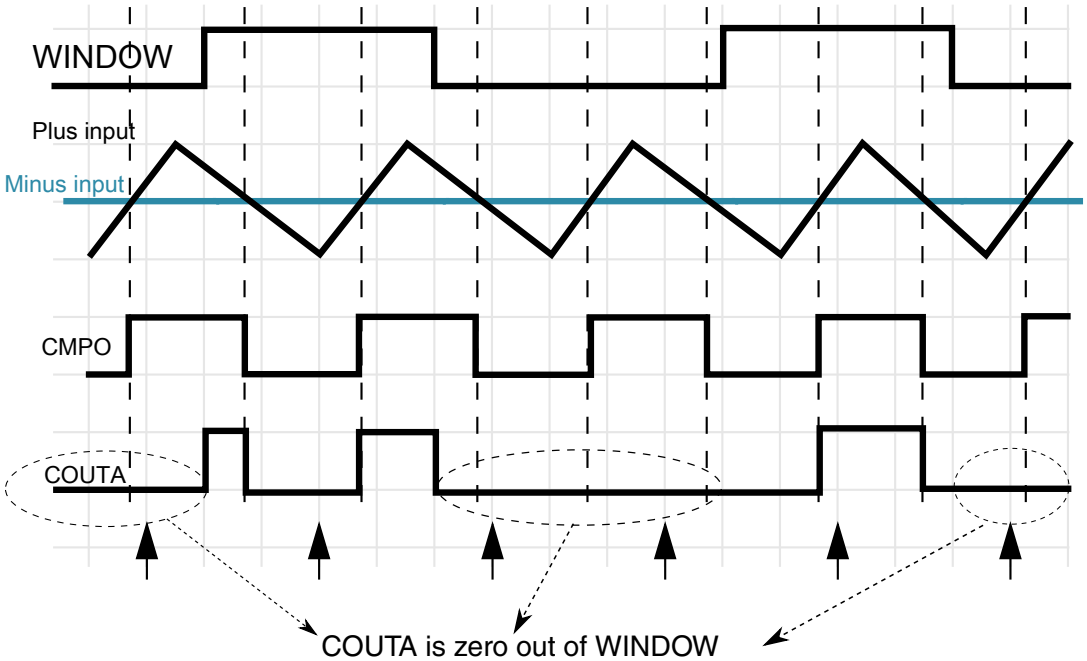


Figure 23-10. Windowed mode operation with COWZ enabled

23.4.1.6 Windowed/Resampled mode (# 6)

The following figure uses the same input stimulus shown in the figure "Windowed mode operation", and adds resampling of COUTA to generate COUT. Samples are taken at the time points indicated by the arrows in the figure. Again, prop delays and latency are ignored for the sake of clarity.

This example was generated solely to demonstrate operation of the comparator in windowed/resampled mode, and does not reflect any specific application. Depending upon the sampling rate and window placement, COUT may not see zero-crossing events detected by the analog comparator. Sampling period and/or window placement must be carefully considered for a given application.

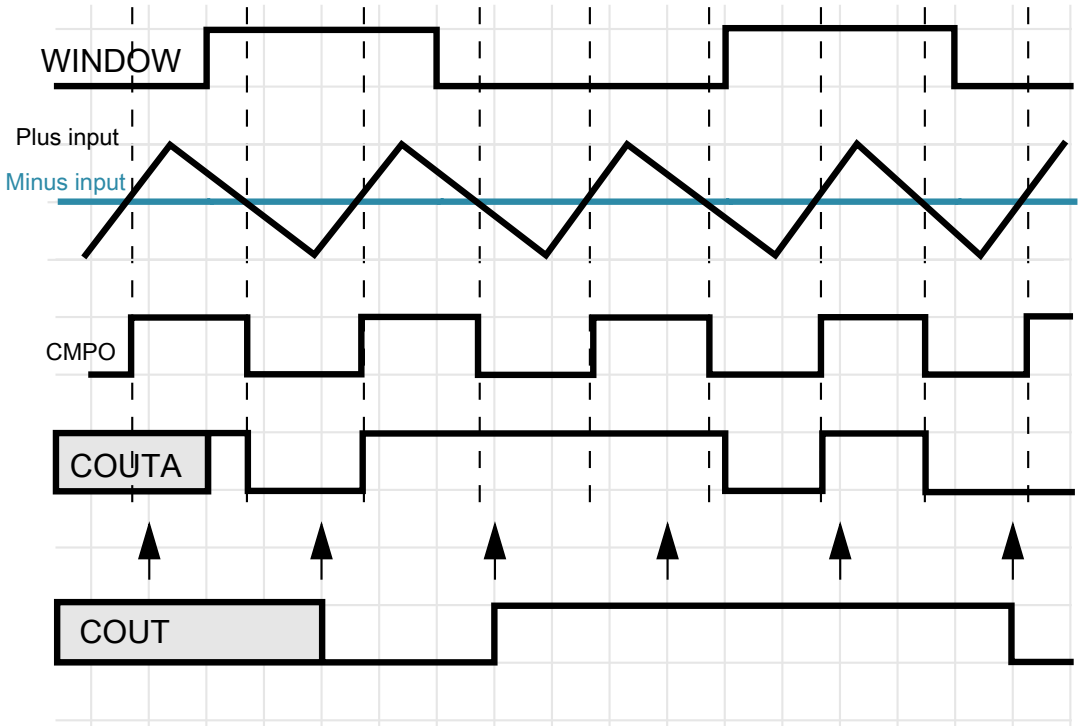


Figure 23-11. Windowed/resampled mode operation

This mode of operation results in an unfiltered string of comparator samples where the interval between the samples is determined by FPR[FILT_PER] and the bus clock rate. Configuration for this mode is virtually identical to that for the Windowed/Filtered Mode shown in the next section. The only difference is that the value of CR0[FILTER_CNT] must be 1.

23.4.1.7 Windowed/Filtered mode (#7)

This is the most complex mode of operation for the comparator block, as it uses both windowing and filtering features. It also has the highest latency of any of the modes. This can be approximated: up to 1 bus clock synchronization in the window function + ((CR0[FILTER_CNT] * FPR[FILT_PER]) + 1) * bus clock for the filter function.

When any windowed mode is active, COUTA is clocked by the bus clock whenever WINDOW = 1. The last latched value is held when WINDOW = 0.

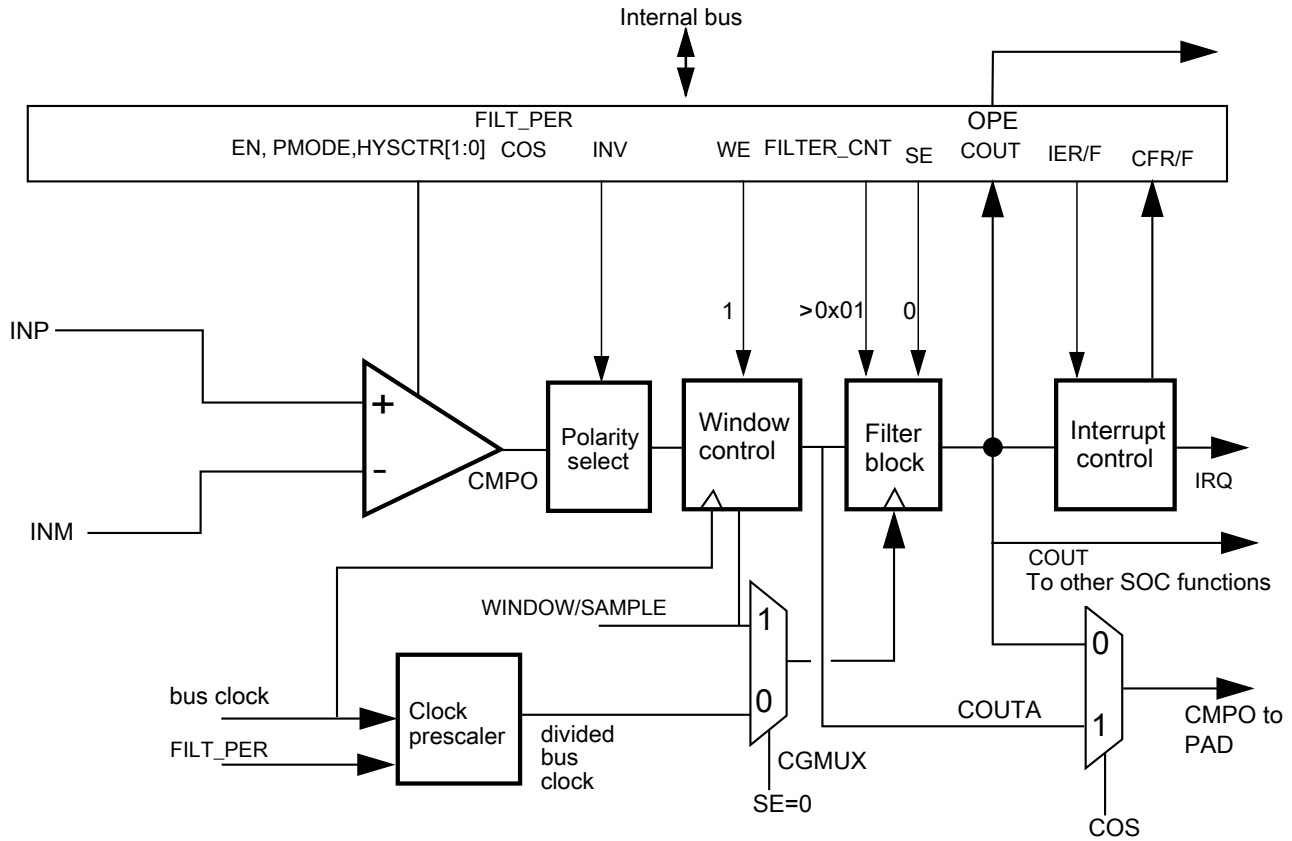


Figure 23-12. Windowed/Filtered mode

23.4.2 Power modes

23.4.2.1 Wait mode operation

During Wait and VLPW modes, the CMP, if enabled, continues to operate normally and a CMP interrupt can wake the MCU.

23.4.2.2 Stop mode operation

Depending on clock restrictions related to the MCU core or core peripherals, the MCU is brought out of stop when a compare event occurs and the corresponding interrupt is enabled. Similarly, if CR1[OPE] is enabled, the comparator output operates as in the normal operating mode and comparator output is placed onto the external pin. In Stop modes, the comparator can be operational in both:

- High-Speed (HS) Comparison mode when CR1[PMODE] = 1
- Low-Speed (LS) Comparison mode when CR1[PMODE] = 0

It is recommended to use the LS mode to minimize power consumption.

If stop is exited with a reset, all comparator registers are put into their reset state.

23.4.3 Startup and operation

A typical startup sequence is listed here.

- The time required to stabilize COUT will be the power-on delay of the comparators plus the largest propagation delay from a selected analog source through the analog comparator, windowing function and filter. See the Data Sheets for power-on delays of the comparators. The windowing function has a maximum of one bus clock period delay. The filter delay is specified in the Low-pass filter.
- During operation, the propagation delay of the selected data paths must always be considered. It may take many bus clock cycles for COUT and SCR[CFR]/SCR[CFF] to reflect an input change or a configuration change to one of the components involved in the data path.
- When programmed for filtering modes, COUT will initially be equal to 0, until sufficient clock cycles have elapsed to fill all stages of the filter. This occurs even if COUTA is at a logic 1.

23.4.4 Low-pass filter

The low-pass filter operates on the unfiltered and unsynchronized and optionally inverted comparator output COUTA and generates the filtered and synchronized output COUT.

Both COUTA and COUT can be configured as module outputs and are used for different purposes within the system.

Synchronization and edge detection are always used to determine status register bit values. They also apply to COUT for all sampling and windowed modes. Filtering can be performed using an internal timebase defined by FPR[FILT_PER], or using an external SAMPLE input to determine sample time.

The need for digital filtering and the amount of filtering is dependent on user requirements. Filtering can become more useful in the absence of an external hysteresis circuit. Without external hysteresis, high-frequency oscillations can be generated at COUTA when the selected INM and INP input voltages differ by less than the offset voltage of the differential comparator.

23.4.4.1 Enabling filter modes

Filter modes can be enabled by:

- Setting CR0[FILTER_CNT] > 0x01 and
- Setting FPR[FILT_PER] to a nonzero value or setting CR1[SE]=1

If using the divided bus clock to drive the filter, it will take samples of COUTA every FPR[FILT_PER] bus clock cycles.

The filter output will be at logic 0 when first initalized, and will subsequently change when all the consecutive CR0[FILTER_CNT] samples agree that the output value has changed. In other words, SCR[COUT] will be 0 for some initial period, even when COUTA is at logic 1.

Setting both CR1[SE] and FPR[FILT_PER] to 0 disables the filter and eliminates switching current associated with the filtering process.

Note

Always switch to this setting prior to making any changes in filter parameters. This resets the filter to a known state. Switching CR0[FILTER_CNT] on the fly without this intermediate step can result in unexpected behavior.

If CR1[SE]=1, the filter takes samples of COUTA on each positive transition of the sample input. The output state of the filter changes when all the consecutive CR0[FILTER_CNT] samples agree that the output value has changed.

23.4.4.2 Latency issues

The value of FPR[FILT_PER] or SAMPLE period must be set such that the sampling period is just longer than the period of the expected noise. This way a noise spike will corrupt only one sample. The value of CR0[FILTER_CNT] must be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the power of CR0[FILTER_CNT].

The values of FPR[FILT_PER] or SAMPLE period and CR0[FILTER_CNT] must also be traded off against the desire for minimal latency in recognizing actual comparator output transitions. The probability of detecting an actual output change within the nominal latency is the probability of a correct sample raised to the power of CR0[FILTER_CNT].

The following table summarizes maximum latency values for the various modes of operation *in the absence of noise*. Filtering latency is restarted each time an actual output transition is masked by noise.

Table 23-3. Comparator sample/filter maximum latencies

Mode #	CR1[EN]	CR1[WE]	CR1[SE]	CR0[FILTER_CNT]	FPR[FILT_PER]	Operation	Maximum latency[1]
1	0	X	X	X	X	Disabled	N/A
2A	1	0	0	0x00	X	Continuous Mode	T_{PD}
2B	1	0	0	X	0x00		
3A	1	0	1	0x01	X	Sampled, Non-Filtered mode	$T_{PD} + T_{SAMPLE} + T_{per}$
3B	1	0	0	0x01	> 0x00		T_{PD} + (FPR[FILT_PER] * T_{per}) + T_{per}
4A	1	0	1	> 0x01	X	Sampled, Filtered mode	T_{PD} + (CR0[FILTER_CNT] * T_{SAMPLE}) + T_{per}
4B	1	0	0	> 0x01	> 0x00		T_{PD} + (CR0[FILTER_CNT] * FPR[FILT_PER] x T_{per}) + T_{per}
5A	1	1	0	0x00	X	Windowed mode	$T_{PD} + T_{per}$
5B	1	1	0	X	0x00		$T_{PD} + T_{per}$
6	1	1	0	0x01	0x01 - 0xFF	Windowed / Resampled mode	T_{PD} + (FPR[FILT_PER] * T_{per}) + $2T_{per}$
7	1	1	0	> 0x01	0x01 - 0xFF	Windowed / Filtered mode	T_{PD} + (CR0[FILTER_CNT] * FPR[FILT_PER] x T_{per}) + $2T_{per}$

1. T_{PD} represents the intrinsic delay of the analog component plus the polarity select logic. T_{SAMPLE} is the clock period of the external sample clock. T_{per} is the period of the bus clock.

23.5 CMP interrupts

The CMP module is capable of generating an interrupt on either the rising- or falling-edge of the comparator output, or both.

The following table gives the conditions in which the interrupt request is asserted and deasserted.

When	Then
SCR[IER] and SCR[CFR] are set	The interrupt request is asserted
SCR[IEF] and SCR[CFF] are set	The interrupt request is asserted
SCR[IER] and SCR[CFR] are cleared for a rising-edge interrupt	The interrupt request is deasserted
SCR[IEF] and SCR[CFF] are cleared for a falling-edge interrupt	The interrupt request is deasserted

23.6 DMA support

Normally, the CMP generates a CPU interrupt if there is a change on the COUT. When DMA support is enabled by setting SCR[DMAEN] and the interrupt is enabled by setting SCR[IER], SCR[IEF], or both, the corresponding change on COUT forces a DMA transfer request rather than a CPU interrupt instead. When the DMA has completed the transfer, it sends a transfer completing indicator that deasserts the DMA transfer request and clears the flag to allow a subsequent change on comparator output to occur and force another DMA request.

The comparator can remain functional in STOP modes.

When DMA support is enabled by setting SCR[DMAEN] and the interrupt is enabled by setting SCR[IER], SCR[IEF], or both, the corresponding change on COUT forces a DMA transfer request to wake up the system from STOP modes. After the data transfer has finished, system will go back to STOP modes. Refer to DMA chapters in the device reference manual for the asynchronous DMA function for details.

23.7 Digital-to-analog converter

The figure found here shows the block diagram of the DAC module.

It contains a 256-tap resistor ladder network and a 256-to-1 multiplexer, which selects an output voltage from one of 256 distinct levels that outputs from DACO. It is controlled through the DAC Control Register (DACCR). Its supply reference source can be selected from two sources V_{in1} and V_{in2}. The module can be powered down or disabled when not in use. When in Disabled mode, DACO is connected to the analog ground.

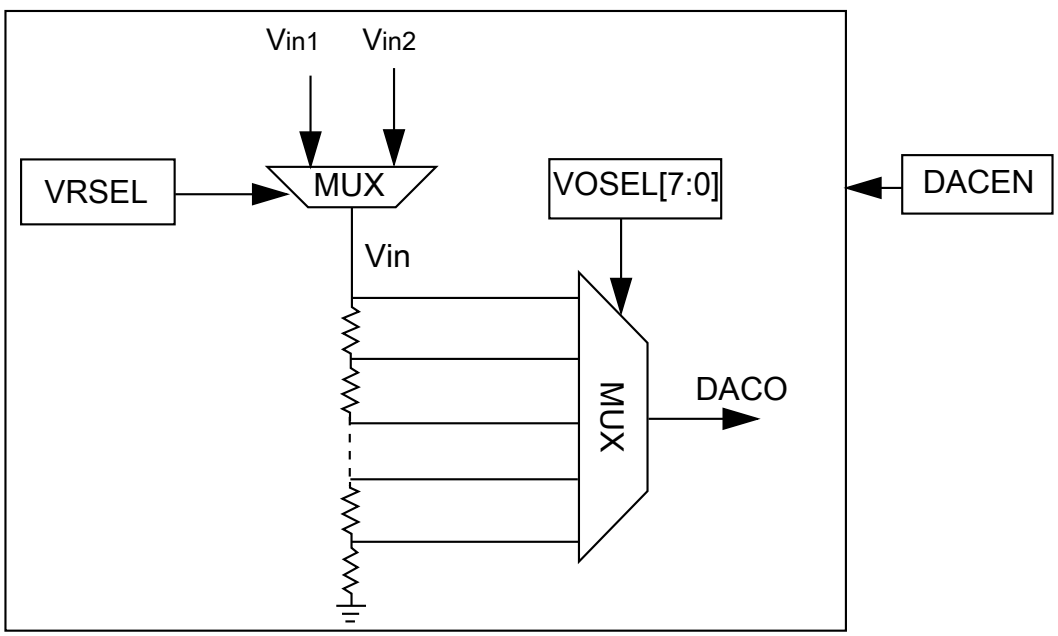


Figure 23-13. 6-bit DAC block diagram

23.8 DAC functional description

This section provides DAC functional description information.

23.8.1 Voltage reference source select

- V_{in1} connects to the primary voltage source as supply reference of 64 tap resistor ladder
- V_{in2} connects to an alternate voltage source

23.9 DAC resets

This module has a single reset input, corresponding to the chip-wide peripheral reset.

23.10 DAC clocks

This module has a single clock input, the bus clock.

23.11 DAC interrupts

This module has no interrupts.

Chapter 24
12-bit Digital-to-Analog Converter (DAC)

24.1 Introduction

24.1.1 Overview

The 12-bit digital-to-analog converter (DAC) provides a voltage reference to on-chip modules or an output to a package pin. It can also be used as a waveform generator to generate square, triangle, and sawtooth waveforms for various applications, such as slope compensation in cycle-by-cycle control. The DAC can be put in power-down mode if needed.

24.1.2 Features

DAC features include:

- 12-bit resolution
- Power-down mode
- Asynchronous or synchronous updates by on-chip peripherals or off-chip circuitries
- Automatic mode for waveform generation:
  - square, triangle, and sawtooth output waveforms
  - programmable period, update rate, and swing range
- DMA support with configurable watermark level
- High-speed/low-speed mode selection
- Support of two digital formats: right-justified or left-justified
- Glitch filter to suppress output glitch during data conversion

24.1.3 Block Diagram

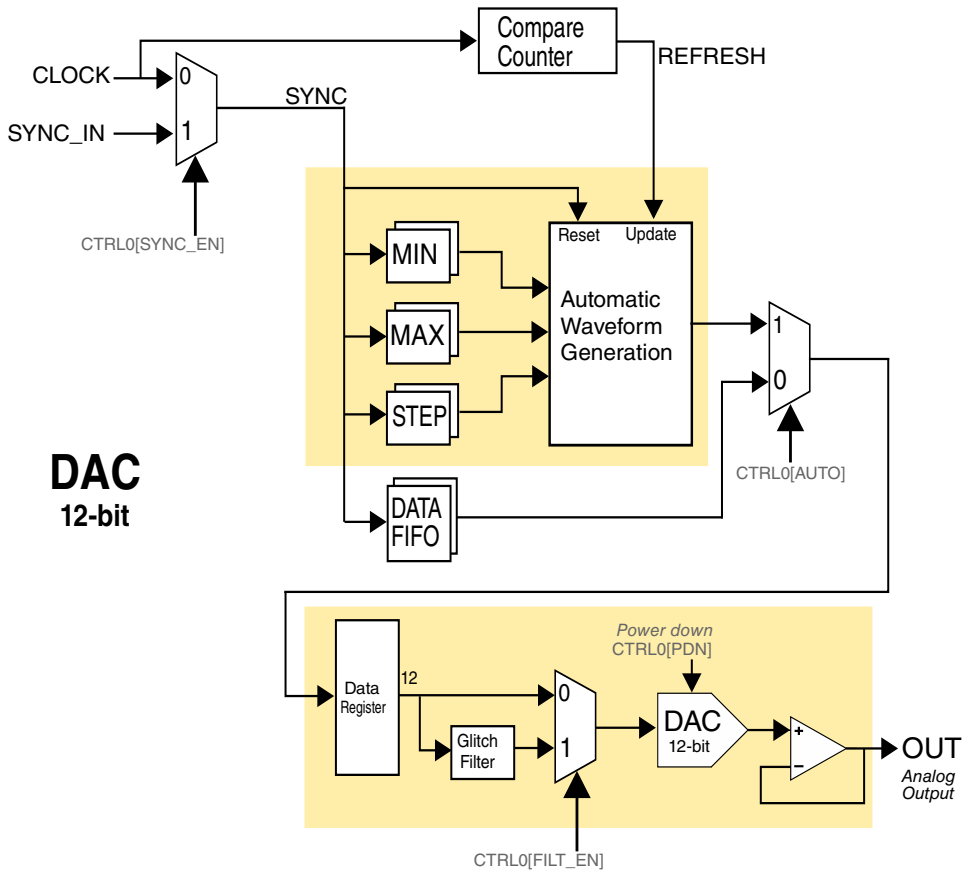


Figure 24-1. DAC Block Diagram

24.2 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. Only 16-bit accesses are supported; no 8-bit or 32-bit accesses can be done.

DAC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E000	Control Register 0 (DAC_CTRL0)	16	R/W	1101h	24.2.1/655

Table continues on the next page...

DAC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E001	Buffered Data Register (DAC_DATAREG_FMT0)	16	R/W	0000h	24.2.2/658
E001	Buffered Data Register (DAC_DATAREG_FMT1)	16	R/W	0000h	24.2.3/659
E002	Step Size Register (DAC_STEPVAL_FMT0)	16	R/W	0000h	24.2.4/659
E002	Step Size Register (DAC_STEPVAL_FMT1)	16	R/W	0000h	24.2.5/660
E003	Minimum Value Register (DAC_MINVAL_FMT0)	16	R/W	0000h	24.2.6/660
E003	Minimum Value Register (DAC_MINVAL_FMT1)	16	R/W	0000h	24.2.7/661
E004	Maximum Value Register (DAC_MAXVAL_FMT0)	16	R/W	FFFFh	24.2.8/661
E004	Maximum Value Register (DAC_MAXVAL_FMT1)	16	R/W	FFFFh	24.2.9/662
E005	Status Register (DAC_STATUS)	16	R	0001h	24.2.10/663
E006	Control Register 1 (DAC_CTRL1)	16	R/W	001Dh	24.2.11/663
E007	Compare Register (DAC_COMPARE)	16	R/W	0000h	24.2.12/664

24.2.1 Control Register 0 (DAC_CTRL0)

Address: E000h base + 0h offset = E000h

Bit	15	14	13	12	11	10	9	8
Read Write	ONESHOT	LDOK	0	FILT_EN	SYNCEDGE		WTMK_LVL	
Reset	0	0	0	1	0	0	0	1

Bit	7	6	5	4	3	2	1	0
Read Write	DMA_EN	HSLS	UP	DOWN	AUTO	SYNC_EN	FORMAT	PDN
Reset	0	0	0	0	0	0	0	1

DAC_CTRL0 field descriptions

Field	Description
15 ONESHOT	One shot Determines whether automatic waveform generation creates one waveform or a repeated waveform within the period defined by the active SYNC edges. ONESHOT bit has no function when CTRL0[AUTO] is cleared. **NOTE:** If ONESHOT=1, setting both UP and DOWN to 1 generates unpredictable output. 0 Automatic waveform generation logic will create a repeated (continuous) waveform upon receiving an active SYNC edge, otherwise the waveform repeats when it reaches its MIN or MAX value. 1 Automatic waveform generation logic will create a single pattern and stop at the final value. It will remain at this final value until a new active edge occurs on the SYNC input, and then the waveform will be repeated.
14 LDOK	Load Okay

Table continues on the next page...

DAC_CTRL0 field descriptions (continued)

Field	Description
	Allows new values of MINVAL, MAXVAL, and STEPVAL to be updated by active edge of SYNC_IN. This bit should be set once new values of these buffered registers have been written by software. This bit is cleared by an active edge of SYNC_IN. LDOK bit has no function when CTRL0[AUTO] is cleared. 0 Buffered values of STEPVAL, MINVAL, and MAXVAL will not be updated and the existing values will be reused. 1 Buffered values of STEPVAL, MINVAL, and MAXVAL will be updated and used at active edge of SYNC_IN.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 FILT_EN	Glitch Filter Enable Enables the glitch suppression filter. This introduces a latency equivalent to FILT_CNT clock cycles for DAC output updates. It should never be low. 0 Disable glitch filter 1 Enable glitch filter
11–10 SYNCEDGE	Sync edge Selects which SYNC input edge is used for updates. 00 No active edge is selected, therefore the SYNC input is ignored 01 Updates occur on the falling edge of the SYNC input 10 Updates occur on the rising edge of the SYNC input 11 Updates occur on both edges of the SYNC input
9–8 WTMK_LVL	Watermark Level When the level of FIFO is less than or equal to the Watermark Level field, a DMA request should be sent. The FIFO generates a watermark signal depending on the value of WTMK_LVL, which is used for asserting a DMA request. 00 Watermark value is 0 01 Watermark value is 2 (default) 10 Watermark value is 4 11 Watermark value is 6
7 DMA_EN	Enable DMA Support Enables DMA requests to be generated when the FIFO is below the level set by the WTMK_LVL (watermark level) field. **Restriction:** Be careful when setting DMA_EN while SYNC_EN is 0. The buffered registers will be updated every clock cycle and the DMA may not be able to keep up unless the clock is very slow. 0 Disable DMA support (default) 1 Enable DMA support
6 HSLS	High/Low Speed Selects between speed and power. • Setting HSLS low selects high speed mode, which makes the settling time of the DAC = 1 µs (faster response), but the DAC uses more power. • Setting HSLS high selects low speed mode, which saves power but the DAC takes more time to settle.

Table continues on the next page...

DAC_CTRL0 field descriptions (continued)

Field	Description
	0 High speed mode (default) 1 Low speed mode
5 UP	Enable Up-Counting Enables counting up in automatic mode. Also see Automatic Mode, to understand how the UP bit affects automatic waveform generation. 0 Disable up-counting 1 Enable up-counting
4 DOWN	Enable Down Counting Enables counting down in automatic mode. Also see Automatic Mode, to understand how the DOWN bit affects automatic waveform generation. 0 Disable down-counting 1 Enable down-counting
3 AUTO	Automatic Mode Enables automatic waveform generation mode. In automatic mode, an external source (typically a timer module) that is driving SYNC_IN determines the buffered registers update rate, while the STEP, MINVAL, and MAXVAL registers and the UP and DOWN bits along with the REFRESH signal are used to shape the waveform. If the SYNC_EN bit is not set when using automatic mode, then the data for the analog DAC will be updated every clock cycle, but the DAC output may be unable to keep up with this update rate. **Restriction:** Be careful when setting AUTO while SYNC_EN is low as values from the buffered registers will be updated every clock cycle. 0 Normal mode. Automatic waveform generation disabled. 1 Automatic waveform generation enabled.
2 SYNC_EN	Sync Enable Enables the SYNC_IN input to be used to trigger an update of the buffered registers and also a reset of the waveform generation logic with the updated register values. If SYNC_EN is cleared, then asynchronous mode is selected, and the data that is written to the buffered registers is used on the following clock cycle. **Restriction:** Be careful when clearing SYNC_EN while DMA_EN is 1. The buffered registers will be updated every clock cycle and the DMA may not be able to keep up unless the clock is very slow. 0 Asynchronous mode. Data written to the buffered registers is used on the next clock cycle. 1 Synchronous mode. SYNC_IN signal updates data in the buffered registers.
1 FORMAT	Data Format Selects which data format that the DAC uses. The DAC can use one of 2 data formats. • When FORMAT = 0, then the 12 bits of data are right-justified within the 16-bit data register. • When FORMAT = 1, then the 12 bits of data are left-justified within the 16-bit data register. In either case, the 4 unused bits are ignored. To select the justification of the affected registers, FORMAT selects which one of the following pairs of registers is used: • Buffered Data Register:

Table continues on the next page...

DAC_CTRL0 field descriptions (continued)

Field	Description
	• If FORMAT = 0, then DAC_DATAREG_FMT0 is used. • If FORMAT = 1, then DAC_DATAREG_FMT1 is used. • Step Size Register: • If FORMAT = 0, then DAC_STEPVAL_FMT0 is used. • If FORMAT = 1, then DAC_STEPVAL_FMT1 is used. • Minimum Value Register: • If FORMAT = 0, then DAC_MINVAL_FMT0 is used. • If FORMAT = 1, then DAC_MINVAL_FMT1 is used. • Maximum Value Register: • If FORMAT = 0, then DAC_MAXVAL_FMT0 is used. • If FORMAT = 1, then DAC_MAXVAL_FMT1 is used. 0 Data words are right-justified (default) 1 Data words are left-justified
0 PDN	Power Down Powers down the analog portion of the DAC (resulting in its output being pulled low) when the DAC is not in use. The Power Down bit does not reset the registers; after clearing PDN, the analog DAC will output the value currently presented to its inputs. The analog block requires 12 µs to recover from the power-down state before proper operation is guaranteed. 0 DAC is operational. 1 DAC is powered down. (default)

24.2.2 Buffered Data Register (DAC_DATAREG_FMT0)

Address: E000h base + 1h offset = E001h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0				DATA											
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DAC_DATAREG_FMT0 field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
DATA	DAC data (right-justified) Data written to this register is held in a FIFO. Assuming CTRL0[AUTO]==0 the digital data contained in this buffer is presented to the analog DAC upon the active edge of the SYNC_IN signal (or at the next clock cycle if CTRL0[SYNC_EN] is clear) and converted to analog and output by the DAC. Reading this register returns the value being presented to the analog DAC, which may differ from the data value being held in the FIFO. The data in this buffer can be updated at any rate, but the DAC output load impedance may affect the updating rate . When a read occurs at the address of this register, the data read is the value of the data presented to the analog DAC at that time, not the value in this register.

24.2.3 Buffered Data Register (DAC_DATAREG_FMT1)

Address: E000h base + 1h offset = E001h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DATA												0			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DAC_DATAREG_FMT1 field descriptions

Field	Description
15–4 DATA	DAC data (left-justified) Data written to this register is held in a FIFO. Assuming CTRL0[AUTO]==0 the digital data contained in this buffer is presented to the analog DAC upon the active edge of the SYNC_IN signal (or at the next clock cycle if CTRL0[SYNC_EN] is clear) and converted to analog and output by the DAC. Reading this register returns the value being presented to the analog DAC, which may differ from the data value being held in the FIFO. The data in this buffer can be updated at any rate, but the DAC output load impedance may affect the updating rate . When a read occurs at the address of this register, the data read is the value of the data presented to the analog DAC at that time, not the value in this register.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

24.2.4 Step Size Register (DAC_STEPVAL_FMT0)

Address: E000h base + 2h offset = E002h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0				STEP											
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DAC_STEPVAL_FMT0 field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
STEP	STEP size (right-justified) When the DAC is in automatic mode (CTRL0[AUTO] = 1), the step size contained in this register will be added to or subtracted from the current value, to create the next value presented to the DAC inputs. This Step Size register is not used during normal mode operation, but can still be written to and read from. Values written to this Step Size register are buffered, and are updated based on CTRL0[SYNC_EN].

24.2.5 Step Size Register (DAC_STEPVAL_FMT1)

Address: E000h base + 2h offset = E002h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	STEP												0			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DAC_STEPVAL_FMT1 field descriptions

Field	Description
15–4 STEP	STEP size (left-justified) When the DAC is in automatic mode (CTRL0[AUTO] = 1), the step size contained in this register will be added to or subtracted from the current value, to create the next value presented to the DAC inputs. This Step Size register is not used during normal mode operation, but can still be written to and read from. Values written to this Step Size register are buffered, and are updated based on CTRL0[SYNC_EN].
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

24.2.6 Minimum Value Register (DAC_MINVAL_FMT0)

Address: E000h base + 3h offset = E003h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0				MINVAL											
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DAC_MINVAL_FMT0 field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
MINVAL	Minimum value (right-justified) When the DAC is in automatic mode (CTRL0[AUTO] = 1), the minimum value contained in this register acts as the lower range limit during automatic waveform generation. This Minimum Value register is not used during normal mode operation, but can still be written to and read from. Refer to the device data sheet for limitations on the low end voltage output of the DAC. Values written to this Minimum Value register are buffered, and are updated based on CTRL0[SYNC_EN]. **NOTE:** If DAC input data is less than MINVAL, output is limited to MINVAL during automatic waveform generation. If ONESHOT=1, when DOWN=1 and UP=0, MINVAL should be set to "$MINVAL_{target}$ (the target Minimum value) – one STEP_value", where (MAXVAL – $MINVAL_{target}$) /step = integer. For example, for desired ramp down output from 2048 to 512, the configuration shall be: ONESHOT=1, DOWN=1, UP=0, STEPVAL=512, MAXVAL=2048, **MINVAL**=0.

24.2.7 Minimum Value Register (DAC_MINVAL_FMT1)

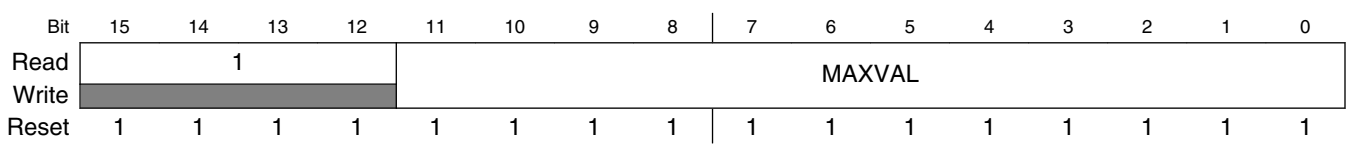
Address: E000h base + 3h offset = E003h

DAC_MINVAL_FMT1 field descriptions

Field	Description
15–4 MINVAL	Minimum value (left-justified) When the DAC is in automatic mode (CTRL0[AUTO] = 1), the minimum value contained in this register acts as the lower range limit during automatic waveform generation. This Minimum Value register is not used during normal mode operation, but can still be written to and read from. Refer to the device data sheet for limitations on the low end voltage output of the DAC. Values written to this Minimum Value register are buffered, and are updated based on CTRL0[SYNC_EN]. **NOTE:** If DAC input data is less than MINVAL, output is limited to MINVAL during automatic waveform generation. If ONESHOT=1, when DOWN=1 and UP=0, MINVAL should be set to "$MINVAL_{target}$ (the target Minimum value) – one STEP_value", where (MAXVAL – $MINVAL_{target}$) /step = integer. For example, for desired ramp down output from 2048 to 512, the configuration shall be: ONESHOT=1, DOWN=1, UP=0, STEPVAL=512, MAXVAL=2048, **MINVAL**=0.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

24.2.8 Maximum Value Register (DAC_MAXVAL_FMT0)

Address: E000h base + 4h offset = E004h



DAC_MAXVAL_FMT0 field descriptions

Field	Description
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
MAXVAL	Maximum value (right-justified) When the DAC is in automatic mode (CTRL0[AUTO] = 1), the maximum value contained in this register acts as the upper range limit during automatic waveform generation. This Maximum Value register is not used during normal mode operation, but can still be written to and read from. Refer to the device data

Table continues on the next page...

DAC_MAXVAL_FMT0 field descriptions (continued)

Field	Description
	sheet for limitations on the high-end voltage output of the DAC. Values written to this Maximum Value register are buffered, and are updated based on CTRL0[SYNC_EN]. **NOTE:** If DAC input data is greater than MAXVAL, output is limited to MAXVAL during automatic waveform generation. If ONESHOT=1, when UP=1 and DOWN=0, MAXVAL should be set to "$MAXVAL_{target}$ (the target Maximum value) + one STEP_value", where ($MAXVAL_{target}$ – MINVAL) /step = integer. For example, for desired ramp up output from 512 to 2048, the configuration shall be: ONESHOT=1, UP=1, DOWN=0, STEPVAL=512, **MAXVAL**=2560, MINVAL=512.

24.2.9 Maximum Value Register (DAC_MAXVAL_FMT1)

Address: E000h base + 4h offset = E004h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	MAXVAL												1			
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

DAC_MAXVAL_FMT1 field descriptions

Field	Description
15–4 MAXVAL	Maximum value (left-justified) When the DAC is in automatic mode (CTRL0[AUTO] = 1), the maximum value contained in this register acts as the upper range limit during automatic waveform generation. This Maximum Value register is not used during normal mode operation, but can still be written to and read from. Refer to the device data sheet for limitations on the high-end voltage output of the DAC. Values written to this Maximum Value register are buffered, and are updated based on CTRL0[SYNC_EN]. **NOTE:** If DAC input data is greater than MAXVAL, output is limited to MAXVAL during automatic waveform generation. If ONESHOT=1, when UP=1 and DOWN=0, MAXVAL should be set to "$MAXVAL_{target}$ (the target Maximum value) + one STEP_value", where ($MAXVAL_{target}$ – MINVAL) /step = integer. For example, for desired ramp up output from 512 to 2048, the configuration shall be: ONESHOT=1, UP=1, DOWN=0, STEPVAL=512, **MAXVAL**=2560, MINVAL=512.
Reserved	This field is reserved. This read-only field is reserved and always has the value 1.

24.2.10 Status Register (DAC_STATUS)

Address: E000h base + 5h offset = E005h

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

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

DAC_STATUS field descriptions

Field	Description
15–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FULL	Indicates that the FIFO is full 0 FIFO is not full (on reset). 1 FIFO is full.
0 EMPTY	Indicates that the FIFO is empty 0 FIFO is not empty 1 FIFO is empty (on reset)

24.2.11 Control Register 1 (DAC_CTRL1)

Address: E000h base + 6h offset = E006h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0				Reserved				0		FILT_CNT					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1

DAC_CTRL1 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 Reserved	This field is reserved. Do not write to this bitfield.
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...

DAC_CTRL1 field descriptions (continued)

Field	Description
FILT_CNT	Glitch Filter Count The Glitch Filter Count field represents the number of clock cycles for which the DAC output is held unchanged after new data is presented to the analog DAC's inputs. The number of clock cycles for which DAC output is held unchanged is equal to the value of FILT_CNT. Approximately 240 ns is needed for worst-case settling of the DAC output, so a value of 29 should be used for 125 MHz operation, a value of 33 for 140 MHz operation, and a value of 12 for 50 MHz operation. The default value is 29 (can be used for 125 MHz operation). **NOTE:** When using the glitch filter, make sure that the filter count is less than the update count. If the filter count is not less than the update count (i.e., the filter count is greater than or equal to the update count), then the DAC output will never be updated.

24.2.12 Compare Register (DAC_COMPARE)

Address: E000h base + 7h offset = E007h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	COMPARE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DAC_COMPARE field descriptions

Field	Description
COMPARE	Compare value In automatic mode, the COMPARE value selects when the automatically generated waveform value is updated. • COMPARE= 0: the generated waveform will be updated every clock cycle. • COMPARE= *N*: the generated waveform will be updated every *N* + 1 clock cycles.

24.3 Functional Description

24.3.1 Conversion modes

This DAC supports two conversion modes: asynchronous conversion and synchronous conversion.

24.3.1.1 Asynchronous conversion mode

When data is written to the DAC's buffered data register, the data can be immediately presented to the DAC and converted to an analog output.

24.3.1.2 Synchronous conversion mode

The SYNC_IN signal controls when the values of the buffered registers are updated. The update occurs on the active edge of the SYNC_IN signal if CTRL0[LDOK] is set. The active edge of SYNC_IN will also cause the automatic waveform to be reset to its start point as defined by the new MAXVAL and MINVAL values. The SYNC_IN signal can come from a timer, comparator, pins, or other sources. The CPU must update the buffered registers and set CTRL0[LDOK] before the next SYNC_IN active edge. If the CPU does not update the buffered registers or set CTRL0[LDOK] before the next SYNC_IN active edge, then the old buffered value is reused.

NOTE

The SYNC_IN signal must be high for at least 1 clock cycle, and the SYNC_IN signal must be low for at least 1 clock cycle.

24.3.2 Operation Modes

The DAC module operates in either Normal or Automatic mode.

- Normal mode: CTRL0[AUTO]==0, the DAC module generates an analog representation of digital words.
- Automatic mode: CTRL0[AUTO]==1, the DAC module generates sawtooth, triangle, and square waveforms without requiring CPU or core assistance.

24.3.2.1 Normal Mode

The DAC receives data words through a memory-mapped register on the IPBus (DATA) either via software writes or via DMA writes. A digital word is applied to the DAC inputs, based on CTRL0[SYNC_EN]. When using DMA to supply the digital values to be converted, software should set CTRL0[WTMK_LVL] to determine when the DMA write request is generated in order to ensure that new data is always present at the active SYNC edge. In the worst case with no DAC output load, approximately 240 ns settling time is needed.

24.3.2.2 Automatic Mode

In Automatic mode, the DAC generates sawtooth, triangle, and square wave waveforms without CPU or core assistance. The update rate, incremental step size, and minimum and maximum values are programmable.

The value in the DATA register is used as a starting-point for the following process:

1. The SYNC_IN input indicates that it is time to update the buffered register values.
2. The REFRESH signal controlled by the COMPARE value determines when the following steps take place.
3. The STEP value is added to/subtracted from the current DATA value and then the DATA is updated.
4. If CTRL0[UP] is set, then STEP is added to DATA each update, until MAXVAL is reached.
5. The generator starts subtracting STEP from DATA if CTRL0[DOWN] is set (down counting enabled) or starts reloading MINVAL if CTRL0[DOWN] is clear (no down counting).
6. When DATA reaches MINVAL while counting down, the generator starts counting up if CTRL0[UP] is set or starts reloading MAXVAL if CTRL0[UP] is clear (up counting disabled).
7. If CTRL0[ONESHOT] is set, then the automatic waveform generation is stopped after the full waveform has been generated. The DAC will hold its last value until the next active edge of SYNC_IN arrives.

The initial count direction depends on which bit (CTRL0[UP] or CTRL0[DOWN]) is set (written to 1) last. See the examples below in triangle waveform generation:

- If CTRL0[DOWN] is set first and then CTRL0[UP] is set, the DAC output starts to ramp up and then ramp down.
- If CTRL0[UP] is set first and then CTRL0[DOWN] is set, the DAC output starts to ramp down and then ramp up.
- If both CTRL0[UP] and CTRL0[DOWN] are set simultaneously, the DAC output starts to ramp up and then ramp down.

The next figures show examples of automatically-generated waveforms. The waveforms shown are idealized (perfect); actual waveforms are limited by the slew rate of the DAC output.

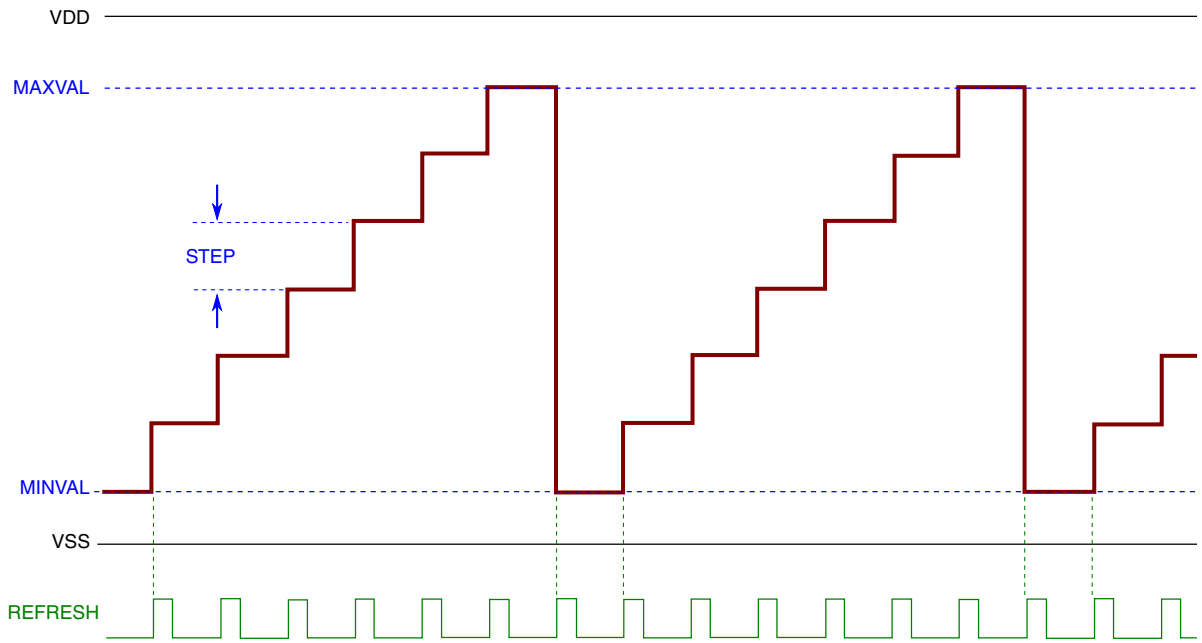


Figure 24-2. Sawtooth Waveform Example with CTRL0[UP]=1 and CTRL0[DOWN]=0

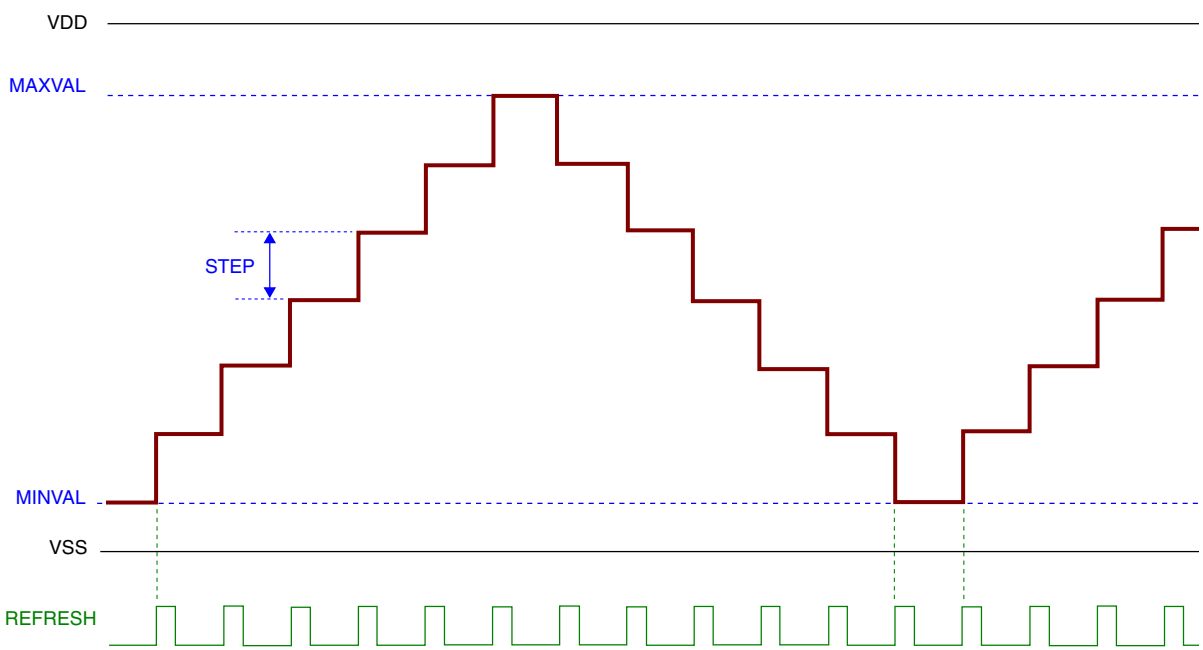


Figure 24-3. Triangle Waveform Example with CTRL0[UP]=1 and CTRL0[DOWN]=1

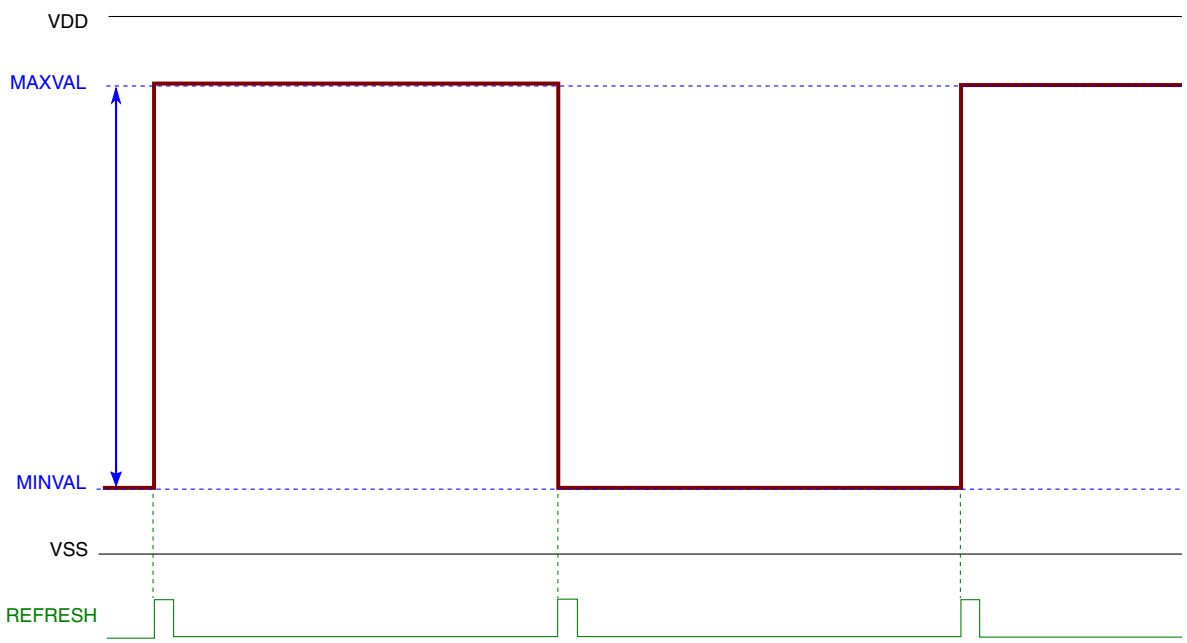


Figure 24-4. Square Waveform Example with CTRL0[UP]=1 and CTRL0[DOWN]=1

These waveform examples show that the waveform period is a function of the difference between MAXVAL and MINVAL, the STEP size, and the update rate:

$$\text{Period} = \left[\frac{\text{MAXVAL} - \text{MINVAL}}{\text{STEP}} \times \text{UpdatePeriod}\right] \times 2$$

Increasing STEP decreases the resolution of the output steps. Increasing the update rate decreases the waveform period. Varying MINVAL and MAXVAL changes the DC offset and the amplitude of the waveform.

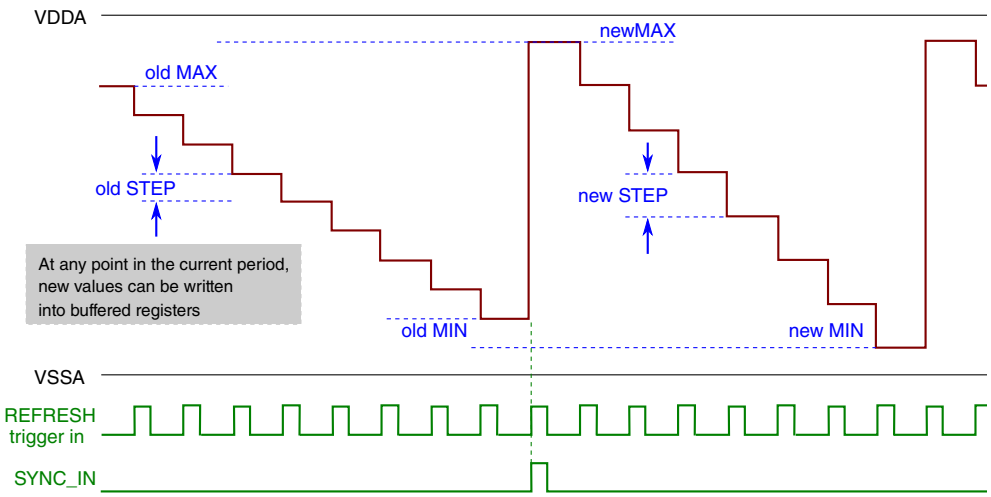


Figure 24-5. Buffered values are updated using the SYNC_IN signal

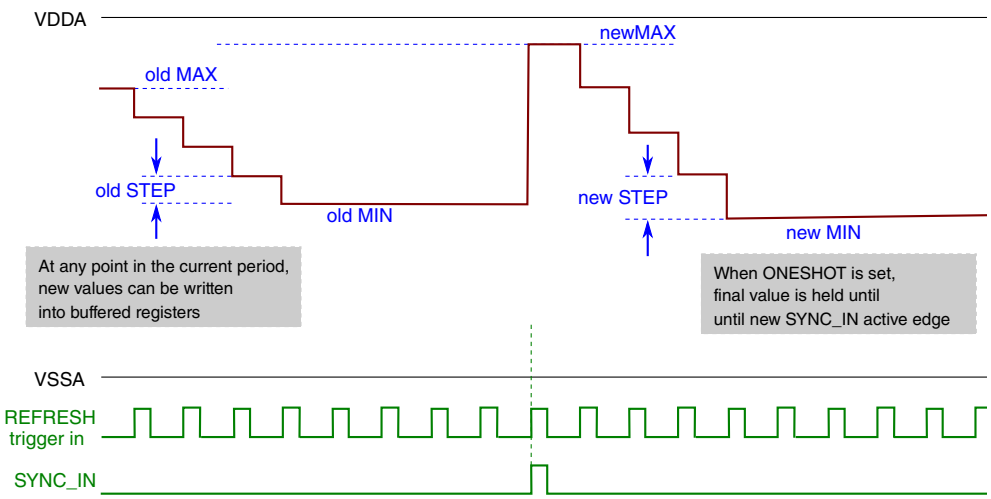


Figure 24-6. ONESHOT waveforms hold final value until active SYNC_IN edge

24.3.3 DAC settling time

Settling time is the interval (within a specified percentage error band) between the time that data is presented to the DAC input to update (change) and the time that its analog output value reaches its final value. Settling time is affected by:

- the circuit propagation delay
- the slew rate of the DAC output capability
- the real load on DAC output

Approximately 240 ns settling time, which is caused by the DAC conversion glitch, is needed in the worst case situations when the DAC output is internally sent to other modules, such as comparator inputs. If the DAC output is to a package pin, then the additional settling time is affected by the slew rate of an output amplifier and the load on the pin. The maximum settling time will not exceed 2 usec with a maximum output load (3 kohm || 400 pf) when the output swings from min output to max output, or swings from max output to min output.

24.3.4 Waveform Programming Example

To create a waveform that decreases from 3.0 V to 1.5 V in 1 millisecond, first calculate MAXVAL and MINVAL. Assuming that the DAC reference is 3.3 V, each DAC LSB represents 0.806 mV, and so MAXVAL = 0xE8A and MINVAL = 0x745. These numbers represent a difference of 1861 LSBs to be accomplished in 1 msec, so we can safely update the DAC 500 times, because the DAC has a maximum 2-usec settling time. Programming calculations:

- To go from MAXVAL to MINVAL in 500 steps, STEPVAL_FMT*n*[STEP] must be 0x004, after rounding the result of the 1861/500 calculation.
- To go from MAXVAL to MINVAL with a STEP size of 0x004 requires 465.25 steps.
- To keep the waveform period of 1 msec using 465.25 updates requires an update period of 465.25 kHz.
- If the system clock operates at 100 MHz, then the COMPARE register will update the waveform generation every 100000000/465250 = 215 counts.

When the COMPARE register (and the MAXVAL, MINVAL, and STEPVAL registers) are programmed , the DATA register is written with a value equal to MAXVAL as a starting point, because the DAC is only down-counting. When writing to the DATA, STEP, MAXVAL, and MINVAL registers, ensure that FORMAT has the desired value and that the data values are properly justified to match FORMAT. The SYNC_EN, DOWN, and AUTO bits of the CTRL register should be set; the CTRL0[UP] and CTRL0[PDN] bits should be cleared. To suppress glitches on the output, set CTRL1[FILT_CNT] and CTRL0[FILT_EN]. The desired waveform starts within 12 usec from the clearing of CTRL0[PDN], and the waveform continues until CTRL0[PDN] is set or until the timer is stopped.

24.3.5 Sources of Waveform Distortion

24.3.5.1 Switching Glitches

When a new digital value is presented to the DAC input, some glitches may appear on the output as the new values propagate through the circuitry. Eventually, these glitches settle out and the output slews to its new value. To avoid these glitches, set CTRL0[FILT_EN] = 1 and give CTRL1[FILT_CNT] a suitable value to cause the DAC to hold its current output for the number of clock cycles specified in the description of the FILT_CNT field, during which time the switching glitches settle out. After the filter time is satisfied, the output smoothly slews to the new value.

24.3.5.2 Slew Effects

The example waveforms are ideal waveforms and show transitions as step functions. In reality, the DAC output has a finite slew rate that rounds off the steps. Whether this rounding off is noticeable depends on the output step size (larger output changes require longer settling times) and on the update period (longer dwell times make the settling times less noticeable).

24.3.5.3 Clipping Effects (Automatic Mode Only)

One form of waveform clipping occurs during automatic waveform generation, when the difference between MAXVAL and MINVAL is not an even multiple (or close to an even multiple) of the STEP value. This causes a partial step as the waveform approaches MAXVAL and MINVAL limits. The next figure shows an example of DAC waveform clipping.

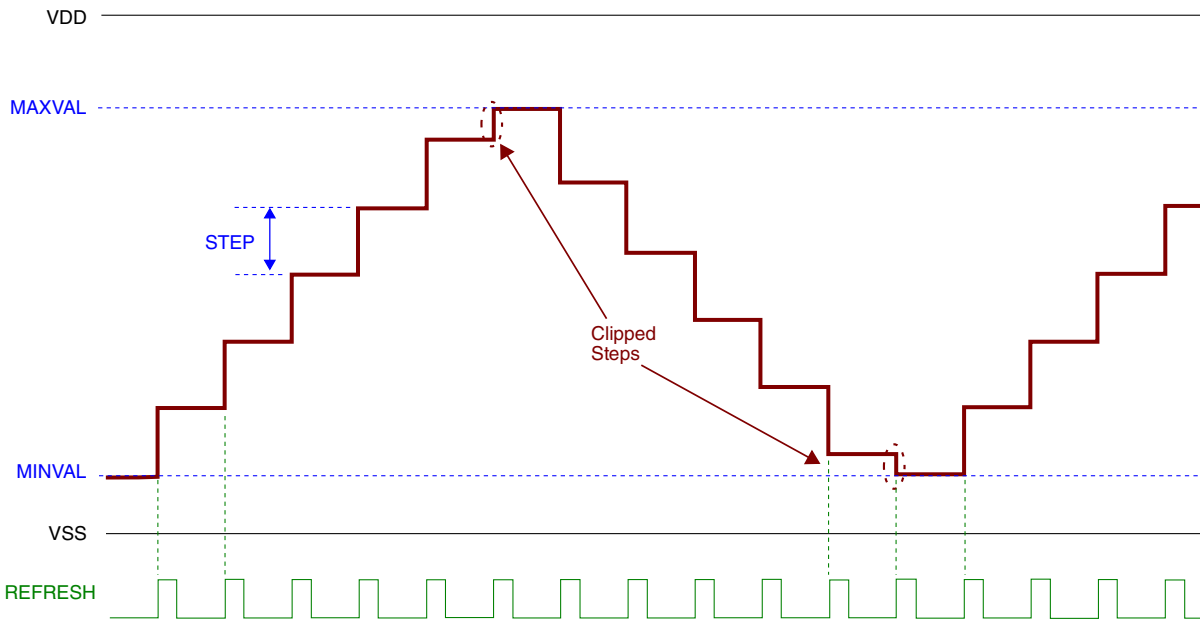


Figure 24-7. Triangle Waveform Example with Clipping

Another form of waveform clipping occurs when the MAXVAL or MINVAL is beyond of the output range of the DAC. The device's data sheet defines the maximum and minimum voltages that the DAC module can successfully drive.

24.4 Resets

When reset, all of the registers return to the reset state.

24.5 Clocks

The DAC uses the system bus clock (IPBus clock).

24.6 Interrupts

The DAC module does not generate any interrupts.

Chapter 25
Operational Amplifier (OPAMP)

25.1 Chip-specific information for this module

25.1.1 OPAMP configuration information

Beside the CTRL[EN] field is set, OPAMP is enabled functioning in the chip when the peripheral clocking is set (in SIM_PCE3[OPAMPx]) and the corresponding pin is configured as OPAMP output in the SIM module, as well as the related GPIO_PER register field is set, as below. Otherwise, OPAMP cannot work properly.

- SIM_GPSCL[C4] must be 11b, and bit 4 of GPIOC_PER must be 1b to enable OPAMPA.
- SIM_GPSFL[F0] must be 11b, and bit 0 of GPIOF_PER must be 1b to enable OPAMPB.

25.1.2 OPAMP channel assignment

The following table lists the OPAMP channel assignment and signal connection.

Table 25-1. OPAMP module signal connection

Module	Input/Output	Channel	Connect to
OPAMPA	Non Inverting Input	VPOS0	GPIOA1 (OPAMPA_IN0)
		VPOS1	GPIOA6 (OPAMPA_IN1)
		VPOS2	12-bit DACA
		VPOS3	GPIOA0 (OPAMPA_IN3)
	Inverting Input	VNEG0	GPIOA1 (OPAMPA_IN0)
		VNEG1	GPIOA6 (OPAMPA_IN1)
		VNEG2	GPIOA7 (OPAMPA_IN2)
		VNEG3	GPIOA0 (OPAMPA_IN3)

Table continues on the next page...

Table 25-1. OPAMP module signal connection (continued)

	Input from other on-chip modules	CFGSELA0	AUXA_SEL0 (in ADCA) [1]
		CFGSELB0	XBAR_OUT34 [2]
		CFGSELC0	XBAR_OUT34
		CFGSELA1	AUXA_SEL1 (in ADCA)
		CFGSELB1	XBAR_OUT14
		CFGSELC1	XBAR_OUT14
	Output	Package pin [3]	GPIOC4 (OPAMPA_OUT)
		Internally connected to other on-chip modules [4]	• Ch17 in ADCA • CMPA CH6 • CMPC CH6
OPAMPB	Non Inverting Input	VPOS0	GPIOB1 (OPAMPB_IN0)
		VPOS1	GPIOB6 (OPAMPB_IN1)
		VPOS2	12-bit DACA
		VPOS3	GPIOB0 (OPAMPB_IN3)
	Inverting Input	VNEG0	GPIOB1 (OPAMPB_IN0)
		VNEG1	GPIOB6 (OPAMPB_IN1)
		VNEG2	GPIOB7 (OPAMPB_IN2)
		VNEG3	GPIOB0 (OPAMPB_IN3)
	Input from other on-chip modules	CFGSELA0	AUXB_SEL0 (in ADCB)
		CFGSELB0	XBAR_OUT35
		CFGSELC0	XBAR_OUT35
		CFGSELA1	AUXB_SEL1 (in ADCB)
		CFGSELB1	XBAR_OUT14
		CFGSELC1	XBAR_OUT14
	Output	Package Pin [3]	GPIOF0 (OPAMPB_OUT)
		Internally connected to other on-chip modules [4]	• Ch19 in ADCB • CMPB CH6 • CMPD CH6

1. See the "Expansion MUX" figure in the ADC chapter.
2. See the "XBARA Outputs" table in the XBAR chapter.
3. A dedicated pad that is directly connected pin to minimize the output impedance.
4. Besides output to dedicated pad, OPAMP output is also connected to ADC and CMP, see "OPAMP block diagram".

25.2 Overview

OPAMP can be used either as a standalone amplifier or as a programmable gain amplifier (PGA). An analog MUX is used for selecting an analog input signal from four external channels of amplifier positive and negative inputs respectively. When OPAMP is enabled, the OPAMP output connects to both internal ADC channels and analog comparator (CMP) inputs, meanwhile a package pin must be configured also as OPAMP output.

25.2.1 Block diagram

OPAMP can be configured as various amplifiers by four configuration registers (CFG0 to CFG3), are shown in the block diagram. Only one configuration register takes effect at any moment. The selection of CFG*x* can be either software or other peripherals through inter-module connections.

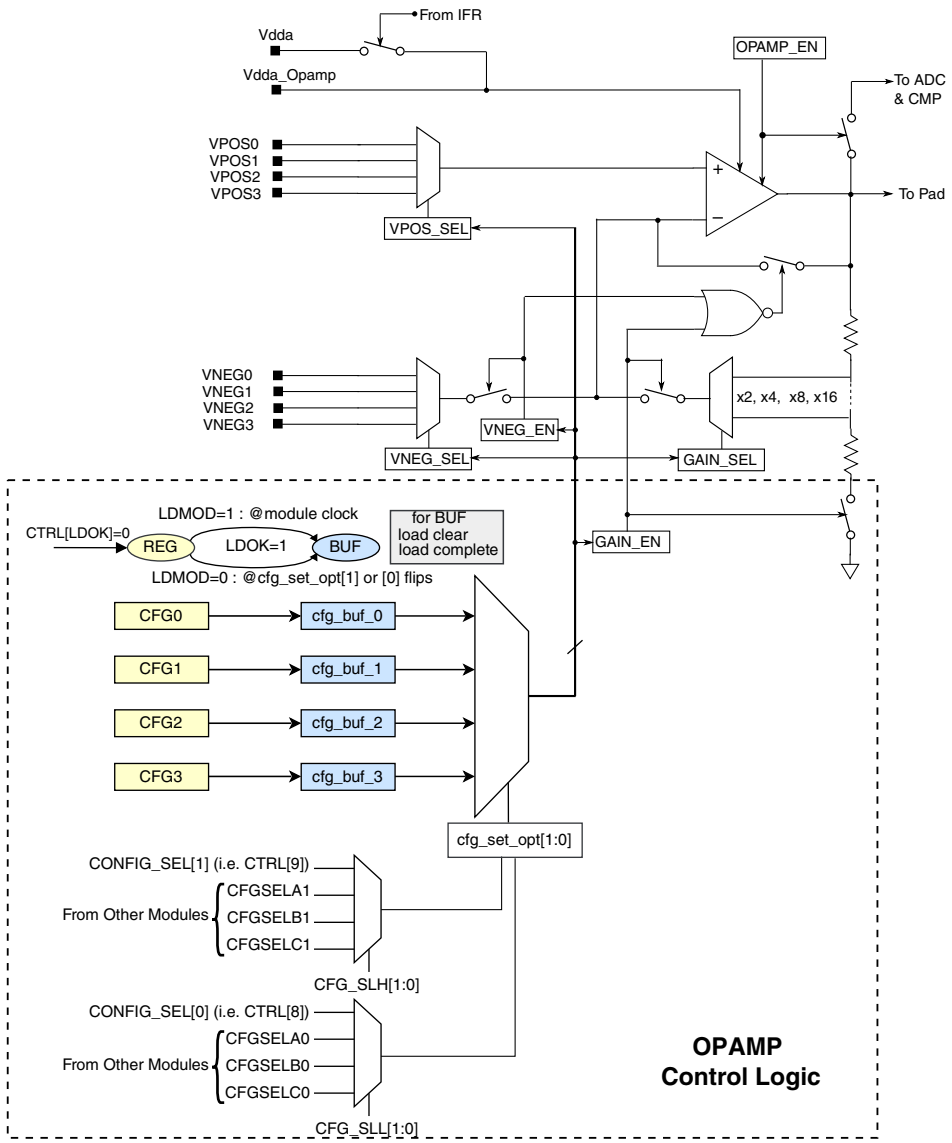


Figure 25-1. OPAMP block diagram

25.2.2 Features

OPAMP includes the following features:

- Capability of being configured as various types of amplifier:
  - standalone operational amplifier
  - unity gain follower (voltage follower)
  - x2, x4, x8, x16 PGA
  - differential amplifier
  - low-pass filter
- 4-to-1 input multiplexer on inverting and non-inverting input
- 4 set configurations including multiplexer inputs can be managed by internal modules and synchronized with ADCs, PWMs and timers
- Operation modes: high speed mode and low power mode

25.3 Functional description

OPAMP has 4 configurations (controlled by registers CFG*x*), which contain selecting the positive and negative inputs and gain information. Only one configuration takes effect at a time. Each CFG*x* register has a corresponding buffer called cfg_buf_*x* (*x*=0~3), and this buffer value really controls the OPAMP settings. See Figure 25-1.

There are a set of internal signals cfg_set_opt[1:0] that choose which cfg_buf_*x* takes effect. When cfg_set_opt[1:0] = 00b, it is the setting in cfg_buf_0 that controls the OPAMP; when cfg_set_opt[1:0] = 01b, it is the setting in cfg_buf_1 that controls the OPAMP, and so on. The internal signals cfg_set_opt[1:0] come from either software or other modules inside the chip.

- When CTRL[CFG_SLH] is 00b and CTRL[CFG_SLL] is 00b, cfg_set_opt[1:0] are directly decided by software, which is the value in CTRL[CONFIG_SEL].
- When CTRL[CFG_SLH] is 01b and CTRL[CFG_SLL] is 01b, cfg_set_opt[1:0] come from the signals on input nodes CFGSELA1 and CFGSELA0.
- When CTRL[CFG_SLH] is 10b and CTRL[CFG_SLL] is 10b, cfg_set_opt[1:0] come from the signals on input nodes CFGSELB1 and CFGSELB0.
- When CTRL[CFG_SLH] is 11b and CTRL[CFG_SLL] is 11b, cfg_set_opt[1:0] come from the signals on input nodes CFGSELC1 and CFGSELC0.

Writing to CFG*x* registers does not directly update its corresponding buffer cfg_buf_*x*.

- When CTRL[LDMOD] = 0, the values in CFG*x* registers are loaded into corresponding cfg_buf_*x* when cfg_set_opt[1:0] signal is changed and CTRL[LDOK] is already set to 1, and then CTRL[LDOK] is cleared automatically.
- When CTRL[LDMOD] = 1, the values in CFG*x* registers are loaded into corresponding cfg_buf_*x* at the next rising edge of module clock right after CTRL[LDOK] is set, and then CTRL[LDOK] is cleared automatically.

To write CFG*x* registers, ensure CTRL[WP] = 0 and CTRL[LDOK] = 0 as well. Reading CFG*x* registers return the values in the registers but not from the buffers. STAT[LDCMF] flag bit is set when values of CFG*x* registers are successfully loaded into the buffer cfg_buf_*x*. An interrupt can be generated if CTRL[LDCMIE] is 1.

To ensure all the buffers cfg_buf_*x* contain desired OPAMP settings, follow the initialization sequence as below:

1. Set CTRL[EN] to 1, and enable OPAMP output function on corresponding pin through SIM and GPIO registers (see the "Chip-specific information" section).
2. Clear CTRL[WP] by setting STAT[WP_CLR] to 10b.
3. Clear CTRL[LDOK] by setting CTRL[LDOK] to 0.
4. Set CTRL[LDMOD] to 1.
5. Set desired values to CFG*x*.
6. Set CTRL[LDOK] to 1.
7. Wait till CTRL[LDOK] turns to 0.
8. Clear STAT[LDCMF] by writing 1 to this bit.
9. Update CTRL[LDMOD] depending on application requirements.

25.3.1 Modes configuration

There is one 4:1 analog multiplexer on the non-inverting input and the inverting input respectively. CFGx[VPOS_SEL] and CFGx[VNEG_SEL] select one of four inputs for the non-inverting input and the inverting input respectively. After reset, OPAMP operates as a follower. CFGx[GAIN_SEL] selects the OPAMP feedback gain when CFGx[GAIN_EN] is set. CFGx[VNEG_EN] configures OPAMP into PGA or difference amplifier.

This section describes all functional operation modes of OPAMP.

CFGx[VNEG_EN]	CFGx[GAIN_EN]	CFGx[GAIN_SEL]	Gain	Function Description
0	0	xx	1	Voltage follower mode
x	1	00	2	PGA mode: Non-inverting gain amplifier through internal connections on chip. [1]
		01	4	
		10	8	
		11	16	
1	0	xx	-	OPAMP mode: Negative input of amplifier is switched to external channels with external resistance net. Different resistance net determines different gain.

1. When VNEG_EN=1 and GAIN_EN=1, internal gain is selected, suspending capacitor between external negative and output channels can optimize transient response.

NOTE

To meet the good noise rejection and optimize the transition response during the gain change when OPAMP is configured as the PGA mode, an external capacitor can be added between OPAMP output pin and a negative input pin if VNRG_EN =1. However, it is highly discouraged to add any resistor between OPAMP output pin and a negative input pin which results uncertain gain.

Diagrams of operation modes are as below:

- The OPAMP can operate as a voltage follower.

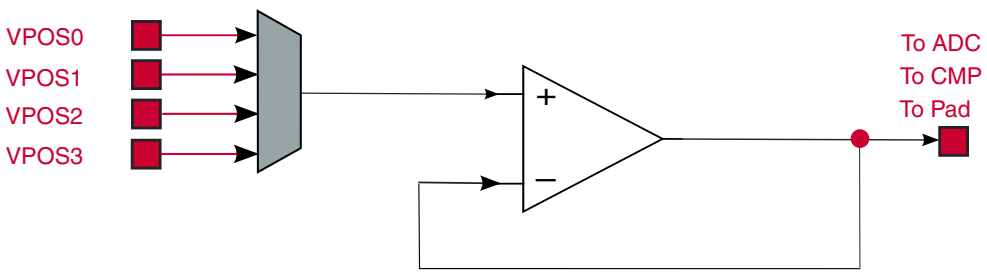


Figure 25-2. Voltage follower mode diagram

- PGA mode: non-inverting gain amplifier, through internal connections on chip. When VNEG_EN=1, suspending capacitor between external negative and output channels can optimize transient response. When VNEG*x* (*x*=0~3) is connected.

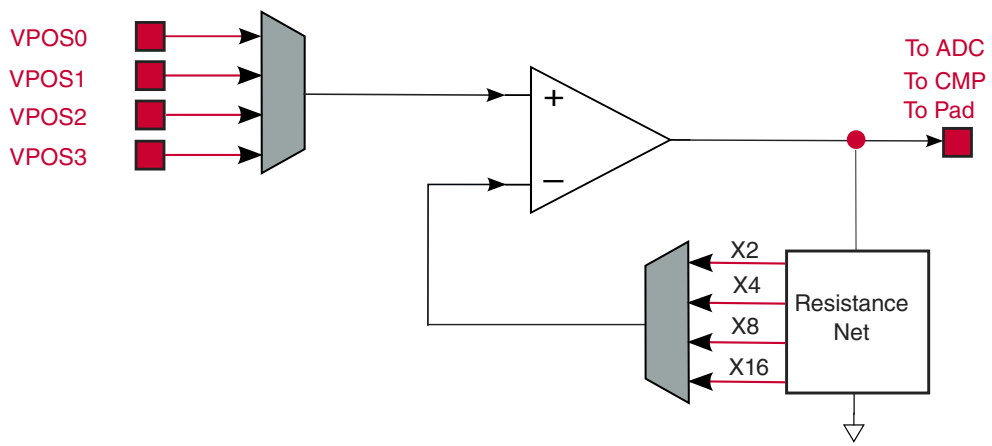


Figure 25-3. PGA mode diagram

- OPAMP mode: negative input of amplifier is switched to external channels with external resistance net. Different resistance net determines different gain.

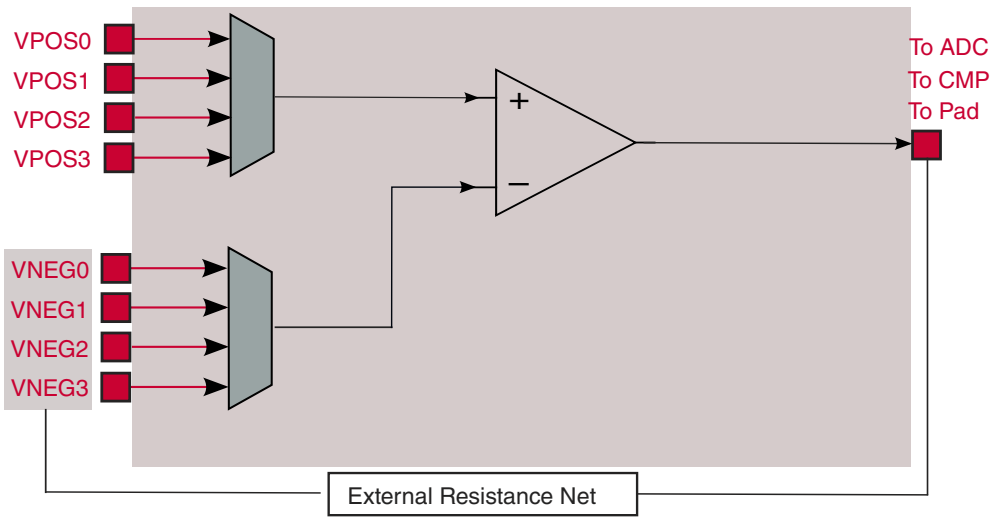


Figure 25-4. OPAMP mode diagram

25.3.2 Power modes

OPAMP operates in two power modes.

- When CTRL[PMOD] is set to 1, OPAMP is in high speed mode, having higher current consumption with faster slew rate and wider unity gain bandwidth performance.
- When CTRL[PMOD] is cleared to 0, OPAMP is in low power mode, having lower current consumption with slower slew rate and narrower unity gain bandwidth performance.

25.3.3 Clocking

This module has no clocking considerations.

25.3.4 Reset

When there is system reset, OPAMP converts to initial state, with buffer cleared and waiting to load new data.

25.3.5 Interrupts

OPAMP contains loading completion interrupt. When both CTRL[LDCMIE] and STAT[LDCMF] are logic 1, loading completion interrupt generates. Write 1 to clear STAT[LDCMF].

25.4 Signals

See the chip-specific information section, for external signal connection.

25.5 Memory map and register definition

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

NOTE

Attempting to access a reserved register location in the OPAMP memory map will generate a bus error.

25.5.1 OPAMP register descriptions

25.5.1.1 OPAMP memory map

OPAMPA base address: E060h

OPAMPB base address: E068h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control register (CTRL)	16	RW	0000h
1h	Status register (STAT)	16	RW	0000h
2h	Configuration register (CFG0)	16	RW	0000h
3h	Configuration register (CFG1)	16	RW	0000h
4h	Configuration register (CFG2)	16	RW	0000h
5h	Configuration register (CFG3)	16	RW	0000h

25.5.1.2 Control register (CTRL)

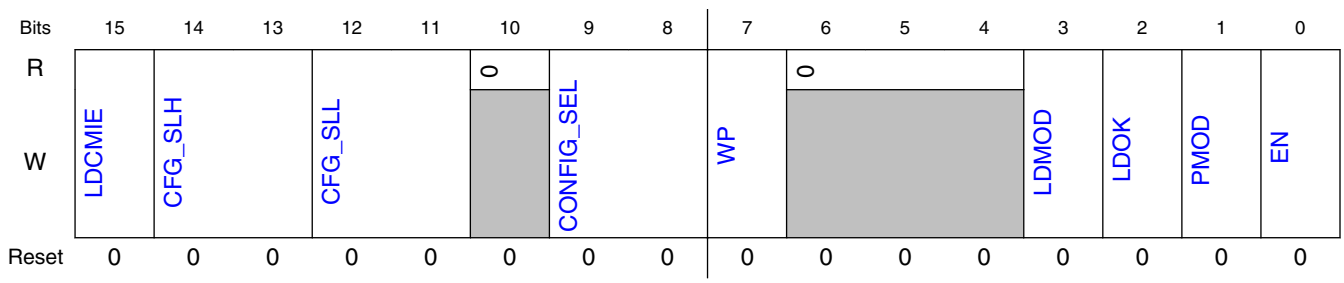
25.5.1.2.1 Offset

Register	Offset
CTRL	0h

25.5.1.2.2 Function

CTRL configures with the purpose to select working modes.

25.5.1.2.3 Diagram



25.5.1.2.4 Fields

Field	Function
15 LDCMIE	Load Completion Interrupt Enable Enables the load completion flag STAT[LDCMF] to generate a CPU interrupt request. 0b - Disables 1b - Enables
14-13 CFG_SLH	Configuration register selection source signal high select Defines one of four signals that acts as the most significant bit of configuration register (CFGx) selection cfg_set_opt[1:0]. 00b - CONFIG_SEL[1] bit selects configuration register. 01b - External signal CFGSELA1 selects configuration register. 10b - External signal CFGSELB1 selects configuration register. 11b - External signal CFGSELC1 selects configuration register.
12-11 CFG_SLL	Configuration register selection source signal low select Defines one of four signals that acts as the least significant bit of configuration register (CFGx) selection cfg_set_opt[1:0]. 00b - CONFIG_SEL[0] bit selects configuration register. 01b - External signal CFGSELA0 selects configuration register. 10b - External signal CFGSELB0 selects configuration register. 11b - External signal CFGSELC0 selects configuration register.

Table continues on the next page...

Field	Function
10 —	Reserved
9-8 CONFIG_SEL	Configuration register software selection Defines which configuration register (CFG0 -CFG3) is activated to configure OPAMP, when both CTRL[CFG_SSL] and CTRL[CFG_SLL] are configured to 00b. 00b - CFG0 activated 01b - CFG1 activated 10b - CFG2 activated 11b - CFG3 activated
7 WP	Register Write Protection The field values of CTRL[LDCMIE], CTRL[CFG_SLH], CTRL[CFG_SLL], CTRL[CONFIG_SEL], CTRL[LDMOD], CTRL[PMOD], CTRL[EN], and CFGx are switched by CTRL[WP] for write protection. When WP is set, it is not possible to write these fields. When WP is cleared, these fields can be written and read properly. WP can be cleared by setting 10b to STAT[WP_CLR]. **NOTE:** Field values of CTRL[LDOK], STAT[LDCMF] and STAT[WP_CLR] are not guarded by this WP bit, therefore can be always written and read. 0b - Write and read. 1b - Read only.
6-4 —	Reserved
3 LDMOD	Load Mode Select Defines load mode. 0b - Values of CFGx registers are loaded into the corresponding buffers cfg_buf_x when cfg_set_opt[1:0] signal is changed and CTRL[LDOK] has been set to 1. CTRL[LDOK] is automatically cleared after CFGx register values are loaded into their buffers. 1b - Values of CFGx registers are loaded into the corresponding buffers cfg_buf_x when CTRL[LDOK] is set to 1. CTRL[LDOK] is automatically cleared after CFGx register values are loaded into their buffers.
2 LDOK	Load Okay Setting this bit means the values in CFGx registers are ready to be loaded to their buffers. LDOK is automatically cleared after a successful load completion. 0b - Values in CFGx registers are not ready to be loaded into their buffers. 1b - Values in CFGx registers are ready to be loaded into their buffers.
1 PMOD	Power Mode Select 0b - Low power mode. 1b - High speed mode.
0 EN	Enables OPAMP and switch registers 0b - OPAMP is disabled. CTRL[WP], CTRL[LDOK], STAT[WR_CLR] and STAT[LDCMF] are not readable and writable. 1b - OPAMP is enabled. CTRL[WP], CTRL[LDOK], STAT[WP_CLR] and STAT[LDCMF] are readable and writable.

25.5.1.3 Status register (STAT)

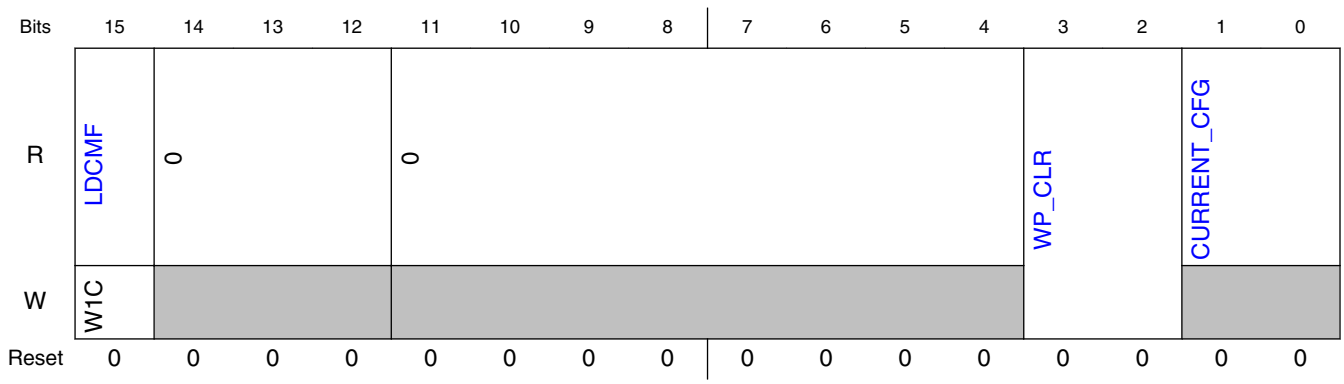
25.5.1.3.1 Offset

Register	Offset
STAT	1h

25.5.1.3.2 Function

Shows some working status, such as current configuration information and loading completion flag.

25.5.1.3.3 Diagram



25.5.1.3.4 Fields

Field	Function
15 LDCMF	Load Completion Flag This bit is set to 1 when CFGx register values are successfully loaded into their buffers. Write 1 to clear this bit.
14-12 —	Reserved
11-4 —	Reserved
3-2 WP_CLR	Clears the CTRL[WP] bit Writing 10b to this field clears CTRL[WP], that could disable the write protection.
1-0 CURRENT_CFG	Current effective Configuration register This field is read-only. Shows the index of current CFGx.

25.5.1.4 Configuration register (CFG0 - CFG3)

25.5.1.4.1 Offset

Register	Offset
CFG0	2h
CFG1	3h
CFG2	4h
CFG3	5h

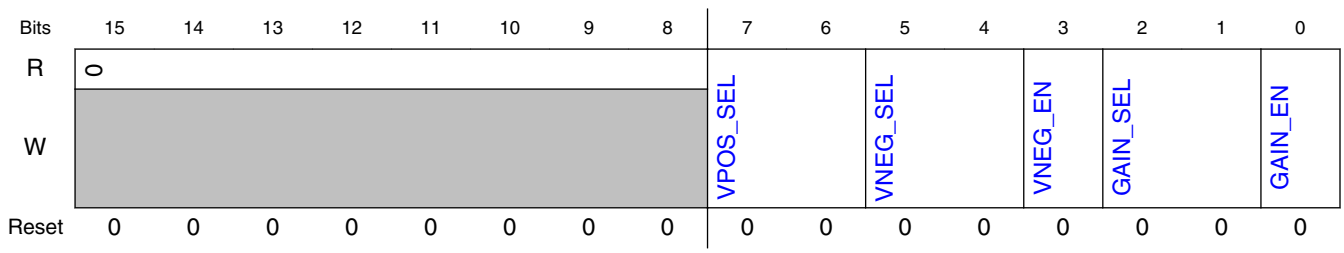
25.5.1.4.2 Function

Provides the first configuration information for amplifier which contains selecting the positive and negative inputs and gain information.

NOTE

Configuration registers are buffered. Reading these registers gets the value in the register and not necessarily the buffered value that OPAMP is currently using.

25.5.1.4.3 Diagram



25.5.1.4.4 Fields

Field	Function
15-8 —	Reserved
7-6 VPOS_SEL	VPOS Selection Decides which VPOSx is selected. 00b - Selects VPOS0. 01b - Selects VPOS1. 10b - Selects VPOS2. 11b - Selects VPOS3.

Table continues on the next page...

Field	Function
5-4 VNEG_SEL	VNEGx Selection Decides which external VNEGx is selected. 00b - Selects VNEG0. 01b - Selects VNEG1. 10b - Selects VNEG2. 11b - Selects VNEG3.
3 VNEG_EN	External VNEG Enable Decides the negative input sources of Amplifier. If VNEG_EN = 1 and GAIN_EN = 0, it selects external VNEGx as the negative input of amplifier. **NOTE:** When GAIN_EN = 1 and VNEG_EN = 1, internal gain is selected, suspending capacitor between external negative and output channels can optimize transient response, but a key point here is that external resistance net can not be added into the external channel which will cause gain confusion.
2-1 GAIN_SEL	Gain Selection Decides gain value for application (non-inverting gain). 00b - Gain value 2X. 01b - Gain value 4X. 10b - Gain value 8X. 11b - Gain value 16X.
0 GAIN_EN	Gain Enable Enables the internal gain. 0b - Internal gain is disabled. 1b - Internal gain is enabled.

Chapter 26
Enhanced Flexible Pulse Width Modulator (PWM)

26.1 Chip-specific information for this module

For this device, PWM_OUT_TRIG0/1 and PWM output are connected out to XBAR directly (see the "XBARA Inputs" table in XBAR chapter), so PWAOT0/PWBOT1 and PWM_MUX_TRIG0/1 are not applicable in the figure "Output logic for 1 submodule".

26.1.1 PWM auxiliary signals and analog inputs

Cyclic ADC (ADC12) conversion high/low limits can be fed to the PWM*x*'s EXTB when they are used as inputs. In this configuration:

- When an ADC conversion is higher than a pre-programmed high limit, the signal going to the EXTB input is low.
- When an ADC conversion is lower than a pre-programmed low limit, the signal going to the EXTB input is high.

26.2 Overview

The Pulse Width Modulator (PWM) module contains PWM submodules, each of which can be used to control a single half-bridge power stage. Fault channel support is provided.

This module generates various switching patterns, including highly sophisticated waveforms. It is ideal for controlling different Switched Mode Power Supplies (SMPS) topologies.

26.2.1 Block diagram

The following figure shows the PWM block diagram.

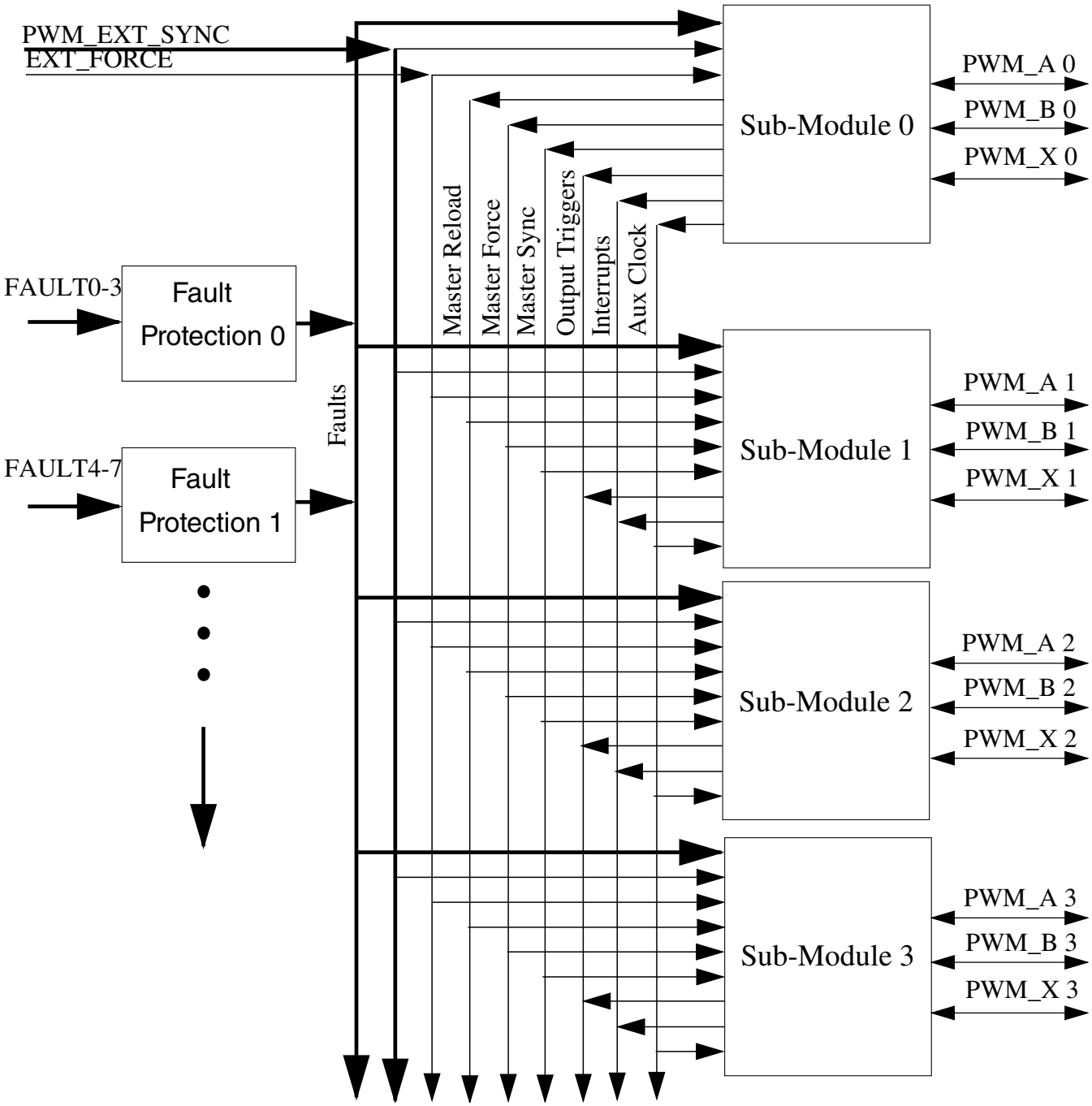


Figure 26-1. PWM block diagram

26.2.2 Features

Following are the features of PWM:

- 16-bit resolution for center, edge-aligned, and asymmetrical PWMs
- Fractional PWM clock generation for enhanced resolution of the PWM period and duty cycle

- PWM outputs that can operate as complementary pairs or independent channels
- Ability to accept signed numbers for PWM generation
- Independent control of both edges of each PWM output
- Support for synchronization to external hardware or other PWM
- Double buffered PWM registers
  - Integral reload rates from 1 to 16
  - Half cycle reload capability
- Multiple output trigger events can be generated per PWM cycle via hardware
- Support for double switching PWM outputs
- Fault inputs can be assigned to control multiple PWM outputs
- Programmable filters for fault inputs
- Independently programmable PWM output polarity
- Independent top and bottom deadtime insertion
- Each complementary pair can operate with its own PWM frequency and deadtime values
- Individual software control for each PWM output
- All outputs can be programmed to change simultaneously via a FORCE_OUT event
- PWM_X pin can optionally output a third PWM signal from each submodule
- Channels not used for PWM generation can be used for buffered output compare functions and for input capture functions
- Enhanced dual edge capture functionality

26.3 PWM register descriptions

26.3.1 PWM memory map

PWMA base address: E600h

Offset	Register	Width (In bits)	Access	Reset value
0h	Counter Register (SM0CNT)	16	RO	0000h
1h	Initial Count Register (SM0INIT)	16	RW	0000h
2h	Control 2 Register (SM0CTRL2)	16	RW	0000h
3h	Control Register (SM0CTRL)	16	RW	0400h
5h	Value Register 0 (SM0VAL0)	16	RW	0000h
6h	Fractional Value Register 1 (SM0FRACVAL1)	16	RW	0000h
7h	Value Register 1 (SM0VAL1)	16	RW	0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
8h	Fractional Value Register 2 (SM0FRACVAL2)	16	RW	0000h
9h	Value Register 2 (SM0VAL2)	16	RW	0000h
Ah	Fractional Value Register 3 (SM0FRACVAL3)	16	RW	0000h
Bh	Value Register 3 (SM0VAL3)	16	RW	0000h
Ch	Fractional Value Register 4 (SM0FRACVAL4)	16	RW	0000h
Dh	Value Register 4 (SM0VAL4)	16	RW	0000h
Eh	Fractional Value Register 5 (SM0FRACVAL5)	16	RW	0000h
Fh	Value Register 5 (SM0VAL5)	16	RW	0000h
10h	Fractional Control Register (SM0FRCTRL)	16	RW	0000h
11h	Output Control Register (SM0OCTRL)	16	RW	0000h
12h	Status Register (SM0STS)	16	W1C	0000h
13h	Interrupt Enable Register (SM0INTEN)	16	RW	0000h
14h	DMA Enable Register (SM0DMAEN)	16	RW	0000h
15h	Output Trigger Control Register (SM0TCTRL)	16	RW	0000h
16h	Fault Disable Mapping Register 0 (SM0DISMAP0)	16	RW	FFFFh
17h	Fault Disable Mapping Register 1 (SM0DISMAP1)	16	RW	FFFFh
18h	Deadtime Count Register 0 (SM0DTCNT0)	16	RW	07FFh
19h	Deadtime Count Register 1 (SM0DTCNT1)	16	RW	07FFh
1Ah	Capture Control A Register (SM0CAPTCTRLA)	16	RW	0000h
1Bh	Capture Compare A Register (SM0CAPTCOMPA)	16	RW	0000h
1Ch	Capture Control B Register (SM0CAPTCTRLB)	16	RW	0000h
1Dh	Capture Compare B Register (SM0CAPTCOMPB)	16	RW	0000h
1Eh	Capture Control X Register (SM0CAPTCTRLX)	16	RW	0000h
1Fh	Capture Compare X Register (SM0CAPTCOMPX)	16	RW	0000h
20h	Capture Value 0 Register (SM0CVAL0)	16	RO	0000h
21h	Capture Value 0 Cycle Register (SM0CVAL0CYC)	16	RO	0000h
22h	Capture Value 1 Register (SM0CVAL1)	16	RO	0000h
23h	Capture Value 1 Cycle Register (SM0CVAL1CYC)	16	RO	0000h
24h	Capture Value 2 Register (SM0CVAL2)	16	RO	0000h
25h	Capture Value 2 Cycle Register (SM0CVAL2CYC)	16	RO	0000h
26h	Capture Value 3 Register (SM0CVAL3)	16	RO	0000h
27h	Capture Value 3 Cycle Register (SM0CVAL3CYC)	16	RO	0000h
28h	Capture Value 4 Register (SM0CVAL4)	16	RO	0000h
29h	Capture Value 4 Cycle Register (SM0CVAL4CYC)	16	RO	0000h
2Ah	Capture Value 5 Register (SM0CVAL5)	16	RO	0000h
2Bh	Capture Value 5 Cycle Register (SM0CVAL5CYC)	16	RO	0000h
2Dh	Capture PWMA Input Filter Register (SM0CAPTFILTA)	16	RW	0000h
2Eh	Capture PWMB Input Filter Register (SM0CAPTFILTB)	16	RW	0000h
2Fh	Capture PWMX Input Filter Register (SM0CAPTFILTX)	16	RW	0000h
30h	Counter Register (SM1CNT)	16	RO	0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
31h	Initial Count Register (SM1INIT)	16	RW	0000h
32h	Control 2 Register (SM1CTRL2)	16	RW	0000h
33h	Control Register (SM1CTRL)	16	RW	0400h
35h	Value Register 0 (SM1VAL0)	16	RW	0000h
36h	Fractional Value Register 1 (SM1FRACVAL1)	16	RW	0000h
37h	Value Register 1 (SM1VAL1)	16	RW	0000h
38h	Fractional Value Register 2 (SM1FRACVAL2)	16	RW	0000h
39h	Value Register 2 (SM1VAL2)	16	RW	0000h
3Ah	Fractional Value Register 3 (SM1FRACVAL3)	16	RW	0000h
3Bh	Value Register 3 (SM1VAL3)	16	RW	0000h
3Ch	Fractional Value Register 4 (SM1FRACVAL4)	16	RW	0000h
3Dh	Value Register 4 (SM1VAL4)	16	RW	0000h
3Eh	Fractional Value Register 5 (SM1FRACVAL5)	16	RW	0000h
3Fh	Value Register 5 (SM1VAL5)	16	RW	0000h
40h	Fractional Control Register (SM1FRCTRL)	16	RW	0000h
41h	Output Control Register (SM1OCTRL)	16	RW	0000h
42h	Status Register (SM1STS)	16	W1C	0000h
43h	Interrupt Enable Register (SM1INTEN)	16	RW	0000h
44h	DMA Enable Register (SM1DMAEN)	16	RW	0000h
45h	Output Trigger Control Register (SM1TCTRL)	16	RW	0000h
46h	Fault Disable Mapping Register 0 (SM1DISMAP0)	16	RW	FFFFh
47h	Fault Disable Mapping Register 1 (SM1DISMAP1)	16	RW	FFFFh
48h	Deadtime Count Register 0 (SM1DTCNT0)	16	RW	07FFh
49h	Deadtime Count Register 1 (SM1DTCNT1)	16	RW	07FFh
4Ah	Capture Control A Register (SM1CAPTCTRLA)	16	RW	0000h
4Bh	Capture Compare A Register (SM1CAPTCOMPA)	16	RW	0000h
4Ch	Capture Control B Register (SM1CAPTCTRLB)	16	RW	0000h
4Dh	Capture Compare B Register (SM1CAPTCOMPB)	16	RW	0000h
4Eh	Capture Control X Register (SM1CAPTCTRLX)	16	RW	0000h
4Fh	Capture Compare X Register (SM1CAPTCOMPX)	16	RW	0000h
50h	Capture Value 0 Register (SM1CVAL0)	16	RO	0000h
51h	Capture Value 0 Cycle Register (SM1CVAL0CYC)	16	RO	0000h
52h	Capture Value 1 Register (SM1CVAL1)	16	RO	0000h
53h	Capture Value 1 Cycle Register (SM1CVAL1CYC)	16	RO	0000h
54h	Capture Value 2 Register (SM1CVAL2)	16	RO	0000h
55h	Capture Value 2 Cycle Register (SM1CVAL2CYC)	16	RO	0000h
56h	Capture Value 3 Register (SM1CVAL3)	16	RO	0000h
57h	Capture Value 3 Cycle Register (SM1CVAL3CYC)	16	RO	0000h
58h	Capture Value 4 Register (SM1CVAL4)	16	RO	0000h
59h	Capture Value 4 Cycle Register (SM1CVAL4CYC)	16	RO	0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
5Ah	Capture Value 5 Register (SM1CVAL5)	16	RO	0000h
5Bh	Capture Value 5 Cycle Register (SM1CVAL5CYC)	16	RO	0000h
5Ch	Phase Delay Register (SM1PHASEDLY)	16	RW	0000h
5Dh	Capture PWMA Input Filter Register (SM1CAPTFILTA)	16	RW	0000h
5Eh	Capture PWMB Input Filter Register (SM1CAPTFILTB)	16	RW	0000h
5Fh	Capture PWMX Input Filter Register (SM1CAPTFILTX)	16	RW	0000h
60h	Counter Register (SM2CNT)	16	RO	0000h
61h	Initial Count Register (SM2INIT)	16	RW	0000h
62h	Control 2 Register (SM2CTRL2)	16	RW	0000h
63h	Control Register (SM2CTRL)	16	RW	0400h
65h	Value Register 0 (SM2VAL0)	16	RW	0000h
66h	Fractional Value Register 1 (SM2FRACVAL1)	16	RW	0000h
67h	Value Register 1 (SM2VAL1)	16	RW	0000h
68h	Fractional Value Register 2 (SM2FRACVAL2)	16	RW	0000h
69h	Value Register 2 (SM2VAL2)	16	RW	0000h
6Ah	Fractional Value Register 3 (SM2FRACVAL3)	16	RW	0000h
6Bh	Value Register 3 (SM2VAL3)	16	RW	0000h
6Ch	Fractional Value Register 4 (SM2FRACVAL4)	16	RW	0000h
6Dh	Value Register 4 (SM2VAL4)	16	RW	0000h
6Eh	Fractional Value Register 5 (SM2FRACVAL5)	16	RW	0000h
6Fh	Value Register 5 (SM2VAL5)	16	RW	0000h
70h	Fractional Control Register (SM2FRCTRL)	16	RW	0000h
71h	Output Control Register (SM2OCTRL)	16	RW	0000h
72h	Status Register (SM2STS)	16	W1C	0000h
73h	Interrupt Enable Register (SM2INTEN)	16	RW	0000h
74h	DMA Enable Register (SM2DMAEN)	16	RW	0000h
75h	Output Trigger Control Register (SM2TCTRL)	16	RW	0000h
76h	Fault Disable Mapping Register 0 (SM2DISMAP0)	16	RW	FFFFh
77h	Fault Disable Mapping Register 1 (SM2DISMAP1)	16	RW	FFFFh
78h	Deadtime Count Register 0 (SM2DTCNT0)	16	RW	07FFh
79h	Deadtime Count Register 1 (SM2DTCNT1)	16	RW	07FFh
7Ah	Capture Control A Register (SM2CAPTCTRLA)	16	RW	0000h
7Bh	Capture Compare A Register (SM2CAPTCOMPA)	16	RW	0000h
7Ch	Capture Control B Register (SM2CAPTCTRLB)	16	RW	0000h
7Dh	Capture Compare B Register (SM2CAPTCOMPB)	16	RW	0000h
7Eh	Capture Control X Register (SM2CAPTCTRLX)	16	RW	0000h
7Fh	Capture Compare X Register (SM2CAPTCOMPX)	16	RW	0000h
80h	Capture Value 0 Register (SM2CVAL0)	16	RO	0000h
81h	Capture Value 0 Cycle Register (SM2CVAL0CYC)	16	RO	0000h
82h	Capture Value 1 Register (SM2CVAL1)	16	RO	0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
83h	Capture Value 1 Cycle Register (SM2CVAL1CYC)	16	RO	0000h
84h	Capture Value 2 Register (SM2CVAL2)	16	RO	0000h
85h	Capture Value 2 Cycle Register (SM2CVAL2CYC)	16	RO	0000h
86h	Capture Value 3 Register (SM2CVAL3)	16	RO	0000h
87h	Capture Value 3 Cycle Register (SM2CVAL3CYC)	16	RO	0000h
88h	Capture Value 4 Register (SM2CVAL4)	16	RO	0000h
89h	Capture Value 4 Cycle Register (SM2CVAL4CYC)	16	RO	0000h
8Ah	Capture Value 5 Register (SM2CVAL5)	16	RO	0000h
8Bh	Capture Value 5 Cycle Register (SM2CVAL5CYC)	16	RO	0000h
8Ch	Phase Delay Register (SM2PHASEDLY)	16	RW	0000h
8Dh	Capture PWMA Input Filter Register (SM2CAPTFILTA)	16	RW	0000h
8Eh	Capture PWMB Input Filter Register (SM2CAPTFILTB)	16	RW	0000h
8Fh	Capture PWMX Input Filter Register (SM2CAPTFILTX)	16	RW	0000h
90h	Counter Register (SM3CNT)	16	RO	0000h
91h	Initial Count Register (SM3INIT)	16	RW	0000h
92h	Control 2 Register (SM3CTRL2)	16	RW	0000h
93h	Control Register (SM3CTRL)	16	RW	0400h
95h	Value Register 0 (SM3VAL0)	16	RW	0000h
96h	Fractional Value Register 1 (SM3FRACVAL1)	16	RW	0000h
97h	Value Register 1 (SM3VAL1)	16	RW	0000h
98h	Fractional Value Register 2 (SM3FRACVAL2)	16	RW	0000h
99h	Value Register 2 (SM3VAL2)	16	RW	0000h
9Ah	Fractional Value Register 3 (SM3FRACVAL3)	16	RW	0000h
9Bh	Value Register 3 (SM3VAL3)	16	RW	0000h
9Ch	Fractional Value Register 4 (SM3FRACVAL4)	16	RW	0000h
9Dh	Value Register 4 (SM3VAL4)	16	RW	0000h
9Eh	Fractional Value Register 5 (SM3FRACVAL5)	16	RW	0000h
9Fh	Value Register 5 (SM3VAL5)	16	RW	0000h
A0h	Fractional Control Register (SM3FRCTRL)	16	RW	0000h
A1h	Output Control Register (SM3OCTRL)	16	RW	0000h
A2h	Status Register (SM3STS)	16	W1C	0000h
A3h	Interrupt Enable Register (SM3INTEN)	16	RW	0000h
A4h	DMA Enable Register (SM3DMAEN)	16	RW	0000h
A5h	Output Trigger Control Register (SM3TCTRL)	16	RW	0000h
A6h	Fault Disable Mapping Register 0 (SM3DISMAP0)	16	RW	FFFFh
A7h	Fault Disable Mapping Register 1 (SM3DISMAP1)	16	RW	FFFFh
A8h	Deadtime Count Register 0 (SM3DTCNT0)	16	RW	07FFh
A9h	Deadtime Count Register 1 (SM3DTCNT1)	16	RW	07FFh
AAh	Capture Control A Register (SM3CAPTCTRLA)	16	RW	0000h
ABh	Capture Compare A Register (SM3CAPTCOMPA)	16	RW	0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
ACh	Capture Control B Register (SM3CAPTCTRLB)	16	RW	0000h
ADh	Capture Compare B Register (SM3CAPTCOMPB)	16	RW	0000h
AEh	Capture Control X Register (SM3CAPTCTRLX)	16	RW	0000h
AFh	Capture Compare X Register (SM3CAPTCOMPX)	16	RW	0000h
B0h	Capture Value 0 Register (SM3CVAL0)	16	RO	0000h
B1h	Capture Value 0 Cycle Register (SM3CVAL0CYC)	16	RO	0000h
B2h	Capture Value 1 Register (SM3CVAL1)	16	RO	0000h
B3h	Capture Value 1 Cycle Register (SM3CVAL1CYC)	16	RO	0000h
B4h	Capture Value 2 Register (SM3CVAL2)	16	RO	0000h
B5h	Capture Value 2 Cycle Register (SM3CVAL2CYC)	16	RO	0000h
B6h	Capture Value 3 Register (SM3CVAL3)	16	RO	0000h
B7h	Capture Value 3 Cycle Register (SM3CVAL3CYC)	16	RO	0000h
B8h	Capture Value 4 Register (SM3CVAL4)	16	RO	0000h
B9h	Capture Value 4 Cycle Register (SM3CVAL4CYC)	16	RO	0000h
BAh	Capture Value 5 Register (SM3CVAL5)	16	RO	0000h
BBh	Capture Value 5 Cycle Register (SM3CVAL5CYC)	16	RO	0000h
BCh	Phase Delay Register (SM3PHASEDLY)	16	RW	0000h
BDh	Capture PWMA Input Filter Register (SM3CAPTFILTA)	16	RW	0000h
BEh	Capture PWMB Input Filter Register (SM3CAPTFILTB)	16	RW	0000h
BFh	Capture PWMX Input Filter Register (SM3CAPTFILTX)	16	RW	0000h
C0h	Output Enable Register (OUTEN)	16	RW	0000h
C1h	Mask Register (MASK)	16	RW	0000h
C2h	Software Controlled Output Register (SWCOUT)	16	RW	0000h
C3h	PWM Source Select Register (DTSRCSEL)	16	RW	0000h
C4h	Master Control Register (MCTRL)	16	RW	0000h
C5h	Master Control 2 Register (MCTRL2)	16	RW	0000h
C6h	Fault Control Register (FCTRL0)	16	RW	0000h
C7h	Fault Status Register (FSTS0)	16	RW	0000h
C8h	Fault Filter Register (FFILT0)	16	RW	0000h
C9h	Fault Test Register (FTST0)	16	RW	0000h
CAh	Fault Control 2 Register (FCTRL20)	16	RW	0000h
CCh	Fault Control Register (FCTRL1)	16	RW	0000h
CDh	Fault Status Register (FSTS1)	16	RW	0000h
CEh	Fault Filter Register (FFILT1)	16	RW	0000h
CFh	Fault Test Register (FTST1)	16	RW	0000h
D0h	Fault Control 2 Register (FCTRL21)	16	RW	0000h

26.3.2 Counter Register (SM0CNT - SM3CNT)

26.3.2.1 Offset

Register	Offset
SM0CNT	0h
SM1CNT	30h
SM2CNT	60h
SM3CNT	90h

26.3.2.2 Function

This read-only register displays the state of the signed 16-bit submodule counter. This register is not byte accessible. Writing this register generates bus transfer error.

26.3.2.3 Diagram

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

26.3.2.4 Fields

Field	Function
15-0 CNT	Counter Register Bits

26.3.3 Initial Count Register (SM0INIT - SM3INIT)

26.3.3.1 Offset

Register	Offset
SM0INIT	1h
SM1INIT	31h
SM2INIT	61h
SM3INIT	91h

26.3.3.2 Function

The 16-bit signed value in this buffered, read/write register defines the initial count value for the PWM in PWM clock periods. This is the value loaded into the submodule counter when local sync, master sync, or master reload is asserted (based on the value of CTRL2[INIT_SEL]) or when CTRL2[FORCE] is asserted and force init is enabled. For PWM operation, the buffered contents of this register are loaded into the counter at the start of every PWM cycle. This register is not byte accessible.

NOTE

The INIT register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. This register cannot be written when MCTRL[LDOK] is set. Reading INIT reads the value in a buffer and not necessarily the value the PWM generator is currently using.

26.3.3.3 Diagram

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

26.3.3.4 Fields

Field	Function
15-0	Initial Count Register Bits

Field	Function
INIT	

26.3.4 Control 2 Register (SM0CTRL2 - SM3CTRL2)

26.3.4.1 Offset

Register	Offset
SM0CTRL2	2h
SM1CTRL2	32h
SM2CTRL2	62h
SM3CTRL2	92h

26.3.4.2 Function

Contains control fields for clock select and forcing output and initialization.This register is write-protected by MCTRL2[WRPROT] bits.

26.3.4.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DBGEN	WAITEN	INDEP	PWM23_INIT	PWM45_INIT	PWMX_INIT	INIT_SEL		FRCEN	0	FORCE_SEL			RELOAD_SEL	CLK_SEL	
W										FORCE						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.4.4 Fields

Field	Function
15 DBGEN	Debug Enable

Table continues on the next page...

Field	Function
	When set to one, the PWM will continue to run while the chip is in debug mode. If the device enters debug mode and this bit is zero, then the PWM outputs will be disabled until debug mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers.
14 WAITEN	Wait Enable When set to one, the PWM will continue to run while the chip is in Wait mode. In this mode, the peripheral clock continues to run but the CPU clock does not. If the device enters Wait mode and this bit is zero, then the PWM outputs will be disabled until Wait mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers.
13 INDEP	Independent or Complementary Pair Operation This bit determines if the PWM_A and PWM_B channels will be independent PWMs or a complementary PWM pair. 0b - PWM_A and PWM_B form a complementary PWM pair. 1b - PWM_A and PWM_B outputs are independent PWMs.
12 PWM23_INIT	PWM23 Initial Value This read/write bit determines the initial value for PWM23 and the value to which it is forced when FORCE_INIT is asserted.
11 PWM45_INIT	PWM45 Initial Value This read/write bit determines the initial value for PWM45 and the value to which it is forced when FORCE_INIT is asserted.
10 PWMX_INIT	PWM_X Initial Value This read/write bit determines the initial value for PWM_X and the value to which it is forced when FORCE_INIT is asserted.
9-8 INIT_SEL	Initialization Control Select These read/write bits control the source of the INIT signal which goes to the counter. 00b - Local sync (PWM_X) causes initialization. 01b - Master reload from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0. The submodule counter will only reinitialize when a master reload occurs. 10b - Master sync from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0. 11b - EXT_SYNC causes initialization.
7 FRCEN	FRCEN This bit allows the CTRL2[FORCE] signal to initialize the counter without regard to the signal selected by CTRL2[INIT_SEL]. This is a software controlled initialization. A forced initialization will also assert the register reload if MCTRL[LDOK] is set. 0b - Initialization from a FORCE_OUT is disabled. 1b - Initialization from a FORCE_OUT is enabled.
6 FORCE	Force Initialization If CTRL2[FORCE_SEL] is set to 000, writing a 1 to this bit results in a FORCE_OUT event. This causes the following actions to be taken: • The PWM_A and PWM_B output pins will assume values based on DTSRCSEL[SMxSEL23] and DTSRCSEL[SMxSEL45]. • If CTRL2[FRCEN] is set, the counter value will be initialized with the INIT register value.
5-3 FORCE_SEL	This read/write bit determines the source of the FORCE OUTPUT signal for this submodule. 000b - The local force signal, CTRL2[FORCE], from this submodule is used to force updates. 001b - The master force signal from submodule 0 is used to force updates. This setting should not be used in submodule 0 as it will hold the FORCE OUTPUT signal to logic 0. 010b - The local reload signal from this submodule is used to force updates without regard to the state of LDOK.

Table continues on the next page...

Field	Function
	011b - The master reload signal from submodule0 is used to force updates if LDOK is set. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0. 100b - The local sync signal from this submodule is used to force updates. 101b - The master sync signal from submodule0 is used to force updates. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0. 110b - The external force signal, EXT_FORCE, from outside the PWM module causes updates. 111b - The external sync signal, EXT_SYNC, from outside the PWM module causes updates.
2 RELOAD_SEL	Reload Source Select This read/write bit determines the source of the RELOAD signal for this submodule. When this bit is set, MCTRL[LDOK[0]] for submodule 0 should be used since the local MCTRL[LDOK] will be ignored. 0b - The local RELOAD signal is used to reload registers. 1b - The master RELOAD signal (from submodule 0) is used to reload registers. This setting should not be used in submodule 0 as it will force the RELOAD signal to logic 0.
1-0 CLK_SEL	Clock Source Select These read/write bits determine the source of the clock signal for this submodule. 00b - The IPBus clock is used as the clock for the local prescaler and counter. 01b - EXT_CLK is used as the clock for the local prescaler and counter. 10b - Submodule 0's clock (AUX_CLK) is used as the source clock for the local prescaler and counter. This setting should not be used in submodule 0 as it will force the clock to logic 0. 11b - reserved

26.3.5 Control Register (SM0CTRL - SM3CTRL)

26.3.5.1 Offset

Register	Offset
SM0CTRL	3h
SM1CTRL	33h
SM2CTRL	63h
SM3CTRL	93h

26.3.5.2 Function

Includes control settings for timing, loading, and buffering.

26.3.5.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LDFQ				HALF	FULL	DT		COMPMODE	PRSC			SPLIT	LDMOD	DBLX	DBLEN
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

26.3.5.4 Fields

Field	Function
15-12 LDFQ	Load Frequency These buffered read/write bits select the PWM load frequency. Reset clears LDFQ, selecting loading every PWM opportunity. A PWM opportunity is determined by HALF and FULL. The register bits are write-protected by MCTRL2[WRPROT] bits. **NOTE:** LDFQ takes effect when the current load cycle is complete, regardless of the state of MCTRL[LDOK]. Reading LDFQ reads the buffered values and not necessarily the values currently in effect. 0000b - Every PWM opportunity 0001b - Every 2 PWM opportunities 0010b - Every 3 PWM opportunities 0011b - Every 4 PWM opportunities 0100b - Every 5 PWM opportunities 0101b - Every 6 PWM opportunities 0110b - Every 7 PWM opportunities 0111b - Every 8 PWM opportunities 1000b - Every 9 PWM opportunities 1001b - Every 10 PWM opportunities 1010b - Every 11 PWM opportunities 1011b - Every 12 PWM opportunities 1100b - Every 13 PWM opportunities 1101b - Every 14 PWM opportunities 1110b - Every 15 PWM opportunities 1111b - Every 16 PWM opportunities
11 HALF	Half Cycle Reload This read/write bit enables half-cycle reloads. A half cycle is defined by when the submodule counter matches the VAL0 register and does not have to be half way through the PWM cycle. This bit is write-protected by MCTRL2[WRPROT] bits. 0b - Half-cycle reloads disabled. 1b - Half-cycle reloads enabled.
10 FULL	Full Cycle Reload

Table continues on the next page...

Field	Function
	This read/write bit enables full-cycle reloads. A full cycle is defined by when the submodule counter matches the VAL1 register. Either CTRL[HALF] or CTRL[FULL] must be set in order to move the buffered data into the registers used by the PWM generators or CTRL[LDMOD] must be set. If both CTRL[HALF] and CTRL[FULL] are set, then reloads can occur twice per cycle. This bit is write-protected by MCTRL2[WRPROT] bits. 0b - Full-cycle reloads disabled. 1b - Full-cycle reloads enabled.
9-8 DT	Deadtime These read only bits reflect the sampled values of the PWM_X input at the end of each deadtime. Sampling occurs at the end of deadtime 0 for DT[0] and the end of deadtime 1 for DT[1]. Reset clears these bits. The register bits are write-protected by MCTRL2[WRPROT] bits.
7 COMPMODE	Compare Mode This bit controls how comparisons are made between the VAL* registers and the PWM submodule counter. This bit can only be written one time after which it requires a reset to release the bit for writing again. 0b - The VAL* registers and the PWM counter are compared using an "equal to" method. This means that PWM edges are only produced when the counter is equal to one of the VAL* register values. This implies that a PWMA output that is high at the end of a period will maintain this state until a match with VAL3 clears the output in the following period. 1b - The VAL* registers and the PWM counter are compared using an "equal to or greater than" method. This means that PWM edges are produced when the counter is equal to or greater than one of the VAL* register values. This implies that a PWMA output that is high at the end of a period could go low at the start of the next period if the starting counter value is greater than (but not necessarily equal to) the new VAL3 value.
6-4 PRSC	Prescaler These buffered read/write bits select the divide ratio of the PWM clock frequency selected by CTRL2[CLK_SEL]. **NOTE:** Reading CTRL[PRSC] reads the buffered values and not necessarily the values currently in effect. CTRL[PRSC] takes effect at the beginning of the next PWM cycle and only when the load okay bit, MCTRL[LDOK], is set or CTRL[LDMOD] is set. This field cannot be written when MCTRL[LDOK] is set. 000b - Prescaler 1. PWM clock frequency = f_{clk} 001b - Prescaler 2. PWM clock frequency = f_{clk} /2 010b - Prescaler 4. PWM clock frequency = f_{clk} /4 011b - Prescaler 8. PWM clock frequency = f_{clk} /8 100b - Prescaler 16. PWM clock frequency = f_{clk} /16 101b - Prescaler 32. PWM clock frequency = f_{clk} /32 110b - Prescaler 64. PWM clock frequency = f_{clk} /64 111b - Prescaler 128. PWM clock frequency = f_{clk} /128
3 SPLIT	Split the DBLPWM signal to PWMA and PWMB This read/write bit is only used in independent mode when DBLEN is set. This bit allows the two PWM pulses generated by DBLEN to be split with one pulse on PWMA and one on PWMB. The two pulses within the same PWM period are created by an XOR function of the PWMA and PWMB sources. The splitting function causes PWMA to output the pulse that occurs when the PWMA source is 1 and the PWMB source is 0. The PWMB output occurs when the PWMB source is 1 and the PWMA source is 0. (See Double switching PWMs .) This bit is write-protected by MCTRL2[WRPROT] bits. 0b - DBLPWM is not split. PWMA and PWMB each have double pulses. 1b - DBLPWM is split to PWMA and PWMB.
2 LDMOD	Load Mode Select This read/write bit selects the timing of loading the buffered registers for this submodule. This bit is write-protected by MCTRL2[WRPROT] bits.

Table continues on the next page...

Field	Function
	0b - Buffered registers of this submodule are loaded and take effect at the next PWM reload if MCTRL[LDOK] is set. 1b - Buffered registers of this submodule are loaded and take effect immediately upon MCTRL[LDOK] being set. In this case it is not necessary to set CTRL[FULL] or CTRL[HALF].
1 DBLX	PWMX Double Switching Enable This read/write bit enables the double switching behavior on PWMX. When this bit is set, the PWMX output shall be the exclusive OR combination of PWMA and PWMB prior to polarity and masking considerations. This bit is write-protected by MCTRL2[WRPROT] bits. 0b - PWMX double pulse disabled. 1b - PWMX double pulse enabled.
0 DBLEN	Double Switching Enable This read/write bit enables the double switching PWM behavior(See Double switching PWMs). Double switching is not compatible with fractional PWM clock generation. Make sure this bit is clear when setting FRCTRL[FRAC23_EN], FRCTRL[FRAC45_EN], or FRCTRL[FRAC1_EN].This bit is write-protected by MCTRL2[WRPROT] bits. 0b - Double switching disabled. 1b - Double switching enabled.

26.3.6 Value Register 0 (SM0VAL0 - SM3VAL0)

26.3.6.1 Offset

Register	Offset
SM0VAL0	5h
SM1VAL0	35h
SM2VAL0	65h
SM3VAL0	95h

26.3.6.2 Function

NOTE

The VAL0 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL0 cannot be written when MCTRL[LDOK] is set. Reading VAL0 reads the value in a buffer. It is not necessarily the value the PWM generator is currently using.

26.3.6.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VAL0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.6.4 Fields

Field	Function
15-0 VAL0	Value Register 0 The 16-bit signed value in this buffered, read/write register defines the mid-cycle reload point for the PWM in PWM clock periods. This value also defines when the PWM_X signal is set and the local sync signal is reset. This register is not byte accessible. **NOTE:** The actual behavior takes affect when counter equal VAL0+1.

26.3.7 Fractional Value Register 1 (SM0FRACVAL1 - SM3FRACVAL1)

26.3.7.1 Offset

Register	Offset
SM0FRACVAL1	6h
SM1FRACVAL1	36h
SM2FRACVAL1	66h
SM3FRACVAL1	96h

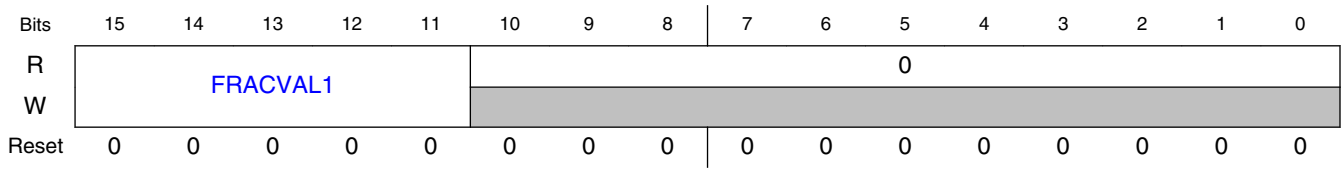
26.3.7.2 Function

NOTE

The FRACVAL1 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL1 cannot be written when MCTRL[LDOK] is set. Reading

FRACVAL1 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

26.3.7.3 Diagram



26.3.7.4 Fields

Field	Function
15-11 FRACVAL1	Fractional Value 1 Register These bits act as a fractional addition to the value in the VAL1 register which controls the PWM period. With fractional delay enabled, PWM works in high-resolution mode. The number of cycles of PWM period = VAL1[15:0] + FRAVAL1[15:11]/32. In case of the maximum value, where VAL1[15:0] = 1111,1111,1111,1111b and FRAVAL1[15:0] = 1111,1xxx,xxxx,xxxxb, is 65535 + 31/32 cycles of IPBus clock.
10-0 —	RESERVED

26.3.8 Value Register 1 (SM0VAL1 - SM3VAL1)

26.3.8.1 Offset

Register	Offset
SM0VAL1	7h
SM1VAL1	37h
SM2VAL1	67h
SM3VAL1	97h

26.3.8.2 Function

NOTE

The VAL1 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL1 cannot be written when MCTRL[LDOK] is set. Reading VAL1 reads the value in a buffer. It is not necessarily the value the PWM generator is currently using.

NOTE

When using FRACVAL1, limit the maximum value of VAL1 to 0xFFFE for unsigned applications or to 0x7FFE for signed applications, to avoid counter rollovers caused by accumulating the fractional period defined by FRACVAL1.

NOTE

If the VAL1 register defines the timer period (Local Sync is selected as the counter initialization signal), a 100% duty cycle cannot be achieved on the PWMX output. After the count reaches VAL1, the PWMX output is low for a minimum of one count every cycle. When the Master Sync signal (only originated by the Local Sync from sub-module 0) is used to control the timer period, the VAL1 register can be free for other functions such as PWM generation without the duty cycle limitation.

26.3.8.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VAL1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.8.4 Fields

Field	Function
15-0 VAL1	Value Register 1

Field	Function
	The 16-bit signed value written to this buffered, read/write register defines the modulo count value (maximum count) for the submodule counter. Upon reaching this count value, the counter reloads itself with the contents of the INIT register and asserts the local sync signal while resetting PWM_X. This register is not byte accessible. **NOTE:** The actual behavior takes affect when counter equal VAL1+1.

26.3.9 Fractional Value Register 2 (SM0FRACVAL2 - SM3FRACVAL2)

26.3.9.1 Offset

Register	Offset
SM0FRACVAL2	8h
SM1FRACVAL2	38h
SM2FRACVAL2	68h
SM3FRACVAL2	98h

26.3.9.2 Function

NOTE

The FRACVAL2 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL2 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL2 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

NOTE

FRCTRL[FRAC23_EN] should be set to 0 when the values of VAL2 and VAL3 cause the high or low time of the PWM output to be 3 cycles or less.

26.3.9.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FRACVAL2					0										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.9.4 Fields

Field	Function
15-11 FRACVAL2	Fractional Value 2 These bits act as a fractional addition to the value in the VAL2 register which controls the PWM_A turn on timing. It is also used to control the fractional addition to the turn off delay of PWM_B when MCTRL[IPOLx]=0 in complementary mode, CTRL2[INDEP]=0. With fractional delay enabled, PWM works in high-resolution mode. The number of cycles of PWM_A turn on delay = VAL2[15:0] + FRAVAL2[15:11]/32. In case of the maximum value, where VAL2[15:0] = 1111,1111,1111,1111b and FRAVAL2[15:0] = 1111,1xxx,xxxx,xxxxb, is 65535 + 31/32 cycles of IPBus clock.
10-0 —	RESERVED

26.3.10 Value Register 2 (SM0VAL2 - SM3VAL2)

26.3.10.1 Offset

Register	Offset
SM0VAL2	9h
SM1VAL2	39h
SM2VAL2	69h
SM3VAL2	99h

26.3.10.2 Function

NOTE

The VAL2 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL2 cannot be written when MCTRL[LDOK] is set. Reading VAL2 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

26.3.10.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VAL2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.10.4 Fields

Field	Function
15-0 VAL2	Value Register 2 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 high. This register is not byte accessible. **NOTE:** The actual behavior takes affect when counter equal VAL2+1.

26.3.11 Fractional Value Register 3 (SM0FRACVAL3 - SM3FRACVAL3)

26.3.11.1 Offset

Register	Offset
SM0FRACVAL3	Ah
SM1FRACVAL3	3Ah
SM2FRACVAL3	6Ah
SM3FRACVAL3	9Ah

26.3.11.2 Function

NOTE

The FRACVAL3 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL3 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL3 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

26.3.11.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FRACVAL3								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.11.4 Fields

Field	Function
15-11 FRACVAL3	Fractional Value 3 These bits act as a fractional addition to the value in the VAL3 register which controls the PWM_A turn off timing. It is also used to control the fractional addition to the turn on delay of PWM_B when MCTRL[IPOLx]=0 in complementary mode, CTRL2[INDEP]=0. **NOTE:** FRCTRL[FRAC23_EN] should be set to 0 when the values of VAL2 and VAL3 cause the high or low time of the PWM output to be 3 cycles or less. With fractional delay enabled, PWM works in high-resolution mode. The number of cycles of PWM_A turn off delay = VAL3[15:0] + FRAVAL3[15:11]/32. In case of the maximum value, where VAL3[15:0] = 1111,1111,1111,1111b and FRAVAL3[15:0] = 1111,1xxx,xxxx,xxxxb, is 65535 + 31/32 cycles of IPBus clock.
10-0 —	RESERVED

26.3.12 Value Register 3 (SM0VAL3 - SM3VAL3)

26.3.12.1 Offset

Register	Offset
SM0VAL3	Bh
SM1VAL3	3Bh
SM2VAL3	6Bh
SM3VAL3	9Bh

26.3.12.2 Function

NOTE

The VAL3 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL3 cannot be written when MCTRL[LDOK] is set. Reading VAL3 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

26.3.12.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VAL3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.12.4 Fields

Field	Function
15-0 VAL3	Value Register 3 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 low. This register is not byte accessible. **NOTE:** The actual behavior takes affect when counter equal VAL3+1.

26.3.13 Fractional Value Register 4 (SM0FRACVAL4 - SM3FRACVAL4)

26.3.13.1 Offset

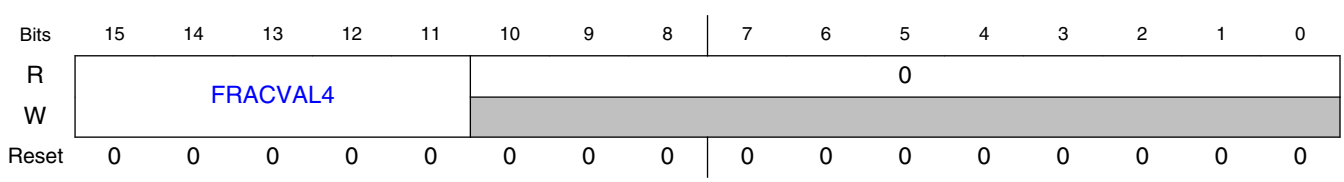
Register	Offset
SM0FRACVAL4	Ch
SM1FRACVAL4	3Ch
SM2FRACVAL4	6Ch
SM3FRACVAL4	9Ch

26.3.13.2 Function

NOTE

The FRACVAL4 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL4 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL4 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

26.3.13.3 Diagram



26.3.13.4 Fields

Field	Function
15-11 FRACVAL4	Fractional Value 4 These bits act as a fractional addition to the value in the VAL4 register which controls the PWM_B turn on timing. It is also used to control the fractional addition to the turn off delay of PWM_A when MCTRL[IPOLx]=1 in complementary mode, CTRL2[INDEP]=0.

Table continues on the next page...

Field	Function
	NOTE: FRCTRL[FRAC45_EN] should be set to 0 when the values of VAL4 and VAL5 cause the high or low time of the PWM output to be 3 cycles or less. With fractional delay enabled, PWM works in high-resolution mode. The number of cycles of PWM_B turn on delay = VAL4[15:0] + FRAVAL4[15:11]/32. In case of the maximum value, where VAL4[15:0] = 1111,1111,1111,1111b and FRAVAL4[15:0] = 1111,1xxx,xxxx,xxxxb, is 65535 + 31/32 cycles of IPBus clock.
10-0 —	RESERVED

26.3.14 Value Register 4 (SM0VAL4 - SM3VAL4)

26.3.14.1 Offset

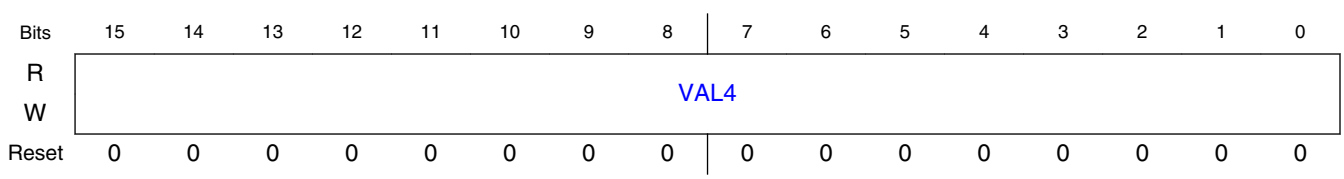
Register	Offset
SM0VAL4	Dh
SM1VAL4	3Dh
SM2VAL4	6Dh
SM3VAL4	9Dh

26.3.14.2 Function

NOTE

The VAL4 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL4 cannot be written when MCTRL[LDOK] is set. Reading VAL4 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

26.3.14.3 Diagram



26.3.14.4 Fields

Field	Function
15-0 VAL4	Value Register 4 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 high. This register is not byte accessible. **NOTE:** The actual behavior takes affect when counter equal VAL4+1.

26.3.15 Fractional Value Register 5 (SM0FRACVAL5 - SM3FRACVAL5)

26.3.15.1 Offset

Register	Offset
SM0FRACVAL5	Eh
SM1FRACVAL5	3Eh
SM2FRACVAL5	6Eh
SM3FRACVAL5	9Eh

26.3.15.2 Function

NOTE

The FRACVAL5 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL5 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL5 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

26.3.15.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FRACVAL5					0										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.15.4 Fields

Field	Function
15-11 FRACVAL5	Fractional Value 5 These bits act as a fractional addition to the value in the VAL5 register which controls the PWM_B turn off timing. It is also used to control the fractional addition to the turn on delay of PWM_A when MCTRL[IPOLx]=1 in complementary mode, CTRL2[INDEP]=0. **NOTE:** FRCTRL[FRAC45_EN] should be set to 0 when the values of VAL4 and VAL5 cause the high or low time of the PWM output to be 3 cycles or less. With fractional delay enabled, PWM works in high-resolution mode. The number of cycles of PWM_B turn off delay = VAL5[15:0] + FRAVAL5[15:11]/32. In case of the maximum value, where VAL5[15:0] = 1111,1111,1111,1111b and FRAVAL5[15:0] = 1111,1xxx,xxxx,xxxxb, is 65535 + 31/32 cycles of IPBus clock.
10-0 —	RESERVED

26.3.16 Value Register 5 (SM0VAL5 - SM3VAL5)

26.3.16.1 Offset

Register	Offset
SM0VAL5	Fh
SM1VAL5	3Fh
SM2VAL5	6Fh
SM3VAL5	9Fh

26.3.16.2 Function

NOTE

The VAL5 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL5 cannot be written when MCTRL[LDOK] is set. Reading VAL5 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

26.3.16.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VAL5															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.16.4 Fields

Field	Function
15-0 VAL5	Value Register 5 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 low. This register is not byte accessible. **NOTE:** The actual behavior takes affect when counter equal VAL5+1.

26.3.17 Fractional Control Register (SM0FRCTRL - SM3FRCTRL)

26.3.17.1 Offset

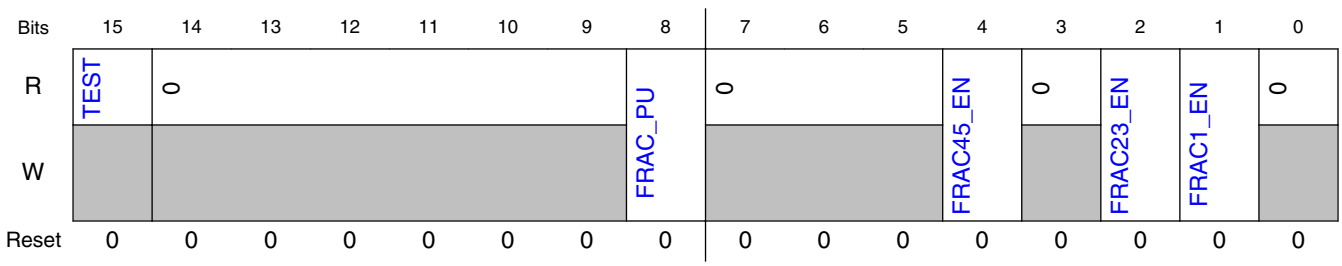
Register	Offset
SM0FRCTRL	10h
SM1FRCTRL	40h
SM2FRCTRL	70h
SM3FRCTRL	A0h

26.3.17.2 Function

NOTE

The FRAC1_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC1_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC1_EN reads the value in a buffer and not necessarily the value the PWM generator is currently using.

26.3.17.3 Diagram



26.3.17.4 Fields

Field	Function
15 TEST	Test Status Bit This is a read only test bit for factory use. This bit will reset to 0 but may be either 0 or 1 during PWM operation.
14-9 —	RESERVED
8 FRAC_PU	Fractional Delay Circuit Power Up This bit is used to power up the fractional delay analog block. The fractional delay block takes 25 us to power up after the first FRAC_PU bit in any submodule is set. The fractional delay block only powers down when the FRAC_PU bits in all submodules are 0. The fractional delay logic can only be used when the IPBus clock is running at 100 MHz. When turned off, fractional placement is disabled. After setting this bit and waiting the 25usec, load the PWM VAL* registers with values to create a PWM output with greater than 0% duty cycle and run for at least one PWM period. This can be done without the outputs enabled and is used to clear the state of the analog block that produces the fractional delays. 0b - Turn off fractional delay logic. 1b - Power up fractional delay logic.
7-5	RESERVED

Table continues on the next page...

Field	Function
—	
4 FRAC45_EN	Fractional Cycle Placement Enable for PWM_B This bit is used to enable the fractional cycle edge placement of PWM_B using the FRACVAL4 and FRACVAL5 registers. When disabled, the fractional cycle edge placement of PWM_B is bypassed. **NOTE:** The FRAC45_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC45_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC45_EN reads the value in a buffer and not necessarily the value the PWM generator is currently using. 0b - Disable fractional cycle placement for PWM_B. 1b - Enable fractional cycle placement for PWM_B.
3 —	RESERVED
2 FRAC23_EN	Fractional Cycle Placement Enable for PWM_A This bit is used to enable the fractional cycle edge placement of PWM_A using the FRACVAL2 and FRACVAL3 registers. When disabled, the fractional cycle edge placement of PWM_A is bypassed. **NOTE:** The FRAC23_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC23_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC23_EN reads the value in a buffer and not necessarily the value the PWM generator is currently using. 0b - Disable fractional cycle placement for PWM_A. 1b - Enable fractional cycle placement for PWM_A.
1 FRAC1_EN	Fractional Cycle PWM Period Enable This bit is used to enable the fractional cycle length of the PWM period using the FRACVAL1 register. When disabled, the fractional cycle length of the PWM period is bypassed. 0b - Disable fractional cycle length for the PWM period. 1b - Enable fractional cycle length for the PWM period.
0 —	RESERVED

26.3.18 Output Control Register (SM0OCTRL - SM3OCTRL)

26.3.18.1 Offset

Register	Offset
SM0OCTRL	11h
SM1OCTRL	41h
SM2OCTRL	71h
SM3OCTRL	A1h

26.3.18.2 Function

Contains output controls for fault states. This register is write-protected by MCTRL2[WRPROT] bits.

26.3.18.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PWMA_IN	PWMB_IN	PWMX_IN	0		POLA	POLB	POLX	0		PWMAFS		PWMBFS		PWMXFS	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.18.4 Fields

Field	Function
15 PWMA_IN	PWM_A Input This read only bit shows the logic value currently being driven into the PWM_A input. The bit's reset state is undefined.
14 PWMB_IN	PWM_B Input This read only bit shows the logic value currently being driven into the PWM_B input. The bit's reset state is undefined.
13 PWMX_IN	PWM_X Input This read only bit shows the logic value currently being driven into the PWM_X input. The bit's reset state is undefined.
12-11 —	RESERVED
10 POLA	PWM_A Output Polarity This bit inverts the PWM_A output polarity. 0b - PWM_A output not inverted. A high level on the PWM_A pin represents the "on" or "active" state. 1b - PWM_A output inverted. A low level on the PWM_A pin represents the "on" or "active" state.
9 POLB	PWM_B Output Polarity This bit inverts the PWM_B output polarity. 0b - PWM_B output not inverted. A high level on the PWM_B pin represents the "on" or "active" state. 1b - PWM_B output inverted. A low level on the PWM_B pin represents the "on" or "active" state.
8	PWM_X Output Polarity

Table continues on the next page...

Field	Function
POLX	This bit inverts the PWM_X output polarity. 0b - PWM_X output not inverted. A high level on the PWM_X pin represents the "on" or "active" state. 1b - PWM_X output inverted. A low level on the PWM_X pin represents the "on" or "active" state.
7-6 —	RESERVED
5-4 PWMAFS	PWM_A Fault State These bits determine the fault state for the PWM_A output during fault conditions and Stop mode. It may also define the output state during Wait and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00b - Output is forced to logic 0 state prior to consideration of output polarity control. 01b - Output is forced to logic 1 state prior to consideration of output polarity control. 10b,11b - Output is tristated.
3-2 PWMBFS	PWM_B Fault State These bits determine the fault state for the PWM_B output during fault conditions and Stop mode. It may also define the output state during Wait and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00b - Output is forced to logic 0 state prior to consideration of output polarity control. 01b - Output is forced to logic 1 state prior to consideration of output polarity control. 10b,11b - Output is tristated.
1-0 PWMXFS	PWM_X Fault State These bits determine the fault state for the PWM_X output during fault conditions and Stop mode. It may also define the output state during Wait and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00b - Output is forced to logic 0 state prior to consideration of output polarity control. 01b - Output is forced to logic 1 state prior to consideration of output polarity control. 10b,11b - Output is tristated.

26.3.19 Status Register (SM0STS - SM3STS)

26.3.19.1 Offset

Register	Offset
SM0STS	12h
SM1STS	42h
SM2STS	72h
SM3STS	A2h

26.3.19.2 Function

Contains Compare and Capture flag status.

26.3.19.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	RUF	REF	RF	CFA1	CFA0	CFB1	CFB0	CFX1	CFX0	CMPF					
W			W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.19.4 Fields

Field	Function
15 —	RESERVED
14 RUF	Registers Updated Flag This read-only flag is set when one of the INIT, VALx, FRACVALx, or CTRL[PRSC] registers has been written, which indicates potentially non-coherent data in the set of double buffered registers. Clear this bit by a proper reload sequence consisting of a reload signal while MCTRL[LDOK] = 1. Reset clears this bit. 0b - No register update has occurred since last reload. 1b - At least one of the double buffered registers has been updated since the last reload.
13 REF	Reload Error Flag This read/write flag is set when a reload cycle occurs while MCTRL[LDOK] is 0 and the double buffered registers are in a non-coherent state (STS[RUF] = 1). Clear this bit by writing a logic one to this location. Reset clears this bit. 0b - No reload error occurred. 1b - Reload signal occurred with non-coherent data and MCTRL[LDOK] = 0.
12 RF	Reload Flag This read/write flag is set at the beginning of every reload cycle regardless of the state of MCTRL[LDOK]. Clear this bit by writing a logic one to this location when DMAEN[VALDE] is clear (non-DMA mode). This flag can also be cleared by the DMA done signal when DMAEN[VALDE] is set (DMA mode) . Reset clears this bit. 0b - No new reload cycle since last STS[RF] clearing 1b - New reload cycle since last STS[RF] clearing
11 CFA1	Capture Flag A1 This bit is set when a capture event occurs on the Capture A1 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CA1DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CA1DE] is set (DMA mode) . Reset clears this bit.
10	Capture Flag A0

Table continues on the next page...

Field	Function
CFA0	This bit is set when a capture event occurs on the Capture A0 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CA0DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CA0DE] is set (DMA mode) . Reset clears this bit.
9 CFB1	Capture Flag B1 This bit is set when a capture event occurs on the Capture B1 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CB1DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CB1DE] is set (DMA mode) . Reset clears this bit.
8 CFB0	Capture Flag B0 This bit is set when a capture event occurs on the Capture B0 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CB0DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CB0DE] is set (DMA mode) . Reset clears this bit.
7 CFX1	Capture Flag X1 This bit is set when a capture event occurs on the Capture X1 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CX1DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CX1DE] is set (DMA mode) . Reset clears this bit.
6 CFX0	Capture Flag X0 This bit is set when a capture event occurs on the Capture X0 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CX0DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CX0DE] is set (DMA mode) . Reset clears this bit.
5-0 CMPF	Compare Flags These bits are set when the submodule counter value matches the value of one of the VALx registers. Clear these bits by writing a 1 to a bit position. 00_0000b - No compare event has occurred for a particular VALx value. 00_0001b - A compare event has occurred for a particular VALx value.

26.3.20 Interrupt Enable Register (SM0INTEN - SM3INTEN)

26.3.20.1 Offset

Register	Offset
SM0INTEN	13h
SM1INTEN	43h
SM2INTEN	73h
SM3INTEN	A3h

26.3.20.2 Function

Contains Compare and Capture interrupt enables.

26.3.20.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		REIE	RIE	CA1IE	CA0IE	CB1IE	CB0IE	CX1IE	CX0IE	CMPIE					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.20.4 Fields

Field	Function
15-14 —	RESERVED
13 REIE	Reload Error Interrupt Enable This read/write bit enables the reload error flag, STS[REF], to generate CPU interrupt requests. Reset clears this bit. 0b - STS[REF] CPU interrupt requests disabled 1b - STS[REF] CPU interrupt requests enabled
12 RIE	Reload Interrupt Enable This read/write bit enables the reload flag, STS[RF], to generate CPU interrupt requests. Reset clears this bit. 0b - STS[RF] CPU interrupt requests disabled 1b - STS[RF] CPU interrupt requests enabled
11 CA1IE	Capture A 1 Interrupt Enable This bit allows the STS[CFA1] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CA1DE]. 0b - Interrupt request disabled for STS[CFA1]. 1b - Interrupt request enabled for STS[CFA1].
10 CA0IE	Capture A 0 Interrupt Enable This bit allows the STS[CFA0] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CA0DE]. 0b - Interrupt request disabled for STS[CFA0]. 1b - Interrupt request enabled for STS[CFA0].
9 CB1IE	Capture B 1 Interrupt Enable This bit allows the STS[CFB1] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CB1DE]. 0b - Interrupt request disabled for STS[CFB1]. 1b - Interrupt request enabled for STS[CFB1].
8 CB0IE	Capture B 0 Interrupt Enable This bit allows the STS[CFB0] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CB0DE]. 0b - Interrupt request disabled for STS[CFB0].

Table continues on the next page...

Field	Function
	1b - Interrupt request enabled for STS[CFB0].
7 CX1IE	Capture X 1 Interrupt Enable This bit allows the STS[CFX1] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CX1DE]. 0b - Interrupt request disabled for STS[CFX1]. 1b - Interrupt request enabled for STS[CFX1].
6 CX0IE	Capture X 0 Interrupt Enable This bit allows the STS[CFX0] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CX0DE]. 0b - Interrupt request disabled for STS[CFX0]. 1b - Interrupt request enabled for STS[CFX0].
5-0 CMPIE	Compare Interrupt Enables These bits enable the STS[CMPF] flags to cause a compare interrupt request to the CPU. 00_0000b - The corresponding STS[CMPF] bit will not cause an interrupt request. 00_0001b - The corresponding STS[CMPF] bit will cause an interrupt request.

26.3.21 DMA Enable Register (SM0DMAEN - SM3DMAEN)

26.3.21.1 Offset

Register	Offset
SM0DMAEN	14h
SM1DMAEN	44h
SM2DMAEN	74h
SM3DMAEN	A4h

26.3.21.2 Function

Contains controls for DMA. This register is write-protected by MCTRL2[WRPROT] bits.

26.3.21.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0						VALDE	FAND	CAPTDE		CA1DE	CA0DE	CB1DE	CB0DE	CX1DE	CX0DE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.21.4 Fields

Field	Function
15-10 —	RESERVED
9 VALDE	Value Registers DMA Enable This read/write bit enables DMA write requests for the VALx and FRACVALx registers when STS[RF] is set. Reset clears this bit. 0b - DMA write requests disabled 1b - Enabled. DMA write requests for the VALx and FRACVALx registers enabled
8 FAND	FIFO Watermark AND Control This read/write bit works in conjunction with the DMAEN[CAPTDE] field when it is set to watermark mode (DMAEN[CAPTDE] = 01). While DMAEN[CAxDE], DMAEN[CBxDE], and DMAEN[CXxDE] determine which FIFO watermarks the DMA read request is sensitive to, this bit determines if the selected watermarks are AND'ed together or OR'ed together in order to create the request. 0b - Selected FIFO watermarks are OR'ed together. 1b - Selected FIFO watermarks are AND'ed together.
7-6 CAPTDE	Capture DMA Enable Source Select These read/write bits select the source of enabling the DMA read requests for the capture FIFOs. Reset clears these bits. 00b - Read DMA requests disabled. 01b - Exceeding a FIFO watermark sets the DMA read request. This requires at least one of DMAEN[CA1DE], DMAEN[CA0DE], DMAEN[CB1DE], DMAEN[CB0DE], DMAEN[CX1DE], or DMAEN[CX0DE] to also be set in order to determine to which watermark(s) the DMA request is sensitive. 10b - A local sync (VAL1 matches counter) sets the read DMA request. 11b - A local reload (STS[RF] being set) sets the read DMA request.
5 CA1DE	Capture A1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture A1 FIFO data when STS[CFA1] is set. Reset clears this bit. Do not set both this bit and INTEN[CA1IE].
4 CA0DE	Capture A0 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture A0 FIFO data when STS[CFA0] is set. Reset clears this bit. Do not set both this bit and INTEN[CA0IE].
3 CB1DE	Capture B1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture B1 FIFO data when STS[CFB1] is set. Reset clears this bit. Do not set both this bit and INTEN[CB1IE].

Table continues on the next page...

Field	Function
2 CB0DE	Capture B0 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture B0 FIFO data when STS[CFB0] is set. Reset clears this bit. Do not set both this bit and INTEN[CB0IE].
1 CX1DE	Capture X1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture X1 FIFO data when STS[CFX1] is set. Reset clears this bit. Do not set both this bit and INTEN[CX1IE].
0 CX0DE	Capture X0 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture X0 FIFO data when STS[CFX0] is set. Reset clears this bit. Do not set both this bit and INTEN[CX0IE].

26.3.22 Output Trigger Control Register (SM0TCTRL - SM3TCTRL)

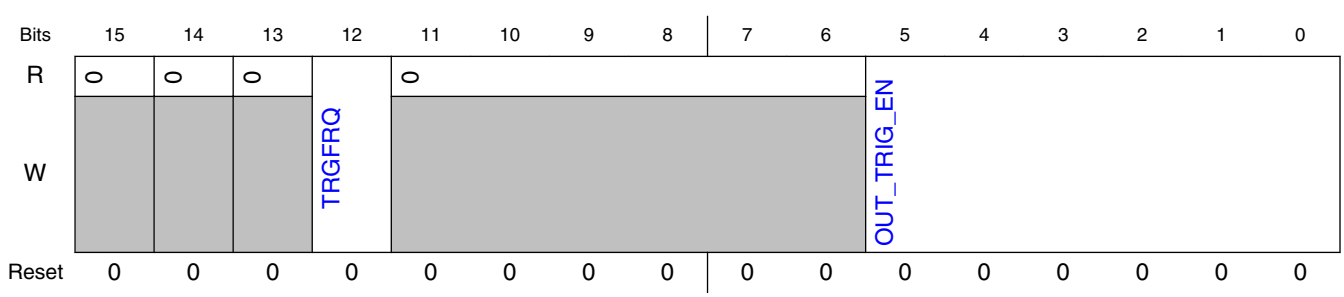
26.3.22.1 Offset

Register	Offset
SM0TCTRL	15h
SM1TCTRL	45h
SM2TCTRL	75h
SM3TCTRL	A5h

26.3.22.2 Function

Contains trigger controls.This register is write-protected by MCTRL2[WRPROT] bits.

26.3.22.3 Diagram



26.3.22.4 Fields

Field	Function
15 —	RESERVED
14 —	RESERVED
13 —	RESERVED
12 TRGFRQ	Trigger frequency This read/write bit allows control over the frequency of the trigger outputs when using non-zero values of CTRL[LDFQ]. 0b - Trigger outputs are generated during every PWM period even if the PWM is not reloaded every period due to CTRL[LDFQ] being non-zero. 1b - Trigger outputs are generated only during the final PWM period prior to a reload opportunity when the PWM is not reloaded every period due to CTRL[LDFQ] being non-zero.
11-6 —	RESERVED
5-0 OUT_TRIG_EN	Output Trigger Enables These bits enable the generation of PWM_OUT_TRIG0 and PWM_OUT_TRIG1 outputs based on the counter value matching the value in one or more of the VAL0-5 registers. **NOTE:** Due to delays in creating the PWM outputs, the output trigger signals will lead the PWM output edges by 2-3 clock cycles depending on the fractional cycle value being used. 1x_xxxxb - PWM_OUT_TRIG1 will set when the counter value matches the VAL5 value. x1_xxxxb - PWM_OUT_TRIG0 will set when the counter value matches the VAL4 value. xx_1xxxb - PWM_OUT_TRIG1 will set when the counter value matches the VAL3 value. xx_x1xxb - PWM_OUT_TRIG0 will set when the counter value matches the VAL2 value. xx_xx1xb - PWM_OUT_TRIG1 will set when the counter value matches the VAL1 value. xx_xxx1b - PWM_OUT_TRIG0 will set when the counter value matches the VAL0 value.

26.3.23 Fault Disable Mapping Register 0 (SM0DISMAP0 - SM3DISMAP0)

26.3.23.1 Offset

Register	Offset
SM0DISMAP0	16h
SM1DISMAP0	46h
SM2DISMAP0	76h
SM3DISMAP0	A6h

26.3.23.2 Function

This register determines which PWM pins are disabled by the fault protection inputs. Reset sets all of the bits in the fault disable mapping register. This register is write-protected by MCTRL2[WRPROT] bits.

26.3.23.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	1				DIS0X				DIS0B				DIS0A			
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

26.3.23.4 Fields

Field	Function
15-12 —	RESERVED
11-8 DIS0X	PWM_X Fault Disable Mask 0 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 0. The PWM_X output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.
7-4 DIS0B	PWM_B Fault Disable Mask 0 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 0. The PWM_B output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.
3-0 DIS0A	PWM_A Fault Disable Mask 0 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 0. The PWM_A output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.

26.3.24 Fault Disable Mapping Register 1 (SM0DISMAP1 - SM3DISMAP1)

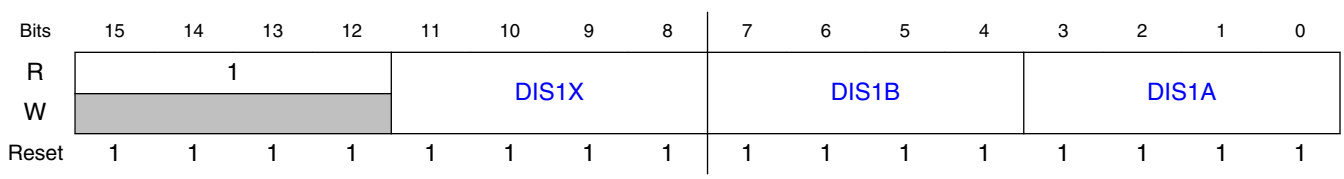
26.3.24.1 Offset

Register	Offset
SM0DISMAP1	17h
SM1DISMAP1	47h
SM2DISMAP1	77h
SM3DISMAP1	A7h

26.3.24.2 Function

This register determines which PWM pins are disabled by the fault protection inputs. Reset sets all of the bits in the fault disable mapping register. This register is write-protected by MCTRL2[WRPROT] bits.

26.3.24.3 Diagram



26.3.24.4 Fields

Field	Function
15-12 —	RESERVED
11-8 DIS1X	PWM_X Fault Disable Mask 1 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 1. The PWM_X output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.
7-4 DIS1B	PWM_B Fault Disable Mask 1 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 1. The PWM_B output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.
3-0 DIS1A	PWM_A Fault Disable Mask 1 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 1. The PWM_A output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.

26.3.25 Deadtime Count Register 0 (SM0DTCNT0 - SM3DTCNT0)

26.3.25.1 Offset

Register	Offset
SM0DTCNT0	18h
SM1DTCNT0	48h
SM2DTCNT0	78h
SM3DTCNT0	A8h

26.3.25.2 Function

Deadtime operation applies only to complementary channel operation. The values written to the DTCNTx registers are in terms of IPBus clock cycles regardless of the setting of CTRL[PRSC] and/or CTRL2[CLK_SEL]. Reset sets the deadtime count registers to a default value of 0x07FF, selecting a deadtime of 2047 IPBus clock cycles, when fractional delay is not enabled. The DTCNTx registers are not byte accessible.

26.3.25.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DTCNT0															
Reset	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

26.3.25.4 Fields

Field	Function
15-0 DTCNT0	DTCNT0 The DTCNT0 field is used to control the deadtime during 0 to 1 transitions of the PWM_A output (assuming normal polarity). The DTCNT0 field is interpreted differently depending on whether the fractional delays are enabled or not.

Field	Function
	• If the fractional delays are disabled (FRCTRL[FRAC23_EN] cleared to 0 when MCTRL[IPOL]=0, or FRCTRL[FRAC45_EN] cleared to 0 when MCTRL[IPOL]=1), then the upper 5 bits of DTCNT0 are ignored and the remaining 11 bits are used to specify the number of PWM clock cycles of deadtime, i.e. the number cycles of deadtime = DTCNT0[10:0] . In this case, the maximum value is 0x07FF which indicates 2047 cycles of deadtime. • If the fractional delays are enabled (FRCTRL[FRAC23_EN] set to 1 when MCTRL[IPOL]=0, or FRCTRL[FRAC45_EN] set to 1 when MCTRL[IPOL]=1), then the upper 11 bits of DTCNT0 represent the number of PWM clock cycles of deadtime, while the lower 5 bits of each register represent the fractional number of cycle. With fractional delay enabled, PWM works in high-resolution mode. The number of cycles of deadtime = DTCNT0[15:5] + DTCNT0[4:0]/32. In this case the maximum value is 0xFFFF which represents 2047 + 31/32 cycles of deadtime.

26.3.26 Deadtime Count Register 1 (SM0DTCNT1 - SM3DTCNT1)

26.3.26.1 Offset

Register	Offset
SM0DTCNT1	19h
SM1DTCNT1	49h
SM2DTCNT1	79h
SM3DTCNT1	A9h

26.3.26.2 Function

Deadtime operation applies only to complementary channel operation. The values written to the DTCNTx registers are in terms of IPBus clock cycles regardless of the setting of CTRL[PRSC] and/or CTRL2[CLK_SEL]. Reset sets the deadtime count registers to a default value of 0x07FF, selecting a deadtime of 2047 IPBus clock cycles, when fractional delay is not enabled. The DTCNTx registers are not byte accessible.

26.3.26.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DTCNT1															
Reset	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

26.3.26.4 Fields

Field	Function
15-0 DTCNT1	DTCNT1 The DTCNT1 field is used to control the deadtime during 0 to 1 transitions of the PWM_B output (assuming normal polarity). The DTCNT1 field is interpreted differently depending on whether the fractional delays are enabled or not. • If the fractional delays are disabled (FRCTRL[FRAC23_EN] cleared to 0 when MCTRL[IPOL]=0, or FRCTRL[FRAC45_EN] cleared to 0 when MCTRL[IPOL]=1), then the upper 5 bits of DTCNT1 are ignored and the remaining 11 bits are used to specify the number of PWM clock cycles of deadtime, i.e. the number cycles of deadtime = DTCNT1[10:0] . In this case, the maximum value is 0x07FF which indicates 2047 cycles of deadtime. • If the fractional delays are enabled (FRCTRL[FRAC23_EN] set to 1 when MCTRL[IPOL]=0, or FRCTRL[FRAC45_EN] set to 1 when MCTRL[IPOL]=1), then the upper 11 bits of DTCNT1 represent the number of PWM clock cycles of deadtime, while the lower 5 bits of each register represent the fractional number of cycle. In this case, the maximum value is 0xFFFF which represents 2047 + 31/32 cycles of deadtime. With fractional delay enabled, PWM works in high-resolution mode. The number of cycles of deadtime = DTCNT1[15:5] + DTCNT1[4:0]/32.

26.3.27 Capture Control A Register (SM0CAPTCTRLA - SM3CAPTCTRLA)

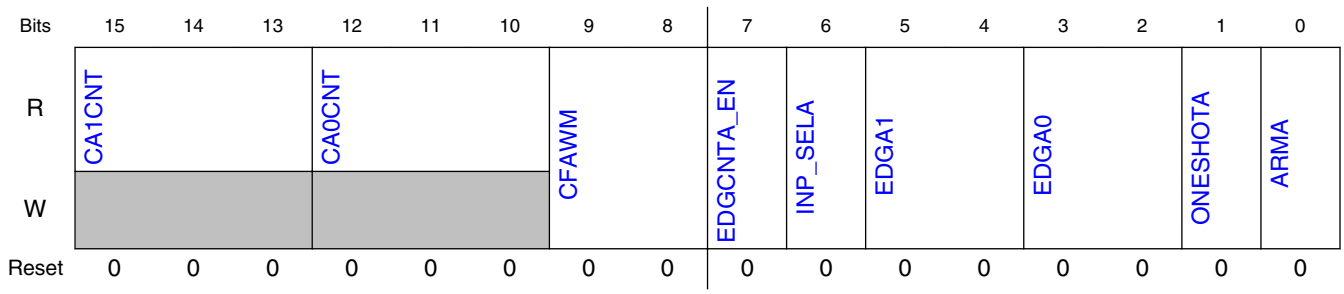
26.3.27.1 Offset

Register	Offset
SM0CAPTCTRLA	1Ah
SM1CAPTCTRLA	4Ah
SM2CAPTCTRLA	7Ah
SM3CAPTCTRLA	AAh

26.3.27.2 Function

Contains capture controls for mode A. This register is write-protected by MCTRL2[WRPROT] bits.

26.3.27.3 Diagram



26.3.27.4 Fields

Field	Function
15-13 CA1CNT	Capture A1 FIFO Word Count This field reflects the number of words in the Capture A1 FIFO. (FIFO depth is 1.)
12-10 CA0CNT	Capture A0 FIFO Word Count This field reflects the number of words in the Capture A0 FIFO. (FIFO depth is 1.)
9-8 CFAWM	Capture A FIFOs Water Mark This field represents the water mark level for capture A FIFOs. The capture flags, STS[CFA1] and STS[CFA0], are not set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTA_EN	Edge Counter A Enable This bit enables the edge counter which counts rising and falling edges on the PWM_A input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6 INP_SELA	Input Select A This bit selects between the raw PWM_A input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_A input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. Note that when this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLA[EDGA0] and CAPTCTRLA[EDGA1] fields are ignored. The software must still place a value other than 00 in either or both of the CAPTCTLRA[EDGA0] and/or CAPTCTRLA[EDGA1] fields in order to enable one or both of the capture registers.
5-4 EDGA1	Edge A 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2 EDGA0	Edge A 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event.

Table continues on the next page...

Field	Function
	00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTA	One Shot Mode A This bit selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1b - One Shot. One shot mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and CAPTCTRLA[ARMA] is cleared. No further captures will be performed until CAPTCTRLA[ARMA] is set again. If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and CAPTCTRLA[ARMA] is then cleared.
0 ARMA	Arm A Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLA[EDGAx] is enabled.

26.3.28 Capture Compare A Register (SM0CAPTCOMPA - SM3CAPTCOMPA)

26.3.28.1 Offset

Register	Offset
SM0CAPTCOMPA	1Bh
SM1CAPTCOMPA	4Bh
SM2CAPTCOMPA	7Bh
SM3CAPTCOMPA	ABh

26.3.28.2 Function

Contains capture and compare values for mode A.This register is write-protected by MCTRL2[WRPROT] bits.

26.3.28.3 Diagram

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

26.3.28.4 Fields

Field	Function
15-8 EDGCNTA	Edge Counter A This read-only field contains the edge counter value for the PWM_A input capture circuitry.
7-0 EDGCMPA	Edge Compare A This read/write field is the compare value associated with the edge counter for the PWM_A input capture circuitry.

26.3.29 Capture Control B Register (SM0CAPTCTRLB - SM3CAPTCTRLB)

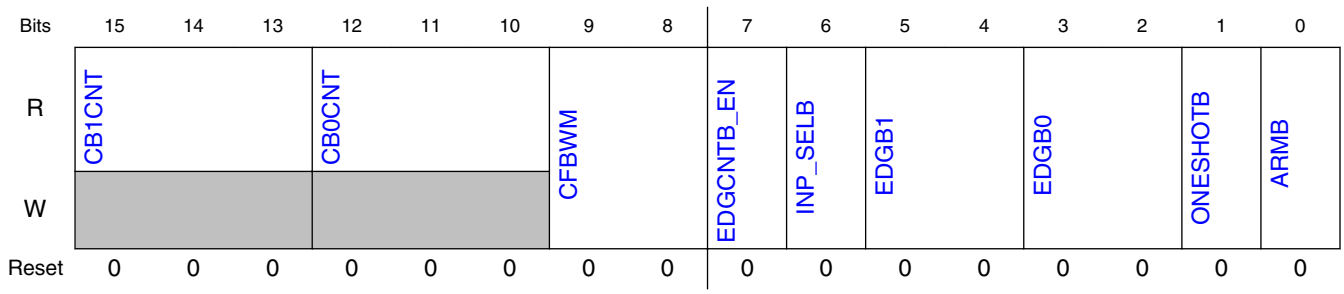
26.3.29.1 Offset

Register	Offset
SM0CAPTCTRLB	1Ch
SM1CAPTCTRLB	4Ch
SM2CAPTCTRLB	7Ch
SM3CAPTCTRLB	ACh

26.3.29.2 Function

Contains capture controls for mode B. This register is write-protected by MCTRL2[WRPROT] bits.

26.3.29.3 Diagram



26.3.29.4 Fields

Field	Function
15-13 CB1CNT	Capture B1 FIFO Word Count This field reflects the number of words in the Capture B1 FIFO. (FIFO depth is 1.)
12-10 CB0CNT	Capture B0 FIFO Word Count This field reflects the number of words in the Capture B0 FIFO. (FIFO depth is 1.)
9-8 CFBWM	Capture B FIFOs Water Mark This field represents the water mark level for capture B FIFOs. The capture flags, STS[CFB1] and STS[CFB0], won't be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTB_EN	Edge Counter B Enable This bit enables the edge counter which counts rising and falling edges on the PWM_B input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6 INP_SELB	Input Select B This bit selects between the raw PWM_B input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_B input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. Note that when this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLB[EDGB0] and CAPTCTRLB[EDGB1] fields are ignored. The software must still place a value other than 00 in either or both of the CAPTCTLRB[EDGB0] and/or CAPTCTRLB[EDGB1] fields in order to enable one or both of the capture registers.
5-4 EDGB1	Edge B 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2 EDGB0	Edge B 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event.

Table continues on the next page...

Field	Function
	00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTB	One Shot Mode B This bit selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1b - One Shot. One shot mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and CAPTCTRLB[ARMB] is cleared. No further captures will be performed until CAPTCTRLB[ARMB] is set again. If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and CAPTCTRLB[ARMB] is then cleared.
0 ARMB	Arm B Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLB[EDGBx] is enabled.

26.3.30 Capture Compare B Register (SM0CAPTCOMPB - SM3CAPTCOMPB)

26.3.30.1 Offset

Register	Offset
SM0CAPTCOMPB	1Dh
SM1CAPTCOMPB	4Dh
SM2CAPTCOMPB	7Dh
SM3CAPTCOMPB	ADh

26.3.30.2 Function

Contains capture and compare values for mode B. This register is write-protected by MCTRL2[WRPROT] bits.

26.3.30.3 Diagram

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

26.3.30.4 Fields

Field	Function
15-8 EDGCNTB	Edge Counter B This read-only field contains the edge counter value for the PWM_B input capture circuitry.
7-0 EDGCMPB	Edge Compare B This read/write field is the compare value associated with the edge counter for the PWM_B input capture circuitry.

26.3.31 Capture Control X Register (SM0CAPTCTRLX - SM3CAPTCTRLX)

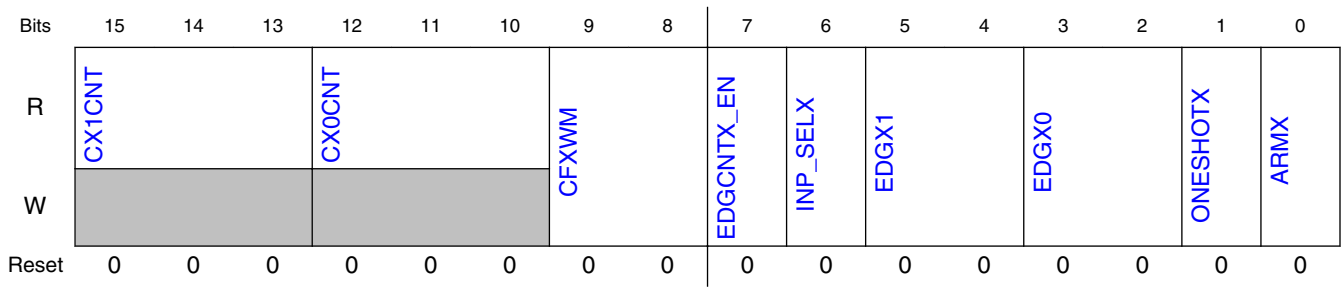
26.3.31.1 Offset

Register	Offset
SM0CAPTCTRLX	1Eh
SM1CAPTCTRLX	4Eh
SM2CAPTCTRLX	7Eh
SM3CAPTCTRLX	AEh

26.3.31.2 Function

Contains capture controls for mode X. This register is write-protected by MCTRL2[WRPROT] bits.

26.3.31.3 Diagram



26.3.31.4 Fields

Field	Function
15-13 CX1CNT	Capture X1 FIFO Word Count This field reflects the number of words in the Capture X1 FIFO. (FIFO depth is 1.)
12-10 CX0CNT	Capture X0 FIFO Word Count This field reflects the number of words in the Capture X0 FIFO. (FIFO depth is 1.)
9-8 CFXWM	Capture X FIFOs Water Mark This field represents the water mark level for capture X FIFOs. The capture flags, STS[CFX1] and STS[CFX0], won't be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTX_EN	Edge Counter X Enable This bit enables the edge counter which counts rising and falling edges on the PWM_X input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6 INP_SELX	Input Select X This bit selects between the raw PWM_X input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_X input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. Note that when this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLX[EDGX0] and CAPTCTRLX[EDGX1] fields are ignored. The software must still place a value other than 00 in either or both of the CAPTCTLRX[EDGX0] and/or CAPTCTRLX[EDGX1] fields in order to enable one or both of the capture registers.
5-4 EDGX1	Edge X 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2 EDGX0	Edge X 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event.

Table continues on the next page...

Field	Function
	00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTX	One Shot Mode Aux This bit selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1b - One Shot. One shot mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and the ARMX bit is cleared. No further captures will be performed until the ARMX bit is set again. If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and the ARMX bit is then cleared.
0 ARMX	Arm X Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLX[EDGXx] is enabled.

26.3.32 Capture Compare X Register (SM0CAPTCOMPX - SM3CAPTCOMPX)

26.3.32.1 Offset

Register	Offset
SM0CAPTCOMPX	1Fh
SM1CAPTCOMPX	4Fh
SM2CAPTCOMPX	7Fh
SM3CAPTCOMPX	AFh

26.3.32.2 Function

Contains capture and control values for mode X. This register is write-protected by MCTRL2[WRPROT] bits.

26.3.32.3 Diagram

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

26.3.32.4 Fields

Field	Function
15-8 EDGCNTX	Edge Counter X This read-only field contains the edge counter value for the PWM_X input capture circuitry.
7-0 EDGCMPX	Edge Compare X This read/write field is the compare value associated with the edge counter for the PWM_X input capture circuitry.

26.3.33 Capture Value 0 Register (SM0CVAL0 - SM3CVAL0)

26.3.33.1 Offset

Register	Offset
SM0CVAL0	20h
SM1CVAL0	50h
SM2CVAL0	80h
SM3CVAL0	B0h

26.3.33.2 Function

Writing this register generates bus transfer error.

26.3.33.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPTVAL0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.33.4 Fields

Field	Function
15-0 CAPTVAL0	CAPTVAL0 This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLX[EDGX0]. Each capture will increase the value of CAPTCTRLX[CX0CNT] by 1 until the maximum value is reached. Each read of this register will decrease the value of CAPTCTRLX[CX0CNT] by 1 until 0 is reached. This register is not byte accessible.

26.3.34 Capture Value 0 Cycle Register (SM0CVAL0CYC - SM3CVAL0CYC)

26.3.34.1 Offset

Register	Offset
SM0CVAL0CYC	21h
SM1CVAL0CYC	51h
SM2CVAL0CYC	81h
SM3CVAL0CYC	B1h

26.3.34.2 Function

Writing this register generates bus transfer error.

26.3.34.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CVAL0CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.34.4 Fields

Field	Function
15-4 —	RESERVED
3-0 CVAL0CYC	CVAL0CYC This read-only register stores the cycle number corresponding to the value captured in CVAL0. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

26.3.35 Capture Value 1 Register (SM0CVAL1 - SM3CVAL1)

26.3.35.1 Offset

Register	Offset
SM0CVAL1	22h
SM1CVAL1	52h
SM2CVAL1	82h
SM3CVAL1	B2h

26.3.35.2 Function

Writing this register generates bus transfer error.

26.3.35.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPTVAL1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.35.4 Fields

Field	Function
15-0 CAPTVAL1	CAPTVAL1 This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLX[EDGX1]. Each capture increases the value of CAPTCTRLX[CX1CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLX[CX1CNT] by 1 until 0 is reached. This register is not byte accessible.

26.3.36 Capture Value 1 Cycle Register (SM0CVAL1CYC - SM3CVAL1CYC)

26.3.36.1 Offset

Register	Offset
SM0CVAL1CYC	23h
SM1CVAL1CYC	53h
SM2CVAL1CYC	83h
SM3CVAL1CYC	B3h

26.3.36.2 Function

Writing this register generates bus transfer error.

26.3.36.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CVAL1CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.36.4 Fields

Field	Function
15-4 —	RESERVED
3-0 CVAL1CYC	CVAL1CYC This read-only register stores the cycle number corresponding to the value captured in CVAL1. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

26.3.37 Capture Value 2 Register (SM0CVAL2 - SM3CVAL2)

26.3.37.1 Offset

Register	Offset
SM0CVAL2	24h
SM1CVAL2	54h
SM2CVAL2	84h
SM3CVAL2	B4h

26.3.37.2 Function

Writing this register generates bus transfer error.

26.3.37.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPTVAL2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.37.4 Fields

Field	Function
15-0 CAPTVAL2	CAPTVAL2 This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLA[EDGA0]. Each capture increases the value of CAPTCTRLA[CA0CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLA[CA0CNT] by 1 until 0 is reached. This register is not byte accessible.

26.3.38 Capture Value 2 Cycle Register (SM0CVAL2CYC - SM3CVAL2CYC)

26.3.38.1 Offset

Register	Offset
SM0CVAL2CYC	25h
SM1CVAL2CYC	55h
SM2CVAL2CYC	85h
SM3CVAL2CYC	B5h

26.3.38.2 Function

Writing this register generates bus transfer error.

26.3.38.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CVAL2CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.38.4 Fields

Field	Function
15-4 —	RESERVED
3-0 CVAL2CYC	CVAL2CYC This read-only register stores the cycle number corresponding to the value captured in CVAL2. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

26.3.39 Capture Value 3 Register (SM0CVAL3 - SM3CVAL3)

26.3.39.1 Offset

Register	Offset
SM0CVAL3	26h
SM1CVAL3	56h
SM2CVAL3	86h
SM3CVAL3	B6h

26.3.39.2 Function

Writing this register generates bus transfer error.

26.3.39.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPTVAL3															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.39.4 Fields

Field	Function
15-0 CAPTVAL3	CAPTVAL3 This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLA[EDGA1]. Each capture increases the value of CAPTCTRLA[CA1CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLA[CA1CNT] by 1 until 0 is reached. This register is not byte accessible.

26.3.40 Capture Value 3 Cycle Register (SM0CVAL3CYC - SM3CVAL3CYC)

26.3.40.1 Offset

Register	Offset
SM0CVAL3CYC	27h
SM1CVAL3CYC	57h
SM2CVAL3CYC	87h
SM3CVAL3CYC	B7h

26.3.40.2 Function

Writing this register generates bus transfer error.

26.3.40.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CVAL3CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.40.4 Fields

Field	Function
15-4 —	RESERVED
3-0 CVAL3CYC	CVAL3CYC This read-only register stores the cycle number corresponding to the value captured in CVAL3. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

26.3.41 Capture Value 4 Register (SM0CVAL4 - SM3CVAL4)

26.3.41.1 Offset

Register	Offset
SM0CVAL4	28h
SM1CVAL4	58h
SM2CVAL4	88h
SM3CVAL4	B8h

26.3.41.2 Function

Writing this register generates bus transfer error.

26.3.41.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPTVAL4															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.41.4 Fields

Field	Function
15-0 CAPTVAL4	CAPTVAL4 This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLB[EDGB0]. Each capture increases the value of CAPTCTRLB[CB0CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLB[CB0CNT] by 1 until 0 is reached. This register is not byte accessible.

26.3.42 Capture Value 4 Cycle Register (SM0CVAL4CYC - SM3CVAL4CYC)

26.3.42.1 Offset

Register	Offset
SM0CVAL4CYC	29h
SM1CVAL4CYC	59h
SM2CVAL4CYC	89h
SM3CVAL4CYC	B9h

26.3.42.2 Function

Writing this register generates bus transfer error.

26.3.42.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CVAL4CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.42.4 Fields

Field	Function
15-4 —	RESERVED
3-0 CVAL4CYC	CVAL4CYC This read-only register stores the cycle number corresponding to the value captured in CVAL4. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

26.3.43 Capture Value 5 Register (SM0CVAL5 - SM3CVAL5)

26.3.43.1 Offset

Register	Offset
SM0CVAL5	2Ah
SM1CVAL5	5Ah
SM2CVAL5	8Ah
SM3CVAL5	BAh

26.3.43.2 Function

Writing this register generates bus transfer error.

26.3.43.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPTVAL5															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.43.4 Fields

Field	Function
15-0 CAPTVAL5	CAPTVAL5 This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLB[EDGB1]. Each capture increases the value of CAPTCTRLB[CB1CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLB[CB1CNT] by 1 until 0 is reached. This register is not byte accessible.

26.3.44 Capture Value 5 Cycle Register (SM0CVAL5CYC - SM3CVAL5CYC)

26.3.44.1 Offset

Register	Offset
SM0CVAL5CYC	2Bh
SM1CVAL5CYC	5Bh
SM2CVAL5CYC	8Bh
SM3CVAL5CYC	BBh

26.3.44.2 Function

Writing this register generates bus transfer error.

26.3.44.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CVAL5CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.44.4 Fields

Field	Function
15-4 —	RESERVED
3-0 CVAL5CYC	CVAL5CYC This read-only register stores the cycle number corresponding to the value captured in CVAL5. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

26.3.45 Capture PWMA Input Filter Register (SM0CAPTFILTA - SM3CAPTFILTA)

26.3.45.1 Offset

Register	Offset
SM0CAPTFILTA	2Dh
SM1CAPTFILTA	5Dh
SM2CAPTFILTA	8Dh
SM3CAPTFILTA	BDh

26.3.45.2 Function

Input filter considerations include:

- The CAPTA_FILT_PER value should be set such that the sampling period is larger than the period of the expected noise. This way a noise spike will only corrupt one sample. The CAPTA_FILT_CNT value should be chosen to reduce the probability of

noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the CAPTA_FILT_CNT+3 power.

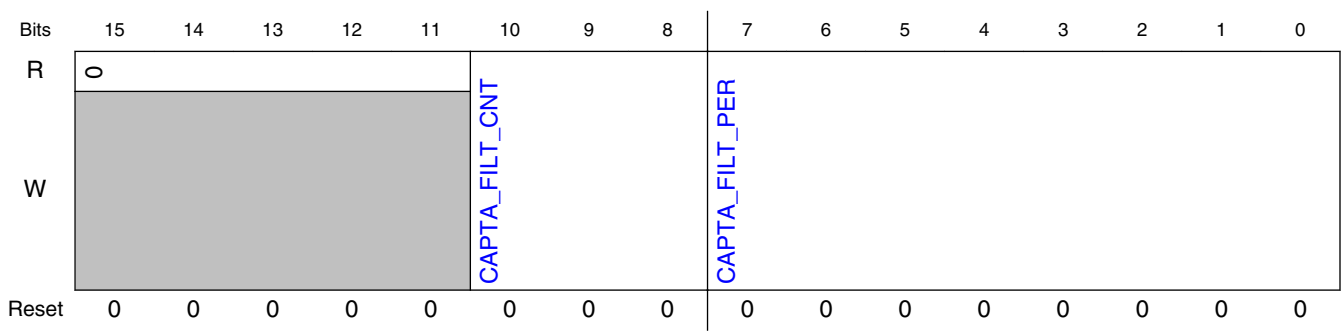
- The values of CAPTA_FILT_PER and CAPTA_FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting CAPTA_FILT_PER to a non-zero value) introduces a latency of ((CAPTA_FILT_CNT+4) x CAPTA_FILT_PER x IPBus clock period).

Note

When the filter is enabled, there is a combinational path to disable the PWM outputs to ensure rapid response to input capture conditions if the PWM module loses its clock. The latency induced by the filter will be seen in the time to set FSTS[FFLAG] and FSTS[FFPIN].

This register is write-protected by MCTRL2[WRPROT] bits.

26.3.45.3 Diagram



26.3.45.4 Fields

Field	Function
15-11 —	RESERVED
10-8 CAPTA_FILT_CNT	Input Capture Filter Count This field represents the number of consecutive samples that must agree prior to the input filter accepting an input transition. The number of samples is the decimal value of this field plus three: the bit field value of 0-7 represents 3-10 samples, respectively. The value of CAPTA_FILT_CNT affects the input latency.
7-0	Input Capture Filter Period This field applies universally to all capture inputs.

Field	Function
CAPTA_FILT_PER	These bits represent the sampling period (in IPBus clock cycles) of the input capture pin input filter. Each input is sampled multiple times at the rate specified by this field. If CAPTA_FILT_PER is 0x00 (default), then the input filter is bypassed. The value of CAPTA_FILT_PER affects the input latency. **NOTE:** When changing values for CAPTA_FILT_PER from one non-zero value to another non-zero value, first write a value of zero to clear the filter.

26.3.46 Capture PWMB Input Filter Register (SM0CAPTFILTB - SM3CAPTFILTB)

26.3.46.1 Offset

Register	Offset
SM0CAPTFILTB	2Eh
SM1CAPTFILTB	5Eh
SM2CAPTFILTB	8Eh
SM3CAPTFILTB	BEh

26.3.46.2 Function

Input filter considerations include:

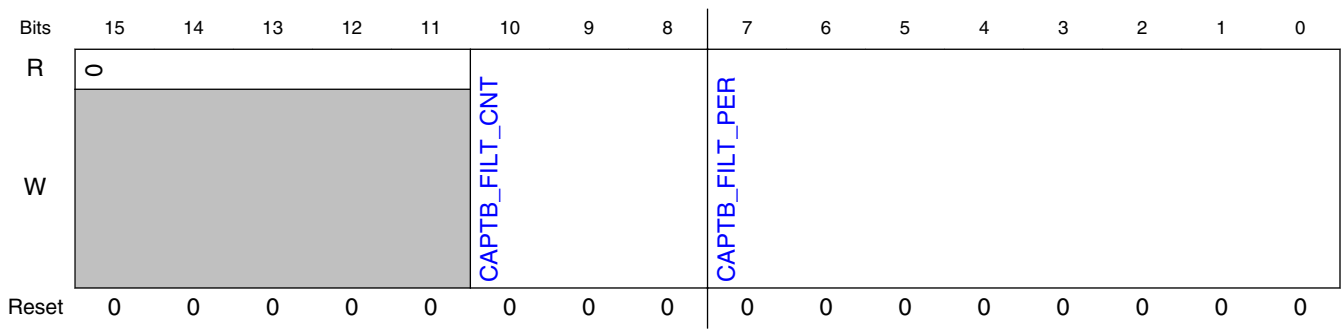
- The CAPTB_FILT_PER value should be set such that the sampling period is larger than the period of the expected noise. This way a noise spike will only corrupt one sample. The CAPTB_FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the CAPTB_FILT_CNT+3 power.
- The values of CAPTB_FILT_PER and CAPTB_FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting CAPTB_FILT_PER to a non-zero value) introduces a latency of ((CAPTB_FILT_CNT+4) x CAPTB_FILT_PER x IPBus clock period).

Note

When the filter is enabled, there is a combinational path to disable the PWM outputs to ensure rapid response to input capture conditions if the PWM module loses its clock. The latency induced by the filter will be seen in the time to set FSTS[FFLAG] and FSTS[FFPIN].

This register is write-protected by MCTRL2[WRPROT] bits.

26.3.46.3 Diagram



26.3.46.4 Fields

Field	Function
15-11 —	RESERVED
10-8 CAPTB_FILT_CNT	Input Capture Filter Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. The number of samples is the decimal value of this field plus three: the bit field value of 0-7 represents 3-10 samples, respectively. The value of CAPTB_FILT_CNT affects the input latency.
7-0 CAPTB_FILT_PER	Input Capture Filter Period This field applies universally to all capture inputs. These bits represent the sampling period (in IPBus clock cycles) of the input capture pin input filter. Each input is sampled multiple times at the rate specified by this field. If CAPTB_FILT_PER is 0x00 (default), then the input filter is bypassed. The value of CAPTB_FILT_PER affects the input latency. **NOTE:** When changing values for CAPTB_FILT_PER from one non-zero value to another non-zero value, first write a value of zero to clear the filter.

26.3.47 Capture PWMX Input Filter Register (SM0CAPTFILTX - SM3CAPTFILTX)

26.3.47.1 Offset

Register	Offset
SM0CAPTFILTX	2Fh
SM1CAPTFILTX	5Fh
SM2CAPTFILTX	8Fh
SM3CAPTFILTX	BFh

26.3.47.2 Function

Input filter considerations include:

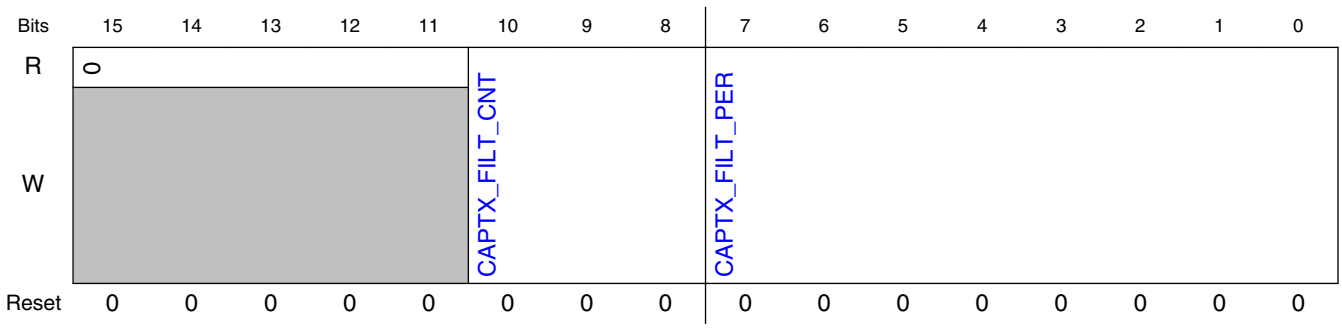
- The CAPTX_FILT_PER value should be set such that the sampling period is larger than the period of the expected noise. This way a noise spike will only corrupt one sample. The CAPTX_FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the CAPTX_FILT_CNT+3 power.
- The values of FILT_PER and CAPTX_FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting CAPTX_FILT_PER to a non-zero value) introduces a latency of ((CAPTX_FILT_CNT+4) x CAPTX_FILT_PER x IPBus clock period).

Note

When the filter is enabled, there is a combinational path to disable the PWM outputs to ensure rapid response to input capture conditions if the PWM module loses its clock. The latency induced by the filter will be seen in the time to set FSTS[FFLAG] and FSTS[FFPIN].

This register is write-protected by MCTRL2[WRPROT] bits.

26.3.47.3 Diagram



26.3.47.4 Fields

Field	Function
15-11 —	RESERVED
10-8 CAPTX_FILT_CNT	Input Capture Filter Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. The number of samples is the decimal value of this field plus three: the bit field value of 0-7 represents 3-10 samples, respectively. The value of CAPTX_FILT_CNT affects the input latency.
7-0 CAPTX_FILT_PER	Input Capture Filter Period This field applies universally to all capture inputs. These bits represent the sampling period (in IPBus clock cycles) of the input capture pin input filter. Each input is sampled multiple times at the rate specified by this field. If CAPTX_FILT_PER is 0x00 (default), then the input filter is bypassed. The value of CAPTX_FILT_PER affects the input latency. **NOTE:** When changing values for CAPTX_FILT_PER from one non-zero value to another non-zero value, first write a value of zero to clear the filter.

26.3.48 Phase Delay Register (SM1PHASEDLY - SM3PHASEDLY)

26.3.48.1 Offset

Register	Offset
SM1PHASEDLY	5Ch
SM2PHASEDLY	8Ch
SM3PHASEDLY	BCh

26.3.48.2 Function

The 16-bit unsigned value in this buffered, read/write register defines the delay from the master sync signal of submodule 0 to the time that this submodule recognizes the master sync in PWM clock periods. CTRL2[INIT_SEL] must be set to 10b in order to select the master sync signal as the source for initialization when using this register. Setting this register with a non-zero value and using the master sync signal as the initialization source, allows the output of this submodule to be a fixed number of cycles delayed from submodule 0. For PWM operation, the buffered contents of this register are updated at the start of every PWM cycle. This register is not byte accessible.

NOTE

The PHASEDLY register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. This register cannot be written when MCTRL[LDOK] is set. Reading PHASEDLY reads the value in a buffer and not necessarily the value the PWM generator is currently using. Also note, the value of this register should not be set to a value larger than the period defined in submodule 0.

26.3.48.3 Diagram

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

26.3.48.4 Fields

Field	Function
15-0 PHASEDLY	Initial Count Register Bits

26.3.49 Output Enable Register (OUTEN)

26.3.49.1 Offset

Register	Offset
OUTEN	C0h

26.3.49.2 Function

Contains PWM output enables. This register is write-protected by MCTRL2[WRPROT] bits.

26.3.49.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				PWMA_EN				PWMB_EN				PWMX_EN			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.49.4 Fields

Field	Function
15-12 —	RESERVED
11-8 PWMA_EN	PWM_A Output Enables The four bits of this field enable the PWM_A outputs of submodules 3-0, respectively. These bits should be set to 0 (output disabled) when a PWM_A pin is being used for input capture. 0000b - PWM_A output disabled. 0001b - PWM_A output enabled.
7-4 PWMB_EN	PWM_B Output Enables The four bits of this field enable the PWM_B outputs of submodules 3-0, respectively. These bits should be set to 0 (output disabled) when a PWM_B pin is being used for input capture. 0000b - PWM_B output disabled. 0001b - PWM_B output enabled.
3-0 PWMX_EN	PWM_X Output Enables

Field	Function
	The four bits of this field enable the PWM_X outputs of submodules 3-0, respectively. These bits should be set to 0 (output disabled) when a PWM_X pin is being used for input capture or deadtime correction. 0000b - PWM_X output disabled. 0001b - PWM_X output enabled.

26.3.50 Mask Register (MASK)

26.3.50.1 Offset

Register	Offset
MASK	C1h

26.3.50.2 Function

MASK is double buffered and does not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading MASK reads the buffered values and not necessarily the values currently in effect. This double buffering can be overridden by setting the UPDATE_MASK bits.

26.3.50.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					MASKA				MASKB				MASKX			
W	UPDATE_MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.50.4 Fields

Field	Function
15-12 UPDATE_MASK	Update Mask Bits Immediately The four bits mask the PWM_X outputs of submodules 3-0, respectively, The four bits of this field force the MASK* bits to be immediately updated within submodules 3-0, respectively, without waiting for a FORCE_OUT event. These self-clearing bits always read as zero. Software may write to any or all of

Table continues on the next page...

Field	Function
	these bits and may set these bits in the same write operation that updates the MASKA, MASKB, and MASKX fields of this register. 0000b - Normal operation. MASK* bits within the corresponding submodule are not updated until a FORCE_OUT event occurs within the submodule. 0001b - Immediate operation. MASK* bits within the corresponding submodule are updated on the following clock edge after setting this bit.
11-8 MASKA	PWM_A Masks The four bits of this field mask the PWM_A outputs of submodules 3-0, respectively, forcing the output to logic 0 prior to consideration of the output polarity. 0000b - PWM_A output normal. 0001b - PWM_A output masked.
7-4 MASKB	PWM_B Masks The four bits of this field mask the PWM_B outputs of submodules 3-0, respectively, forcing the output to logic 0 prior to consideration of the output polarity. 0000b - PWM_B output normal. 0001b - PWM_B output masked.
3-0 MASKX	PWM_X Masks The four bits of this field mask the PWM_X outputs of submodules 3-0, respectively, forcing the output to logic 0 prior to consideration of the output polarity. 0000b - PWM_X output normal. 0001b - PWM_X output masked.

26.3.51 Software Controlled Output Register (SWCOUT)

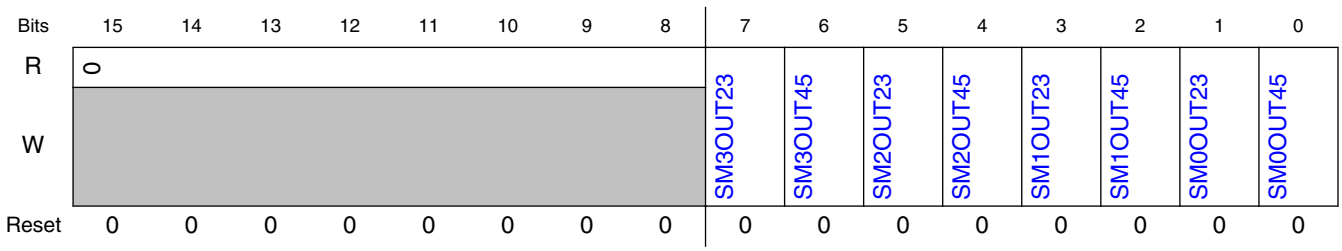
26.3.51.1 Offset

Register	Offset
SWCOUT	C2h

26.3.51.2 Function

These bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading these bits reads the buffered value and not necessarily the value currently in effect.

26.3.51.3 Diagram



26.3.51.4 Fields

Field	Function
15-8 —	RESERVED
7 SM3OUT23	Submodule 3 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM3SEL23] is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM23. 1b - A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM23.
6 SM3OUT45	Submodule 3 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM3SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM45. 1b - A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM45.
5 SM2OUT23	Submodule 2 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM2SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM23. 1b - A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM23.
4 SM2OUT45	Submodule 2 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM2SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM45. 1b - A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM45.
3 SM1OUT23	Submodule 1 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM1SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM23. 1b - A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM23.
2 SM1OUT45	Submodule 1 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM1SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule.

Table continues on the next page...

Field	Function
	0b - A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM45. 1b - A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM45.
1 SM0OUT23	Submodule 0 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM0SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM23. 1b - A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM23.
0 SM0OUT45	Submodule 0 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM0SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM45. 1b - A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM45.

26.3.52 PWM Source Select Register (DTSRCSEL)

26.3.52.1 Offset

Register	Offset
DTSRCSEL	C3h

26.3.52.2 Function

The PWM source select bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading these bits reads the buffered value and not necessarily the value currently in effect. This register is write-protected by MCTRL2[WRPROT] bits.

26.3.52.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SM3SEL23		SM3SEL45		SM2SEL23		SM2SEL45		SM1SEL23		SM1SEL45		SM0SEL23		SM0SEL45	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.52.4 Fields

Field	Function
15-14 SM3SEL23	Submodule 3 PWM23 Control Select This field selects possible over-rides to the generated SM3PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM3PWM23 signal is used by the deadtime logic. 01b - Inverted generated SM3PWM23 signal is used by the deadtime logic. 10b - SWCOUT[SM3OUT23] is used by the deadtime logic. 11b - PWM3_EXTA signal is used by the deadtime logic.
13-12 SM3SEL45	Submodule 3 PWM45 Control Select This field selects possible over-rides to the generated SM3PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM3PWM45 signal is used by the deadtime logic. 01b - Inverted generated SM3PWM45 signal is used by the deadtime logic. 10b - SWCOUT[SM3OUT45] is used by the deadtime logic. 11b - PWM3_EXTB signal is used by the deadtime logic.
11-10 SM2SEL23	Submodule 2 PWM23 Control Select This field selects possible over-rides to the generated SM2PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM2PWM23 signal is used by the deadtime logic. 01b - Inverted generated SM2PWM23 signal is used by the deadtime logic. 10b - SWCOUT[SM2OUT23] is used by the deadtime logic. 11b - PWM2_EXTA signal is used by the deadtime logic.
9-8 SM2SEL45	Submodule 2 PWM45 Control Select This field selects possible over-rides to the generated SM2PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM2PWM45 signal is used by the deadtime logic. 01b - Inverted generated SM2PWM45 signal is used by the deadtime logic. 10b - SWCOUT[SM2OUT45] is used by the deadtime logic. 11b - PWM2_EXTB signal is used by the deadtime logic.
7-6 SM1SEL23	Submodule 1 PWM23 Control Select This field selects possible over-rides to the generated SM1PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM1PWM23 signal is used by the deadtime logic. 01b - Inverted generated SM1PWM23 signal is used by the deadtime logic. 10b - SWCOUT[SM1OUT23] is used by the deadtime logic. 11b - PWM1_EXTA signal is used by the deadtime logic.
5-4 SM1SEL45	Submodule 1 PWM45 Control Select This field selects possible over-rides to the generated SM1PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM1PWM45 signal is used by the deadtime logic. 01b - Inverted generated SM1PWM45 signal is used by the deadtime logic. 10b - SWCOUT[SM1OUT45] is used by the deadtime logic. 11b - PWM1_EXTB signal is used by the deadtime logic.
3-2 SM0SEL23	Submodule 0 PWM23 Control Select

Table continues on the next page...

Field	Function
	This field selects possible over-rides to the generated SM0PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM0PWM23 signal is used by the deadtime logic. 01b - Inverted generated SM0PWM23 signal is used by the deadtime logic. 10b - SWCOUT[SM0OUT23] is used by the deadtime logic. 11b - PWM0_EXTA signal is used by the deadtime logic.
1-0 SM0SEL45	Submodule 0 PWM45 Control Select This field selects possible over-rides to the generated SM0PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM0PWM45 signal is used by the deadtime logic. 01b - Inverted generated SM0PWM45 signal is used by the deadtime logic. 10b - SWCOUT[SM0OUT45] is used by the deadtime logic. 11b - PWM0_EXTB signal is used by the deadtime logic.

26.3.53 Master Control Register (MCTRL)

26.3.53.1 Offset

Register	Offset
MCTRL	C4h

26.3.53.2 Function

In every 4-bit field in this register, each bit acts on a separate submodule. Accordingly, the description of every bit field refers to the effect of an individual bit.

26.3.53.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IPOL				RUN				0				LDOK			
W									CLDOK							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.53.4 Fields

Field	Function
15-12 IPOL	Current Polarity The four buffered read/write bits of this field correspond to submodules 3-0, respectively. Each bit selects between PWM23 and PWM45 as the source for the generation of the complementary PWM pair output for the corresponding submodule. MCTRL[IPOL] is ignored in independent mode. MCTRL[IPOL] does not take effect until a FORCE_OUT event takes place in the appropriate submodule. Reading MCTRL[IPOL] reads the buffered value and not necessarily the value currently in effect. 0000b - PWM23 is used to generate complementary PWM pair in the corresponding submodule. 0001b - PWM45 is used to generate complementary PWM pair in the corresponding submodule.
11-8 RUN	Run The four read/write bits of this field enable the clocks to the PWM generator of submodules 3-0, respectively. The corresponding MCTRL[RUN] bit must be set for each submodule that is using its input capture functions or is using the local reload as its reload source. When this bit equals zero, the submodule counter is reset and PWM outputs are held. A reset clears this field. 0000b - PWM counter is stopped, but PWM outputs will hold the current state. 0001b - PWM counter is started in the corresponding submodule.
7-4 CLDOK	Clear Load Okay The 4 bits of CLDOK field correspond to submodules 3-0, respectively. Each write-only bit is used to clear the corresponding bit of MCTRL[LDOK]. Write a 1 to CLDOK to clear the corresponding MCTRL[LDOK] bit. If a reload occurs within a submodule with the corresponding MCTRL[LDOK] bit set at the same time that MCTRL[CLDOK] is written, then the reload in that submodule will not be performed and MCTRL[LDOK] will be cleared. CLDOK bit is self-clearing and always reads as a 0.
3-0 LDOK	Load Okay The 4 bits of LDOK field correspond to submodules 3-0, respectively. Each read/set bit loads CTRL[PRSC] and the INIT, FRACVALx, and VALx registers of the corresponding submodule into a set of buffers. The buffered prescaler divisor, submodule counter modulus value, and PWM pulse width take effect at the next PWM reload if CTRL[LDMOD] is clear or immediately if CTRL[LDMOD] is set. The VALx, FRACVALx, INIT, and CTRL[PRSC] registers of the corresponding submodule cannot be written while the corresponding MCTRL[LDOK] bit is set. In Master Reload Mode (CTRL2[RELOAD_SEL]=1), it is only necessary to set the LDOK bit corresponding to submodule0; however, it is recommended to also set the LDOK bit of the slave submodules, to prevent unwanted writes to the registers in the slave submodules. The MCTRL[LDOK] bit is automatically cleared after the new values are loaded, or it can be manually cleared before a reload by writing a logic 1 to the appropriate MCTRL[CLDOK] bit. LDOK bits cannot be written with a zero. MCTRL[LDOK] can be set in DMA mode when the DMA indicates that it has completed the update of all CTRL[PRSC], INIT, FRACVALx, and VALx registers in the corresponding submodule. Reset clears LDOK field. 0000b - Do not load new values. 0001b - Load prescaler, modulus, and PWM values of the corresponding submodule.

26.3.54 Master Control 2 Register (MCTRL2)

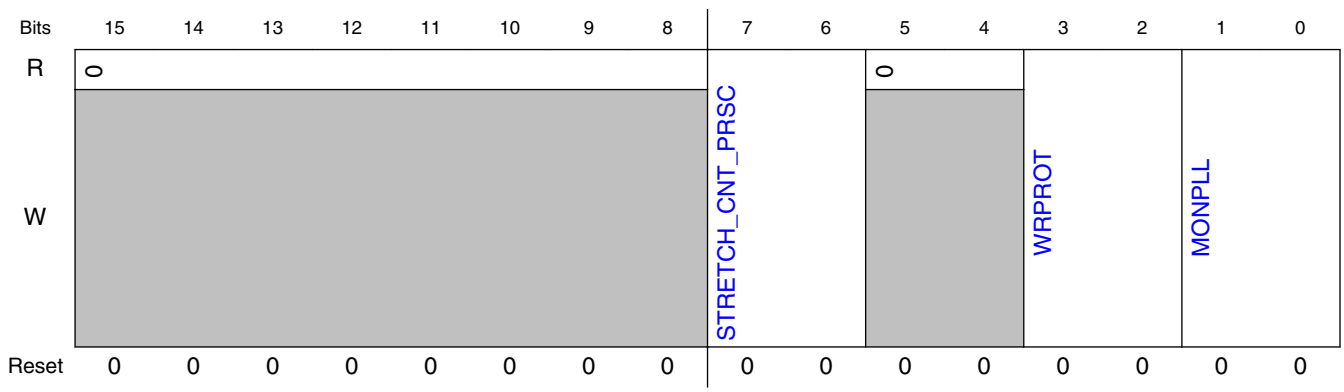
26.3.54.1 Offset

Register	Offset
MCTRL2	C5h

26.3.54.2 Function

Includes control for monitoring the PLL state and write protection of some configuration registers.

26.3.54.3 Diagram



26.3.54.4 Fields

Field	Function
15-8 —	RESERVED
7-6 STRETCH_CNT_PRSC	Stretch IPBus clock count prescaler for mux0_trig/mux1_trig/out0_trig/out1_trig/pwma_trig/pwmb_trig If user config eFlexPWM work in fast clk mode(use SoC level register, eFlexPWM input signal fast_clk_mode is high), then user can use these bits to stretch output signals mux0_trig/mux1_trig/ out0_trig/out1_trig/pwma_trig/pwmb_trig. 00b - Stretch count is zero, no stretch . 01b - Stretch mux0_trig/mux1_trig/out0_trig/out1_trig/pwma_trig/pwmb_trig for 2 IPBus clock period. 10b - Stretch mux0_trig/mux1_trig/out0_trig/out1_trig/pwma_trig/pwmb_trig for 4 IPBus clock period. 11b - Stretch mux0_trig/mux1_trig/out0_trig/out1_trig/pwma_trig/pwmb_trig for 8 IPBus clock period.

Table continues on the next page...

Field	Function
5-4 —	RESERVED
3-2 WRPROT	Write protect Enable write protection of some configuration registers of eFlexPWM. 00b - Write protection off (default). 01b - Write protection on. 10b - Write protection off and locked until chip reset. 11b - Write protection on and locked until chip reset.
1-0 MONPLL	Monitor PLL State These bits are used to control disabling of the fractional delay block when the chip PLL is unlocked and/or missing its input reference. The fractional delay block requires a continuous 200 MHz clock from the PLL. If this clock turns off when the fractional delay block is being used, then the output of the fractional delay block can be stuck high or low even if the PLL restarts. When this control bit is set, PLL problems cause the fractional delay block to be disabled until the PLL returns to a locked state. Once the PLL is receiving a proper reference and is locked, the fractional delay block requires a 25 µs startup time just as if the FRCTRL[FRAC*_EN] bits had been turned off and turned on again. If PLL monitoring is disabled, then software should manually clear and then set the FRCTRL[FRAC*_EN] bits when the PLL loses its reference or loses lock. This will cause the fractional delay block to be disabled and restarted. If the fractional delay block is not being used, then the value of these bits do not matter. 00b - Not locked. Do not monitor PLL operation. Resetting of the fractional delay block in case of PLL losing lock will be controlled by software. 01b - Not locked. Monitor PLL operation to automatically disable the fractional delay block when the PLL encounters problems. 10b - Locked. Do not monitor PLL operation. Resetting of the fractional delay block in case of PLL losing lock will be controlled by software. These bits are write protected until the next reset. 11b - Locked. Monitor PLL operation to automatically disable the fractional delay block when the PLL encounters problems. These bits are write protected until the next reset.

26.3.55 Fault Control Register (FCTRL0 - FCTRL1)

26.3.55.1 Offset

Register	Offset
FCTRL0	C6h
FCTRL1	CCh

26.3.55.2 Function

For every 4-bit field in this register, the bits act on the fault inputs in order. For example, FLVL bits 15-12 act on faults 3-0, respectively. This register is write-protected by MCTRL2[WRPROT] bits.

26.3.55.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FLVL				FAUTO				FSAFE				FIE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.55.4 Fields

Field	Function
15-12 FLVL	Fault Level The four read/write bits of this field select the active logic level of the individual fault inputs 3-0, respectively. A reset clears this field. 0000b - A logic 0 on the fault input indicates a fault condition. 0001b - A logic 1 on the fault input indicates a fault condition.
11-8 FAUTO	Automatic Fault Clearing The four read/write bits of this field select automatic or manual clearing of faults 3-0, respectively. A reset clears this field. 0000b - Manual fault clearing. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear at the start of a half cycle or full cycle depending on the states of FSTS[FHALF] and FSTS[FFULL]. If neither FFULL nor FHALF is set, then the fault condition cannot be cleared. This is further controlled by FCTRL[FSAFE]. 0001b - Automatic fault clearing. PWM outputs disabled by this fault are enabled when FSTS[FFPINx] is clear at the start of a half cycle or full cycle depending on the states of FSTS[FHALF] and FSTS[FFULL] without regard to the state of FSTS[FFLAGx]. If neither FFULL nor FHALF is set, then the fault condition cannot be cleared.
7-4 FSAFE	Fault Safety Mode These read/write bits select the safety mode during manual fault clearing. A reset clears this field. FSTS[FFPINx] may indicate a fault condition still exists even though the actual fault signal at the FAULTx pin is clear due to the fault filter latency. 0000b - Normal mode. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear at the start of a half cycle or full cycle depending on the states of FSTS[FHALF] and FSTS[FFULL] without regard to the state of FSTS[FFPINx]. If neither FHALF nor FFULL is set then the fault condition cannot be cleared. The PWM outputs disabled by this fault input will not be re-enabled until the actual FAULTx input signal de-asserts since the fault input will combinationally disable the PWM outputs (as programmed in DISMAPn).

Table continues on the next page...

Field	Function
	0001b - Safe mode. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear and FSTS[FFPINx] is clear at the start of a half cycle or full cycle depending on the states of FSTS[FHALF] and FSTS[FFULL]. If neither FHLAF nor FFULL is set, then the fault condition cannot be cleared.
3-0 FIE	Fault Interrupt Enables This read/write field enables CPU interrupt requests generated by the FAULTx pins. A reset clears this field. **NOTE:** The fault protection circuit is independent of the FIEx bit and is always active. If a fault is detected, the PWM outputs are disabled according to the disable mapping register. 0000b - FAULTx CPU interrupt requests disabled. 0001b - FAULTx CPU interrupt requests enabled.

26.3.56 Fault Status Register (FSTS0 - FSTS1)

26.3.56.1 Offset

Register	Offset
FSTS0	C7h
FSTS1	CDh

26.3.56.2 Function

Includes controls related to fault conditions.

26.3.56.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FHALF				FFPIN				FFULL				FFLAG			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.3.56.4 Fields

Field	Function
15-12	Half Cycle Fault Recovery

Table continues on the next page...

Field	Function
FHALF	These read/write bits are used to control the timing for re-enabling the PWM outputs after a fault condition. These bits apply to both automatic and manual clearing of a fault condition. These register bits are write-protected by MCTRL2[WRPROT] bits. **NOTE:** Both FHALF and FFULL can be set so that the fault recovery occurs at the start of a full cycle and at the start of a half cycle (as defined by VAL0). If neither FHALF nor FFULL is set, then no fault recovery is possible. 0000b - PWM outputs are not re-enabled at the start of a half cycle. 0001b - PWM outputs are re-enabled at the start of a half cycle (as defined by VAL0).
11-8 FFPIN	Filtered Fault Pins These read-only bits reflect the current state of the filtered FAULTx pins converted to high polarity. A logic 1 indicates a fault condition exists on the filtered FAULTx pin. A reset has no effect on this field. After the system reset de-asserts, these are the possible values of FFPIN: If FCTRL[FLVL] = 0 and FAULTx = 0, then FFPIN is set to 1. If FCTRL[FLVL] = 0 and FAULTx = 1, then FFPIN is kept as 0. If FCTRL[FLVL] = 1 and FAULTx = 0, then FFPIN is kept as 0. If FCTRL[FLVL] = 1 and FAULTx = 1, then FFPIN is set to 1.
7-4 FFULL	Full Cycle These read/write bits are used to control the timing for re-enabling the PWM outputs after a fault condition. These bits apply to both automatic and manual clearing of a fault condition. These register bits are write-protected by MCTRL2[WRPROT] bits. **NOTE:** Both FHALF and FFULL can be set so that the fault recovery occurs at the start of a full cycle and at the start of a half cycle (as defined by VAL0). If neither FHALF nor FFULL is set, then no fault recovery is possible. 0000b - PWM outputs are not re-enabled at the start of a full cycle 0001b - PWM outputs are re-enabled at the start of a full cycle
3-0 FFLAG	Fault Flags These read-only flags are set within two CPU cycles after a transition to active on the FAULTx pin. Clear this bit by writing a logic one to it. A reset clears this field. While the reset value is 0, these bits may be set to 1 by the time they can be read depending on the state of the fault input signals. After the system reset de-asserts, these are the possible values of FFLAG: If FCTRL[FLVL] = 0 and FAULTx = 0, then FFLAG is set to 1. If FCTRL[FLVL] = 0 and FAULTx = 1, then FFLAG is kept as 0. If FCTRL[FLVL] = 1 and FAULTx = 0, then FFLAG is kept as 0. If FCTRL[FLVL] = 1 and FAULTx = 1, then FFLAG is set to 1. 0000b - No fault on the FAULTx pin. 0001b - Fault on the FAULTx pin.

26.3.57 Fault Filter Register (FFILT0 - FFILT1)

26.3.57.1 Offset

Register	Offset
FFILT0	C8h
FFILT1	CEh

26.3.57.2 Function

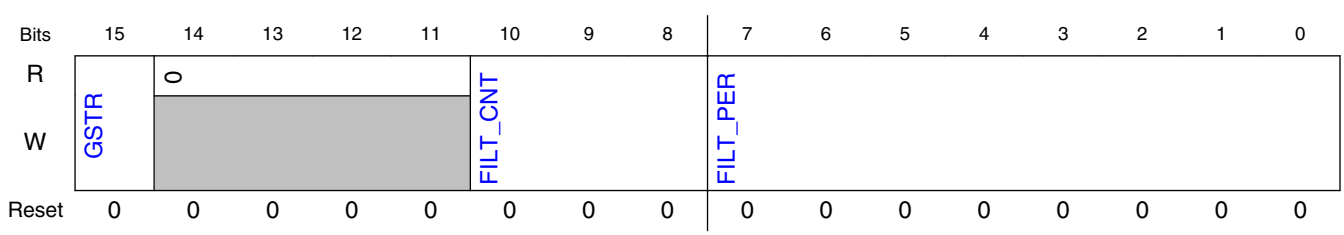
The settings in this register are shared among each of the fault input filters within the fault channel.

Input filter considerations include:

- The FILT_PER value should be set such that the sampling period is larger than the period of the expected noise. This way a noise spike will only corrupt one sample. The FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the FILT_CNT+3 power.
- The values of FILT_PER and FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting FILT_PER to a non-zero value) introduces a latency of ((FILT_CNT+4) x FILT_PER x IPBus clock period). Note that even when the filter is enabled, there is a combinational path to disable the PWM outputs. This is to ensure rapid response to fault conditions and also to ensure fault response if the PWM module loses its clock. The latency induced by the filter will be seen in the time to set FSTS[FFLAG] and FSTS[FFPIN].

This register is write-protected by MCTRL2[WRPROT] bits.

26.3.57.3 Diagram



26.3.57.4 Fields

Field	Function
15 GSTR	Fault Glitch Stretch Enable This bit is used to enable the fault glitch-stretching logic. This logic ensures that narrow fault glitches are stretched to be at least 2 IPBus clock cycles wide. In some cases a narrow fault input can cause problems due to the short PWM output shutdown/re-activation time. The stretching logic ensures that a glitch on the fault input, when the fault filter is disabled, will be registered in the fault flags. 0b - Fault input glitch stretching is disabled. 1b - Input fault signals will be stretched to at least 2 IPBus clock cycles.
14-11 —	RESERVED
10-8 FILT_CNT	Fault Filter Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. The number of samples is the decimal value of this field plus three: the bit field value of 0-7 represents 3-10 samples, respectively. The value of FILT_CNT affects the input latency.
7-0 FILT_PER	Fault Filter Period This 8-bit field applies universally to all fault inputs. These bits represent the sampling period (in IPBus clock cycles) of the fault pin input filter. Each input is sampled multiple times at the rate specified by this field. If FILT_PER is 0x00 (default), then the input filter is bypassed. The value of FILT_PER affects the input latency. **NOTE:** When changing values for FILT_PER from one non-zero value to another non-zero value, first write a value of zero to clear the filter.

26.3.58 Fault Test Register (FTST0 - FTST1)

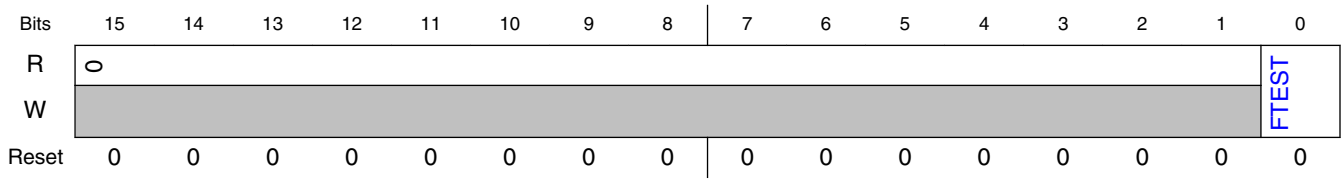
26.3.58.1 Offset

Register	Offset
FTST0	C9h
FTST1	CFh

26.3.58.2 Function

Contains FTEST field for fault simulation.

26.3.58.3 Diagram



26.3.58.4 Fields

Field	Function
15-1 —	RESERVED
0 FTEST	Fault Test This read/write bit is used to simulate a fault condition. Setting this bit causes a simulated fault to be sent into all of the fault filters. The condition propagates to the fault flags and possibly the PWM outputs depending on the DISMAPn settings. Clearing this bit removes the simulated fault condition. This register bit is write-protected by MCTRL2[WRPROT] bits. 0b - No fault 1b - Cause a simulated fault

26.3.59 Fault Control 2 Register (FCTRL20 - FCTRL21)

26.3.59.1 Offset

Register	Offset
FCTRL20	CAh
FCTRL21	D0h

26.3.59.2 Function

Controls combinational link from fault inputs to PWM outputs. This register is write-protected by MCTRL2[WRPROT] bits.

26.3.59.3 Diagram

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

26.3.59.4 Fields

Field	Function
15-4 —	RESERVED
3-0 NOCOMB	No Combinational Path From Fault Input To PWM Output This read/write field is used to control the combinational path from the fault inputs to the PWM outputs. When these bits are low (default), the corresponding fault inputs have a combinational path to the PWM outputs that are sensitive to these fault inputs (as defined by DISMAP0 and DISMAP1). This combinational path is a safety feature that ensures the output is disabled even if the SOC has a failure of its clocking system. The combinational path also means that a pulse on the fault input can cause a brief disable of the PWM output even if the fault pulse is not wide enough to get through the input filter and be latched in the fault logic. Setting these bits removes the combinational path and uses the filtered and latched fault signals as the fault source to disable the PWM outputs. This eliminates fault glitches from creating PWM output glitches but also increases the latency to respond to a real fault. 0000b - There is a combinational link from the fault inputs to the PWM outputs. The fault inputs are combined with the filtered and latched fault signals to disable the PWM outputs. 0001b - The direct combinational path from the fault inputs to the PWM outputs is disabled and the filtered and latched fault signals are used to disable the PWM outputs.

26.4 Functional description

26.4.1 PWM submodule

The following figure shows the PWM Submodule Block Diagram.

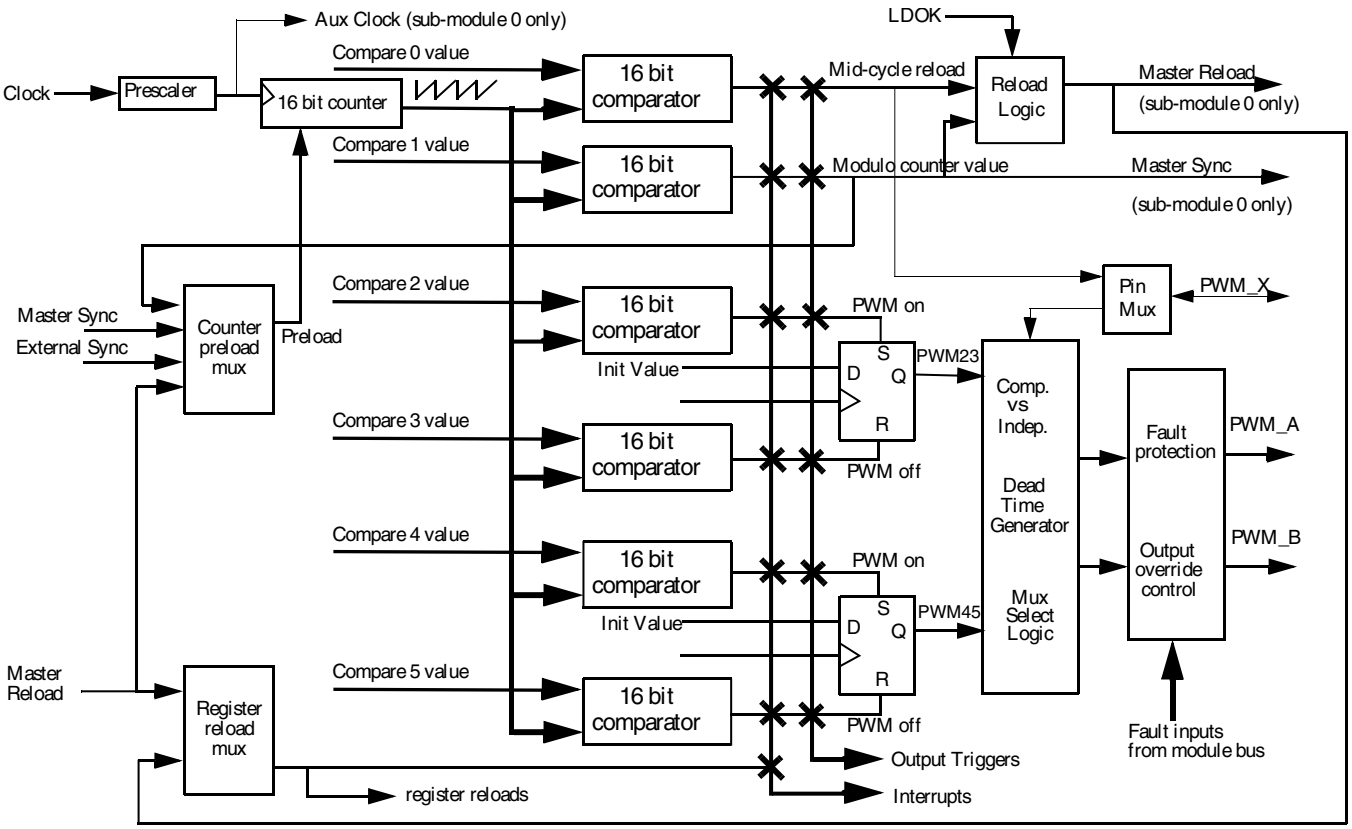


Figure 26-2. PWM submodule block diagram

26.4.2 PWM capabilities

This section describes some capabilities of the PWM module.

26.4.2.1 Center aligned PWMs

Each submodule has its own timer that is capable of generating PWM signals on two output pins. The edges of each of these signals are controlled independently as shown in Figure 26-3.

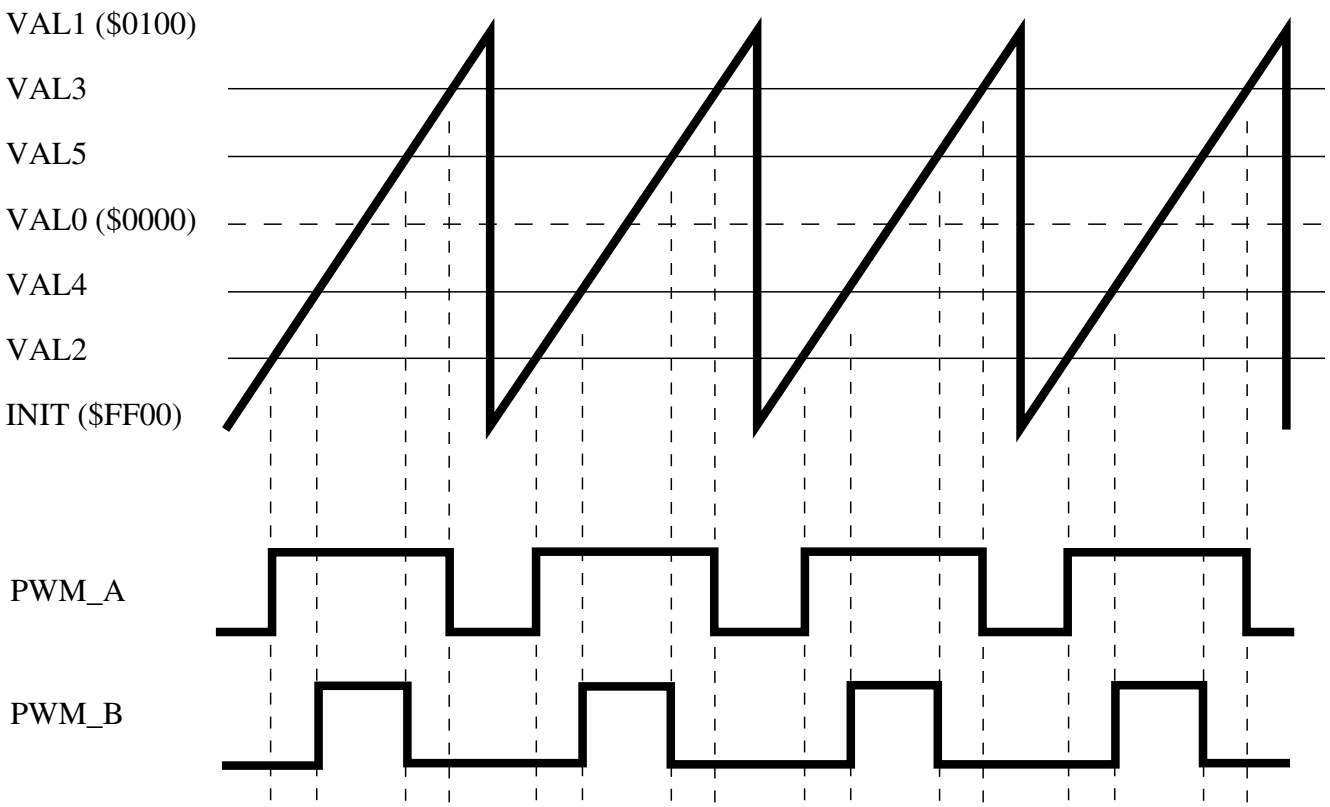


Figure 26-3. Center aligned PWM example

The submodule timers only count in the up direction and then reset to the INIT value. Instead of having a single value that determines pulse width, there are two values that must be specified: the turn-on edge and the turn-off edge. This double-action edge generation provides the user control over the pulse width and also the relative alignment of the signal. As a result, there is no need to support separate PWM alignment modes since the PWM alignment mode is inherently a function of the turn-on and turn-off edge values.

Figure 26-3 also illustrates an additional enhancement to the PWM generation process. When the counter resets, it is reloaded with a user-specified value, which may or may not be zero. If the value chosen happens to be the 2's complement of the modulus value, then the PWM generator operates in "signed" mode. This means if each PWM's turn-on and turn-off edge values are same in numbers but different in their sign, the "on" portion of the output signal is centered around a count value of zero. Therefore, only one PWM value is calculated in software and then this value and its negative are provided to the submodule as the turn-off and turn-on edges respectively. This technique results in a pulse width consists of an odd number of timer counts. If all PWM signal edge calculations follow this convention, then the signals will be center aligned with each other, which is the goal. The center alignment between the signals is not restricted to

symmetry around the zero count value, as any other number would also work. However, centering on zero provides the greatest range in signed mode and also simplifies the calculations.

26.4.2.2 Edge aligned PWMs

Figure 26-4 shows the results of edge aligned operation when the turn-on edge for each pulse is specified to be the INIT value. Therefore, only the turn-off edge value needs to be periodically updated to change the pulse width.

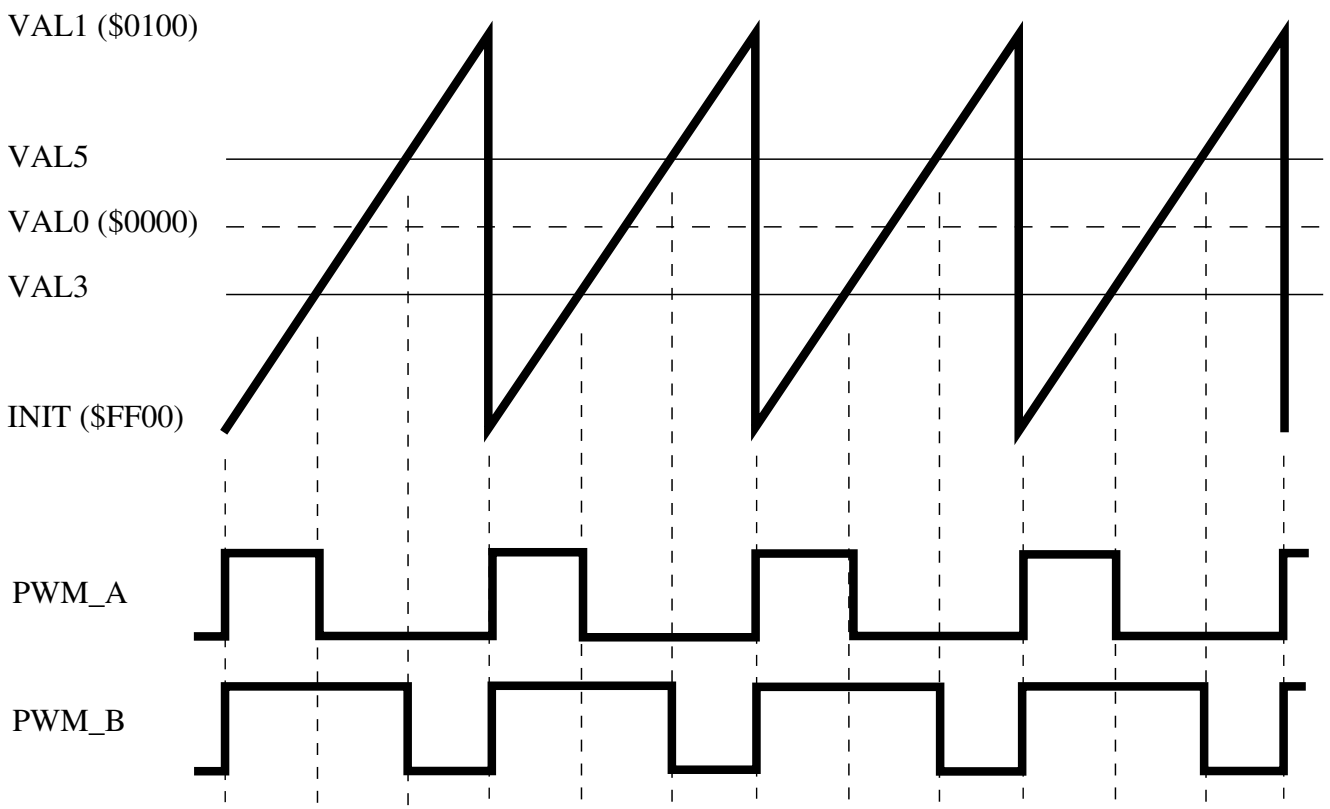


Figure 26-4. Edge aligned example (INIT=VAL2=VAL4)

With edge aligned PWMs, another example of the benefits of signed mode can be seen. Use "bipolar" PWMs to drive an H-bridge, where a 50% duty cycle results in zero volts on the load. Duty cycles less than 50% generate negative load voltages and duty cycles greater than 50% generate positive load voltages. If the module is set to signed mode operation (the INIT and VAL1 values are the same number with opposite signs), then there is a direct proportionality between the PWM turn-off edge value and the motor inverter voltage, including the sign. Therefore, signed mode of operation simplifies the software interface to the PWM module since no offset calculations are required to translate the output variable control algorithm to the voltage on an H-Bridge load.

26.4.2.3 Phase shifted PWMs

In the previous sections, the benefits of the signed mode of operation were discussed in the context of simplifying the required software calculations by eliminating the requirement to bias up signed variables before applying them to the module. However, if numerical biases are applied to the turn-on and turn-off edges of different PWM signals, the signals will be phase shifted to each other, as shown in Figure 26-5. This results in certain advantages when applied to a power stage. For example, when operating a multi-phase inverter at a low modulation index, all of the PWM switching edges from the different phases occur at the same time. This can be troublesome from a noise standpoint, especially if ADC readings of the inverter must be scheduled near those times. Phase shifting the PWM signals can open up timing windows between the switching edges to allow a signal to be sampled by the ADC. However, phase shifting does not affect the duty cycle so average load voltage is not affected.

If the outputs of submodules 1-3 need to be delayed from the output of submodule 0 (and from each other), instead of just creating a phase delay by adding an offset to the turn on and turn off times of the different submodules, another method is to use the PHASEDLY registers for submodules 1-3 to indicate their delay from the submodule 0 timing. This method can be used when the master sync signal from submodule 0 is selected as the initialization source (CTRL2[INIT_SEL]==b10). This method allows all of the submodules to be programmed with the same turn on and turn off time but submodules 1-3 can still be delayed the time from submodule 0.

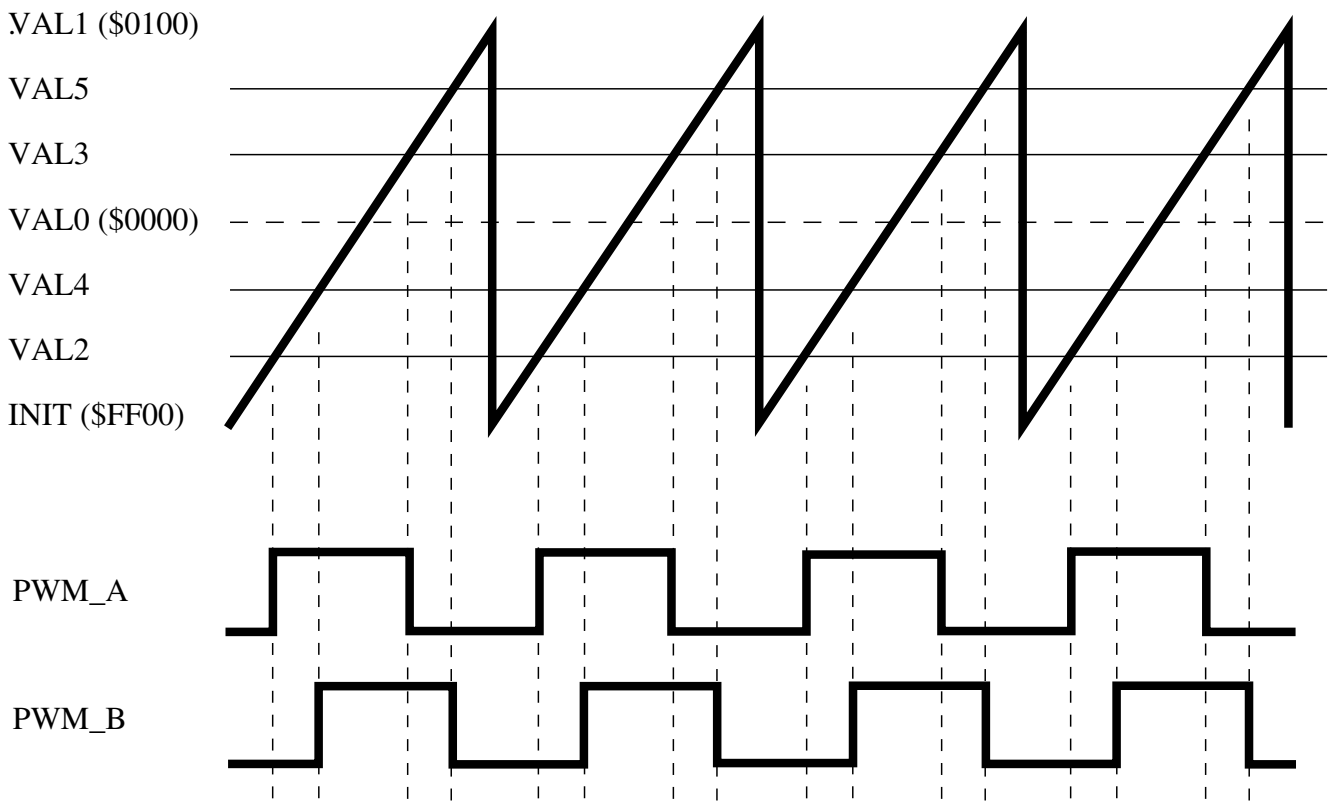


Figure 26-5. Phase shifted outputs - example for 1 submodule

An additional benefit of phase shifted PWMs is shown in Figure 26-6. In this case, an H-Bridge circuit is driven by 4 PWM signals to control the voltage waveform on the primary of a transformer. Both left and right side PWMs are configured to generate a square wave with 50% duty cycle. This works for the H-Bridge since no narrow pulse widths are generated reducing the high frequency switching requirements of the transistors. The RMS value of this waveform is directly controlled by the amount of phase shift of the square waves. Regardless of the phase shift, no DC component appears in the load voltage as long as the duty cycle of each square wave remains at 50% suitable for transformer loads. As a result, this topology is frequently used in industrial welders to adjust the amount of energy delivered to the weld arc.

Note

The square wave on the right side of the H-Bridge is phase shifted compared to the left side of the H-Bridge. As a result, the transformer primary sees the bottom waveform across its terminals.

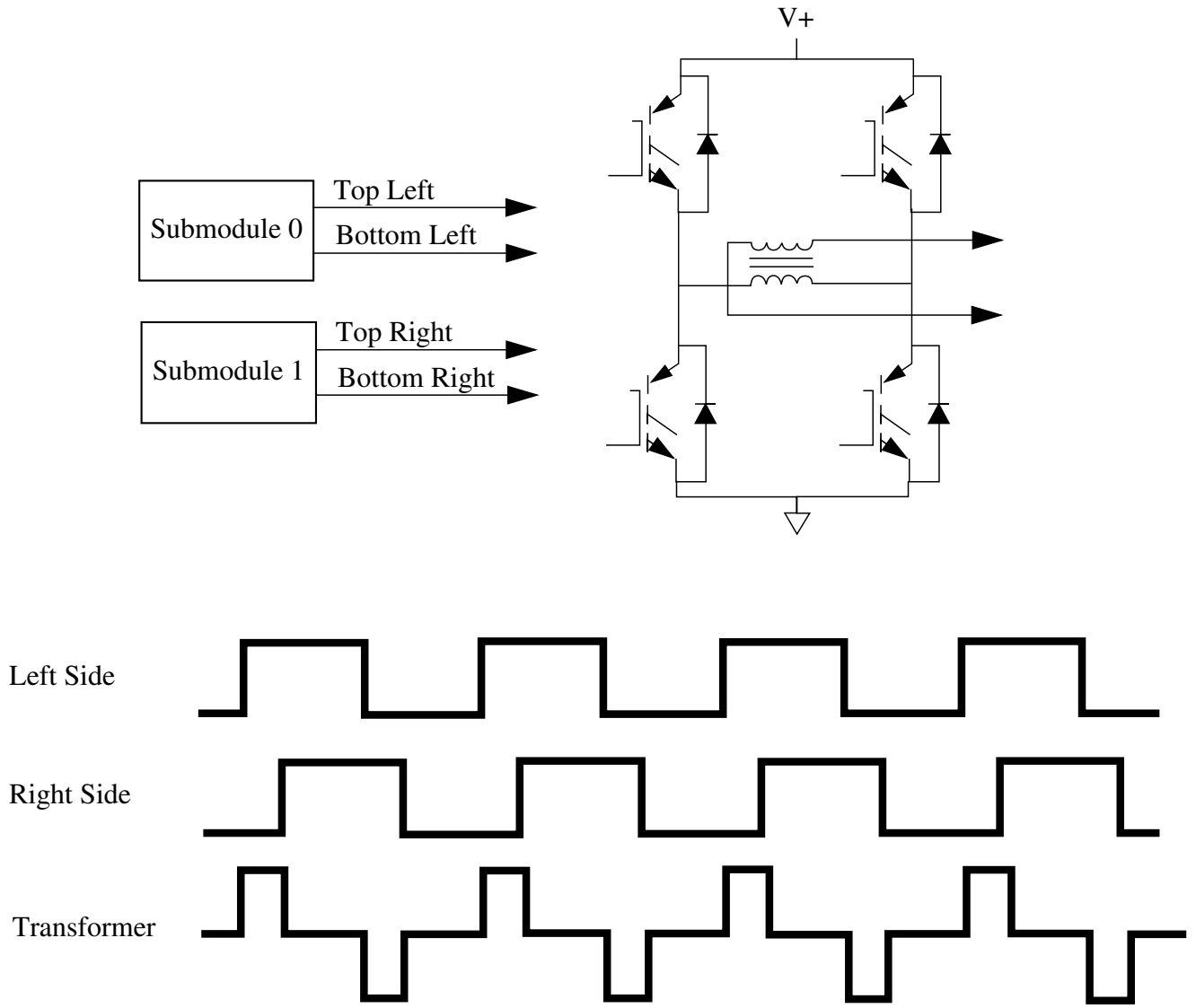


Figure 26-6. Phase Shifted PWMs Applied to a Transformer Primary

26.4.2.4 Double switching PWMs

Double switching PWM output is supported to aid in single shunt current measurement and three-phase reconstruction. This method supports two independent rising edges and two independent falling edges per PWM cycle. The VAL2 and VAL3 registers are used to generate the even channel (labeled as PWM_A in Figure 26-7) while VAL4 and VAL5 are used to generate the odd channel. The two channels (PWM23 or PWM_A and PWM45 or PWM_B from force out logic) are combined using XOR logic (force out logic) as the following figure shows. The DBLPWM signal can be run through the deadtime insertion logic.

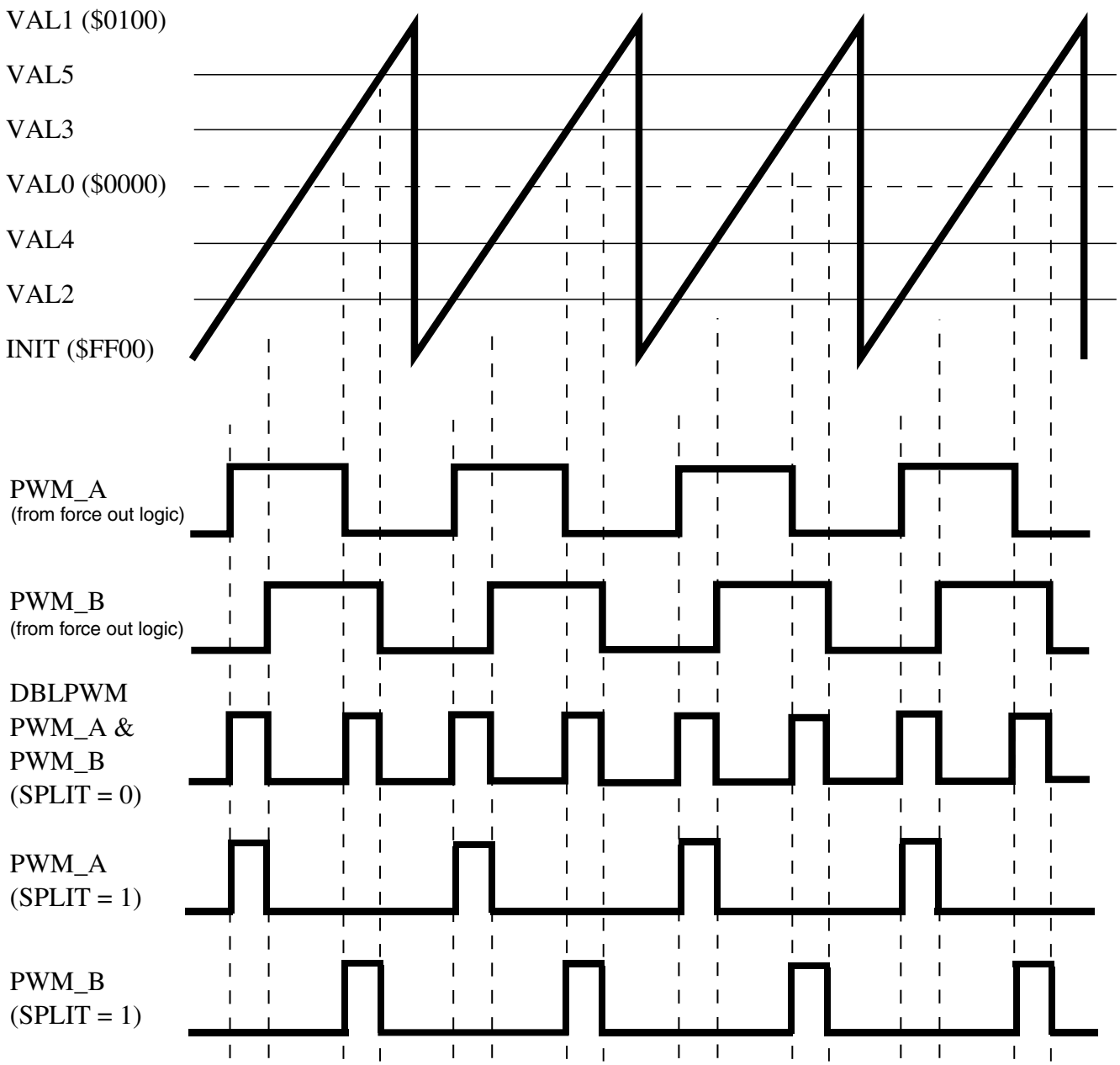


Figure 26-7. Double switching output example

26.4.2.5 ADC triggering

In cases where the timing of the ADC triggering is critical, it must be scheduled as a hardware event instead of software activated. With this PWM module, multiple ADC triggers can be generated in hardware per PWM cycle without the requirement of another timer module. Figure 26-8 shows how this is accomplished. When specifying a complementary mode of operation, only two edge comparators are required to generate the output PWM signals for a given submodule. This means the other comparators are

free to perform other functions. In this example, the software does not need to quickly respond after the first conversion to set up other conversions that must occur in the same PWM cycle.

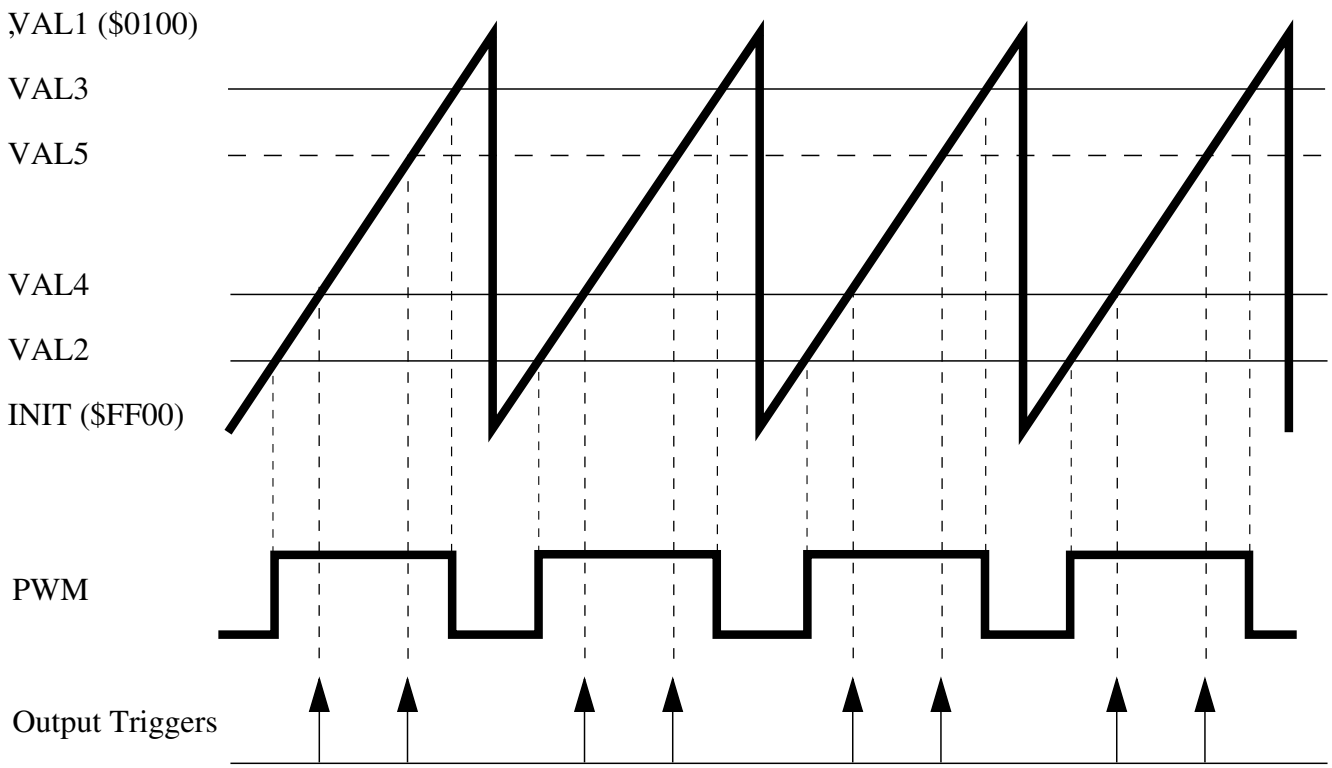


Figure 26-8. Multiple output trigger generation in hardware

Because each submodule has its own timer, it is possible for each submodule to run at a different frequency. One of the options possible with this PWM module is to have one or more submodules running at a lower frequency, but still synchronized to the timer in submodule0. Figure 26-9 shows how this feature can be used to schedule ADC triggers over multiple PWM cycles. You can use the lower-frequency submodule to control the sampling frequency of the software control algorithm where multiple ADC triggers can now be scheduled over the entire sampling period. In Figure 26-9, *all* submodule comparators are shown being used for ADC trigger generation.

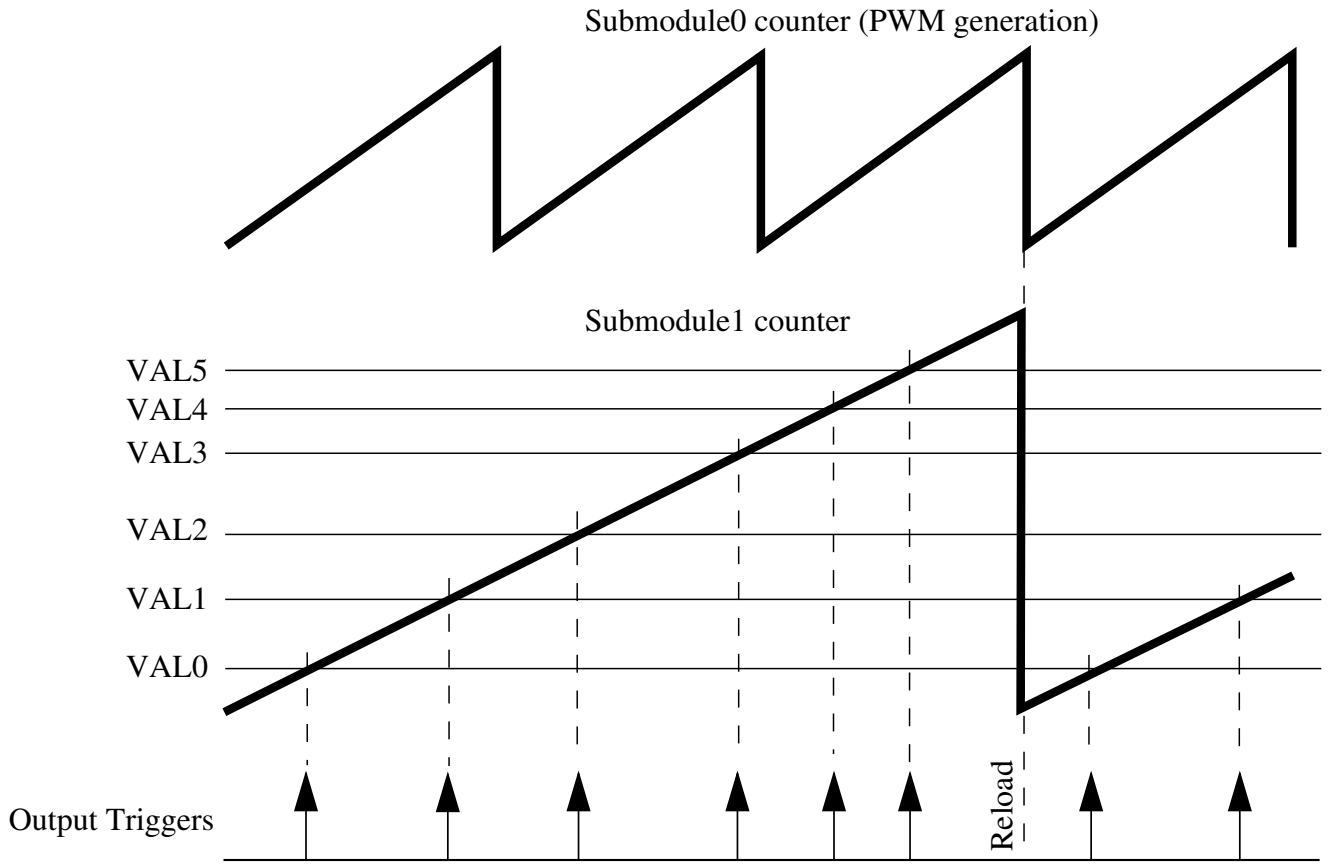


Figure 26-9. Multiple output triggers over several PWM cycles

26.4.2.6 Enhanced capture capabilities (E-Capture)

When a PWM pin is not used for PWM generation, it can be used to perform input captures. For PWM generation, both edges of the PWM signals are specified via separate compare register values. When programmed for input capture, both of these registers work on the same pin to capture multiple edges, toggling from one to the other in either a free running or one-shot fashion. Programming the desired edge of each capture circuit, period, and pulse width of an input signal can easily be measured without the requirement to re-arm the circuit. In addition, each edge of the input signal can clock an 8-bit counter where the counter output is compared to a user specified value (EDGCMP). When the counter output equals EDGCMP, the value of the submodule timer is captured and the counter is automatically reset. This feature counts a specified number of edge events and then performs a capture and interrupt. The following figure illustrates some of the functionality of the E-Capture circuit.

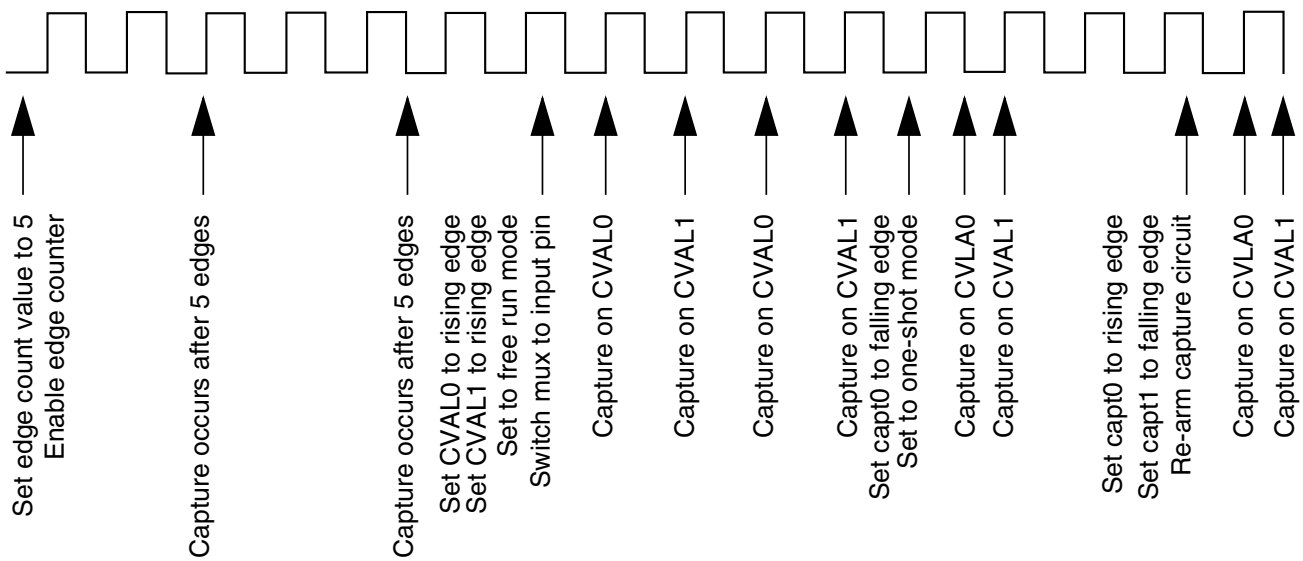


Figure 26-10. Capture capabilities of the E-Capture circuit

When a submodule is used for PWM generation, its timer counts up to the modulus value used to specify the PWM frequency and then is re-initialized. Therefore, using this timer for input captures on one of the other pins (for example, PWM_X) has limited utility since it does not count through all of the numbers and the timer reset represents a discontinuity in the 16-bit number range. However, when measuring a signal that is synchronous to the PWM frequency, the timer modulus range is suited for the application. As shown in Figure 26-11, the output of a PWM power stage is connected to the PWM_X pin that is configured for free running input captures. Specifically, the CVAL0 capture circuitry is programmed for rising edges and the CVAL1 capture circuitry is set for falling edges. This results in new load pulse width data acquired every PWM cycle. To calculate the pulse width, subtract the CVAL0 register value from the CVAL1 register value. This measurement is beneficial when performing deadtime distortion correction on a half bridge circuit driving an inductive load. Also, these values can be directly compared to the VALx registers responsible for generating the PWM outputs to obtain a measurement of system propagation delays. For details, refer to the separate discussion of deadtime distortion correction.

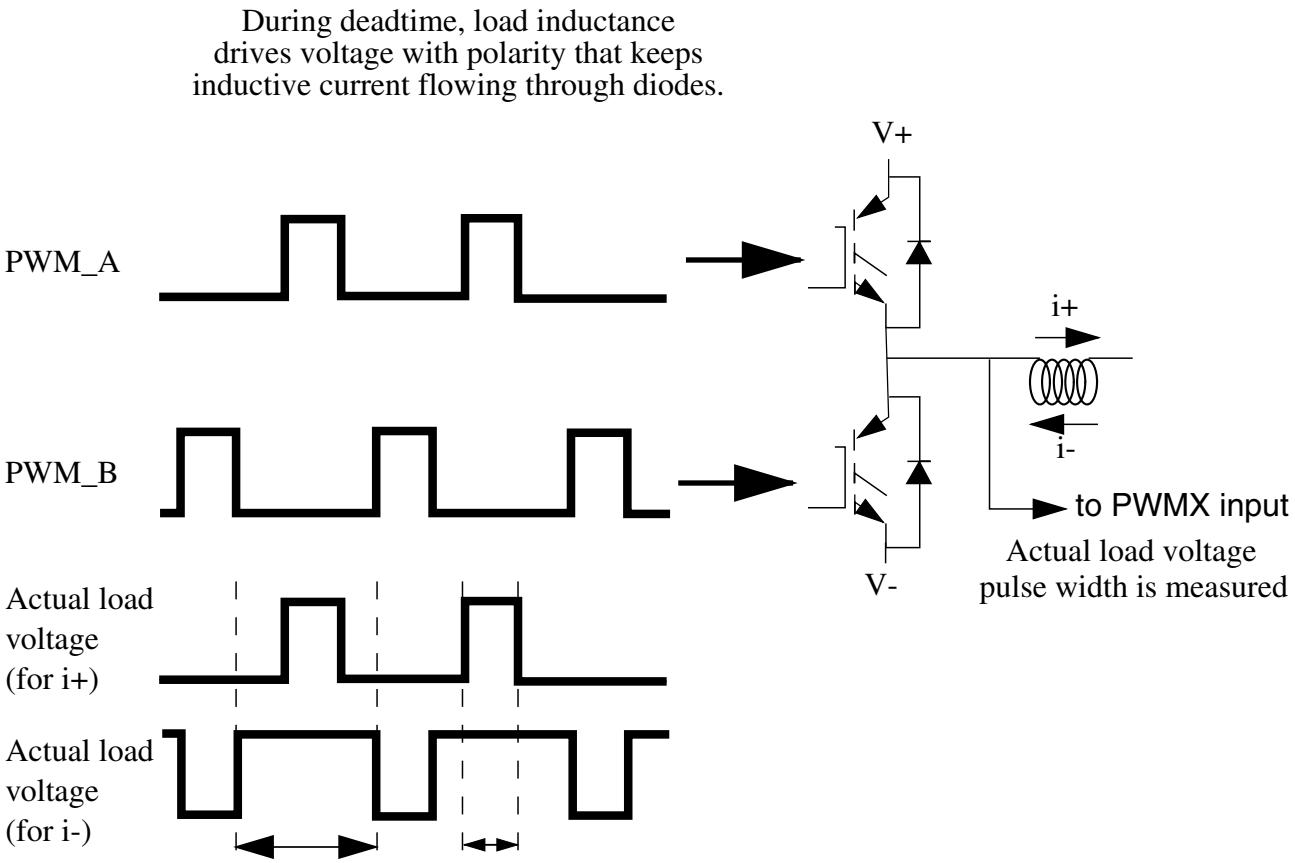


Figure 26-11. Output pulse width measurement possible with the E-capture circuit

26.4.2.7 Synchronous switching of multiple outputs

Before the PWM signals are routed to the output pins, they are processed by a hardware block that permits all submodule outputs to be switched synchronously. This feature is useful in commutated motor applications where the next commutation state can be laid in ahead of time and then immediately switched to the outputs when the appropriate condition or time is reached. All the changes occur immediately after the trigger event occurs eliminating any interrupt latency and also the changes occurs synchronously on all submodule outputs.

The synchronous output switching is accomplished via a signal called FORCE_OUT. This signal originates from the local FORCE bit within the submodule, from submodule0, or from external to the PWM module, and in most cases, is supplied from an external timer channel configured for output compare. In a typical application, software sets up the desired states of the output pins in preparation for the next FORCE_OUT event. This selection lays dormant until the FORCE_OUT signal transitions and then all outputs are

switched simultaneously. The signal switching is performed upstream from the deadtime generator so that any abrupt changes that occur do not violate deadtime on the power stage when in complementary mode.

Figure 26-12 shows an application that can benefit from this feature. On a brushless DC motor in many cases, it is required to spin the motor without need of hall-effect sensor feedback. Instead, the back EMF of the motor phases is monitored and this information is used to schedule the next commutation event. The top waveforms of Figure 26-12 represent these back EMF signals. Timer compare events (represented by the long vertical lines in the diagram) are scheduled based on the zero crossings of the back-EMF waveforms. The PWM module is configured via software ahead of time with the next state of the PWM pins in anticipation of the compare event. When it happens, the output compare of the timer drives the FORCE_OUT signal which immediately changes the state of the PWM pins to the next commutation state with no software latency.

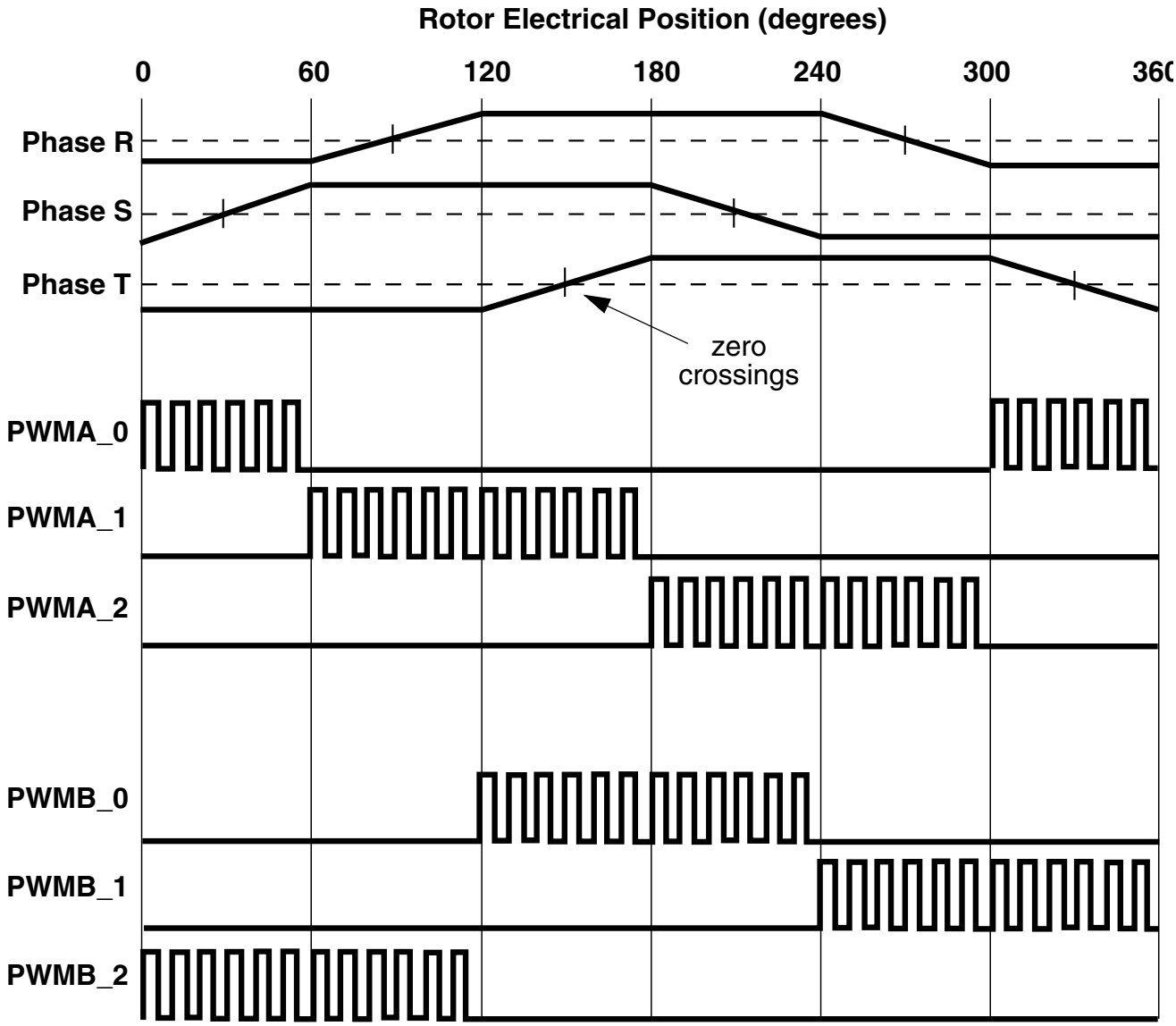


Figure 26-12. Sensorless BLDC commutation using the force out function

26.4.3 Operation

This section describes the implementation of various sections of the PWM in detail.

The following figure is a high-level block diagram of output PWM generation.

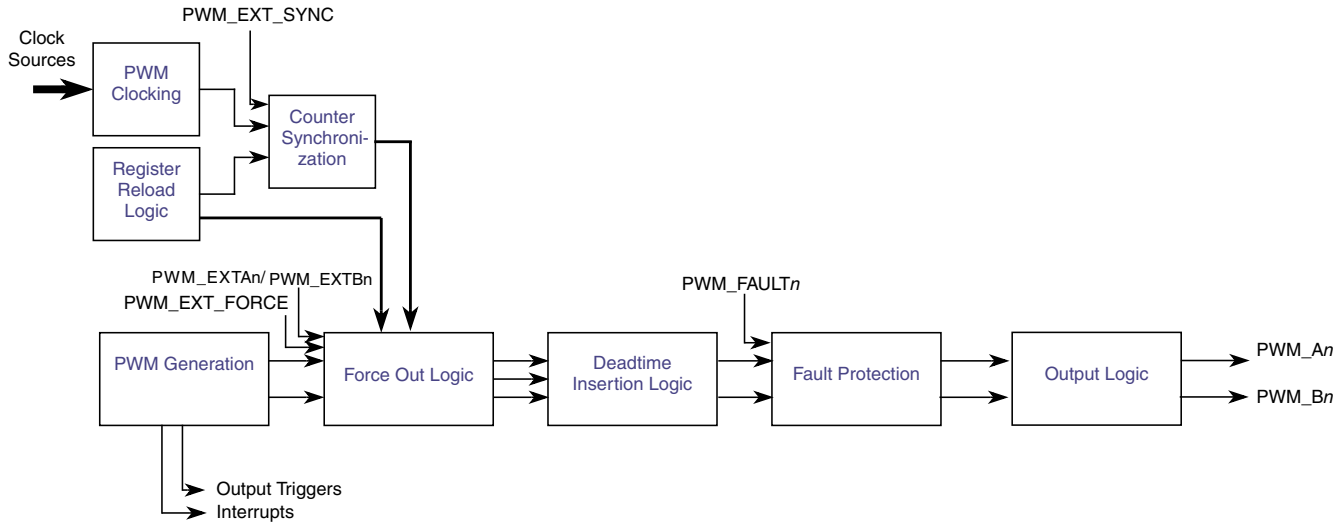


Figure 26-13. High-level output PWM generation block diagram

26.4.3.1 Register reload logic

The register reload logic is used to determine when the outer set of registers for all double buffered register pairs will be transferred to the inner set of registers. The register reload event can be scheduled to occur every "n" PWM cycles using CTRL[LDFQ] and CTRL[FULL], which is defined by VAL1 register. A half cycle reload option is also supported (CTRL[HALF]) where the reload can take place in the middle of a PWM cycle. The half cycle point is defined by the VAL0 register and does not have to be exactly in the middle of the PWM cycle.

As shown in Figure 26-14 the reload signal from submodule0 can be broadcast as the Master Reload signal allowing the reload logic from submodule0 to control the reload of registers in other submodules.

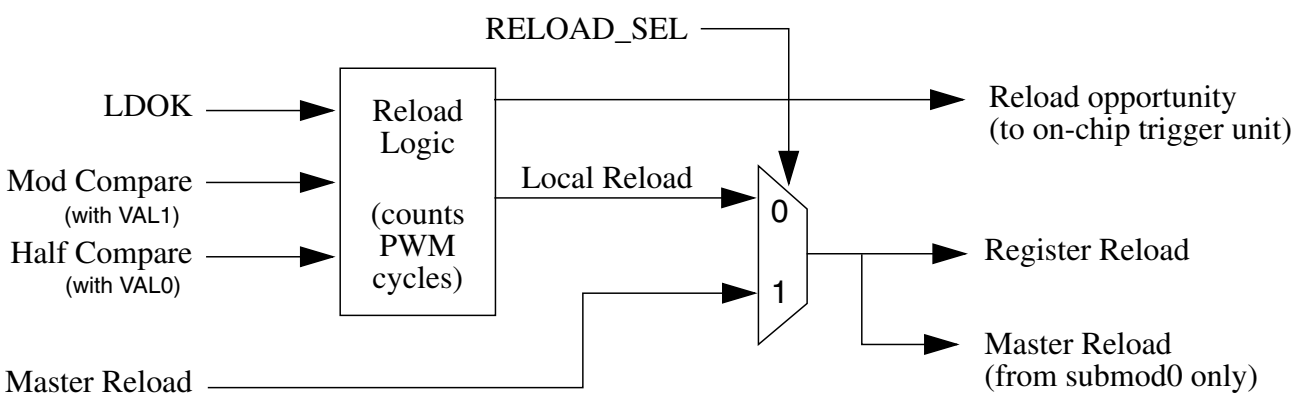


Figure 26-14. Register reload logic of 1 submodule

26.4.3.2 Counter synchronization

As shown in Figure 26-15, the 16-bit counter counts up until its output equals VAL1 which is used to specify the counter modulus value. The resulting compare causes a rising edge to occur on the Local Synchronization signal which is one of four possible sources used to cause the 16-bit counter to be initialized with INIT. If Local Synchronization is selected as the counter initialization signal, VAL1 within the submodule effectively controls the timer period (and then the PWM frequency generated by that submodule), and everything operates or functions at a local level.

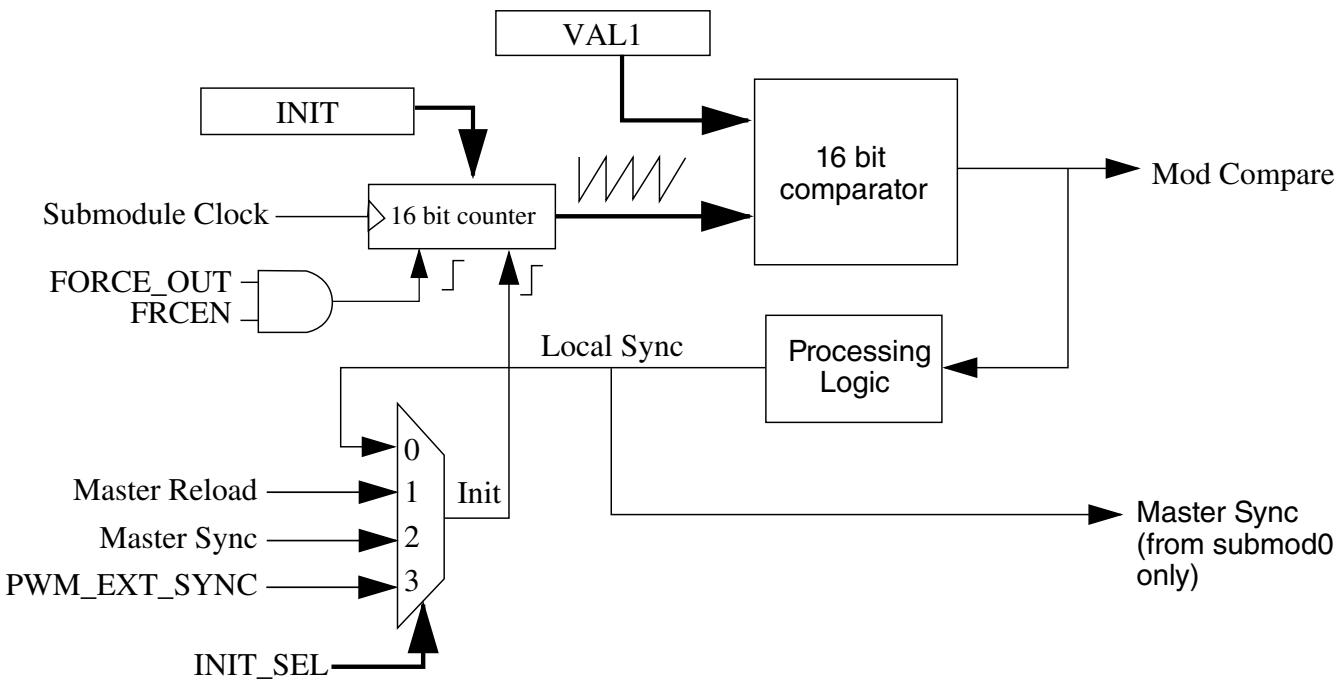


Figure 26-15. Submodule timer synchronization for 1 submodule

The Master Synchronization signal originates as the Local Sync from submodule0. If configured to do so, the timer period of any submodule can be locked to the period of the timer in submodule0.

The PWM_EXT_SYNC signal originates on chip or off chip depending on the system architecture. This signal may be selected as the source for counter initialization so that an external source can control the period of all submodules.

If the Master Reload signal is selected as the source for counter initialization, then the period of the counter will be locked to the register reload frequency of submodule0. Since the reload frequency is commensurate to the sampling frequency of the software control algorithm, the submodule counter period is equal the sampling period. As a result, this

timer can be used to generate output compares or output triggers over the entire sampling period which may consist of several PWM cycles. The Master Reload signal can only originate from submodule0.

The counter can optionally initialize upon the assertion of the FORCE_OUT signal assuming that CTRL2[FRCEN] is set. As shown in Figure 26-15, this constitutes a second initialization input into the counter, which causes the counter to initialize regardless of which signal is selected as the counter initialization signal. A forced initialization causes a register reload if MCTRL[LDOK] is set.

The counter can be initialized by FORCE_OUT signal only when MCTRL[RUN] = 0 or CTLR2[CLK_SEL] = 2 which chooses auxiliary clock from SM0.

The FORCE_OUT signal is provided mainly for commutated applications. When PWM signals are commutated on an inverter controlling a brushless DC motor, it is necessary to restart the PWM cycle at the beginning of the commutation interval. This action effectively resynchronizes the PWM waveform to the commutation timing. Otherwise, the average voltage applied to a motor winding integrated over the entire commutation interval will be a function of the timing between the asynchronous commutation event for the PWM cycle. The effect is more critical at higher motor speeds where each commutation interval may consist of only a few PWM cycles. If the counter is not initialized at the start of each commutation interval, the result is an oscillation caused by the beating between the PWM frequency and the commutation frequency.

26.4.3.3 PWM generation

Figure 26-16 illustrates how PWM generation is accomplished in each submodule. In each case, two comparators and associated VALx registers are utilized for each PWM output signal. One comparator and VALx register are used to control the turn-on edge, while a second comparator and VALx register control the turn-off edge.

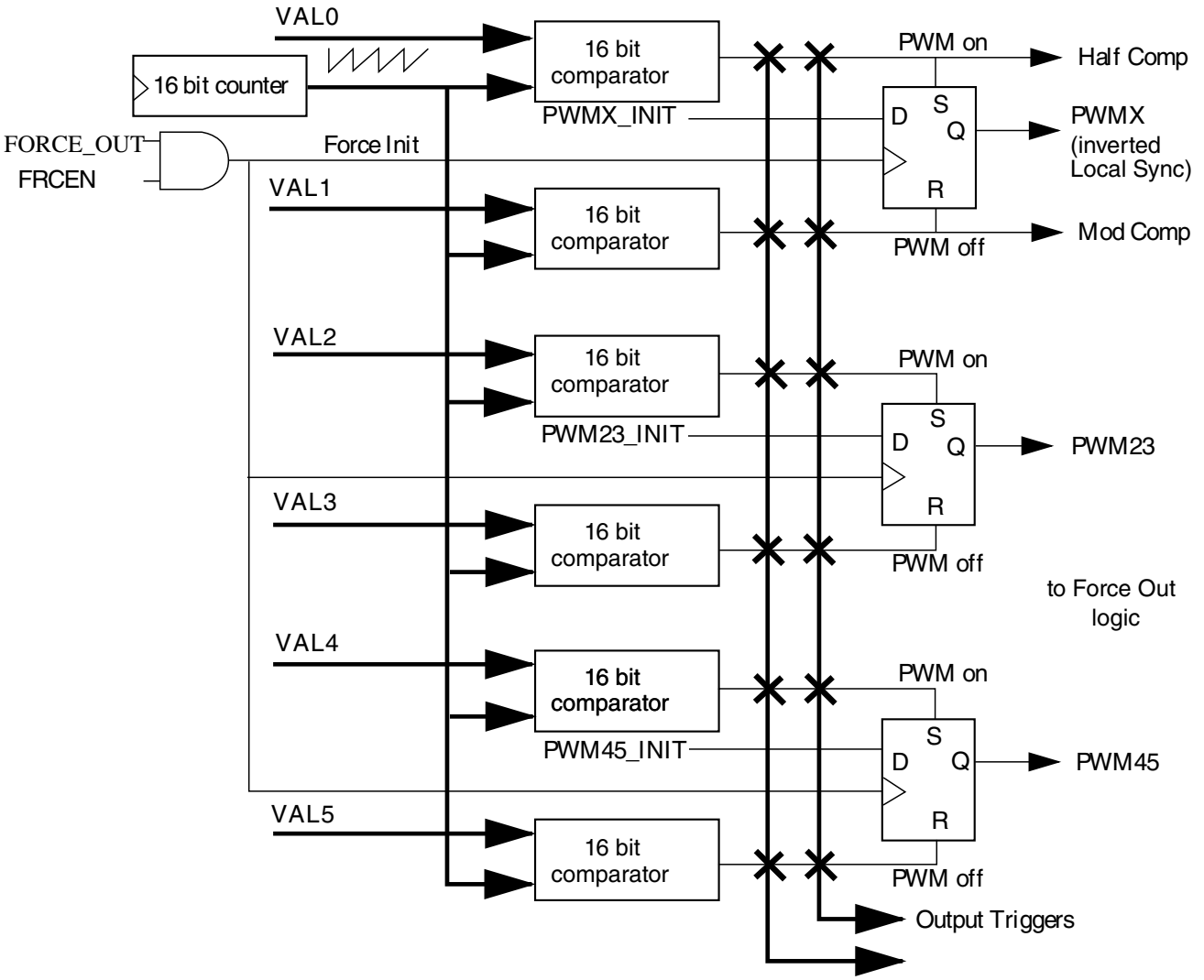


Figure 26-16. PWM Generation Hardware of 1 submodule

The generation of the Local Synchronization signal is performed exactly the same way as the other PWM signals in the submodule. While comparator 0 causes a falling edge of the Local Synchronization signal, comparator 1 generates a rising edge. Comparator 1 is also hardwired to the reload logic to generate the full cycle reload indicator.

If VAL1 is controlling the modulus of the counter and VAL0 is half of the VAL1 register minus the INIT value, then the half cycle reload pulse occurs exactly half way through the timer count period and the Local Synchronization will have a 50% duty cycle. On the other hand, if the VAL1 and VAL0 registers are not required for register reloading or counter initialization, they can be used to modulate the duty cycle of the Local Synchronization signal, effectively turning it into an auxiliary PWM signal (PWM_X) assuming that the PWM_X pin is not being used for another function such as input capture or deadtime distortion correction. Including the Local Synchronization signal, each submodule is capable of generating three PWM signals where software has complete control over each edge of each of the signals.

If the comparators and edge value registers are not required for PWM generation, they can be used for other functions such as output compares, generating output triggers, or generating interrupts at timed intervals.

The 16-bit comparators shown in Figure 26-16 are "equal to" comparators. In addition, if both the set and reset of the flip-flop are asserted, then the flop output goes to 0.

26.4.3.4 Output compare capabilities

By using the VALx registers in conjunction with the submodule timer and 16-bit comparators, buffered output compare functionality can be achieved with no additional hardware required. Specifically, the following output compare functions are possible:

- An output compare sets the output high
- An output compare sets the output low
- An output compare generates an interrupt
- An output compare generates an output trigger

In PWM generation, an output compare is initiated by programming a VALx register for a timer compare, which in turn causes the output of the D flip-flop to either set or reset. For example, if an output compare is desired on the PWM_A signal that sets it high, VAL2 would be programmed with the counter value where the output compare should take place. However, to prevent the D flip-flop from being reset again after the compare has occurred, the VAL3 register must be programmed to a value out of the modulus range of the counter. Therefore, a comparison that would result in resetting the D flip-flop output would never occur. Conversely, if an output compare is desired on the PWM_A signal that sets it low, the VAL3 register is programmed with the appropriate count value and the VAL2 register is programmed with a value out of the counter modulus range. Regardless of whether a high compare or low compare is programmed, an interrupt, or output trigger can be generated when the compare event occurs.

26.4.3.5 Force out logic

For each submodule, the software can select between eight signal sources for the FORCE_OUT signal depending on the chip architecture:

1. Local CTRL2[FORCE]
2. Master Force signal from submodule0
3. Local Reload signal
4. Master Reload signal from submodule0

5. Local Synchronization signal
6. Master Synchronization signal from submodule0
7. EXT_SYNC signal from on or off chip
8. EXT_FORCE signal from on or off chip

The local signals are used to change the signals on the output pins of the submodule without regard for synchronization with other submodules. However, if it is required that all signals on all submodule outputs change at the same time, the Master, EXT_SYNC, or EXT_FORCE signals must be selected.

Figure 26-17 illustrates the Force logic. The SEL23 and SEL45 fields each choose from one of four signals that can be supplied to the submodule outputs: the PWM signal, the inverted PWM signal, a binary level specified by software via the OUT23 and OUT45 bits, or the PWM_EXTA or PWM_EXTB alternate external control signals. The selection can be determined ahead of time, and when a FORCE_OUT event occurs, these values are presented to the signal selection mux that immediately switches the requested signal to the output of the mux for further processing downstream.

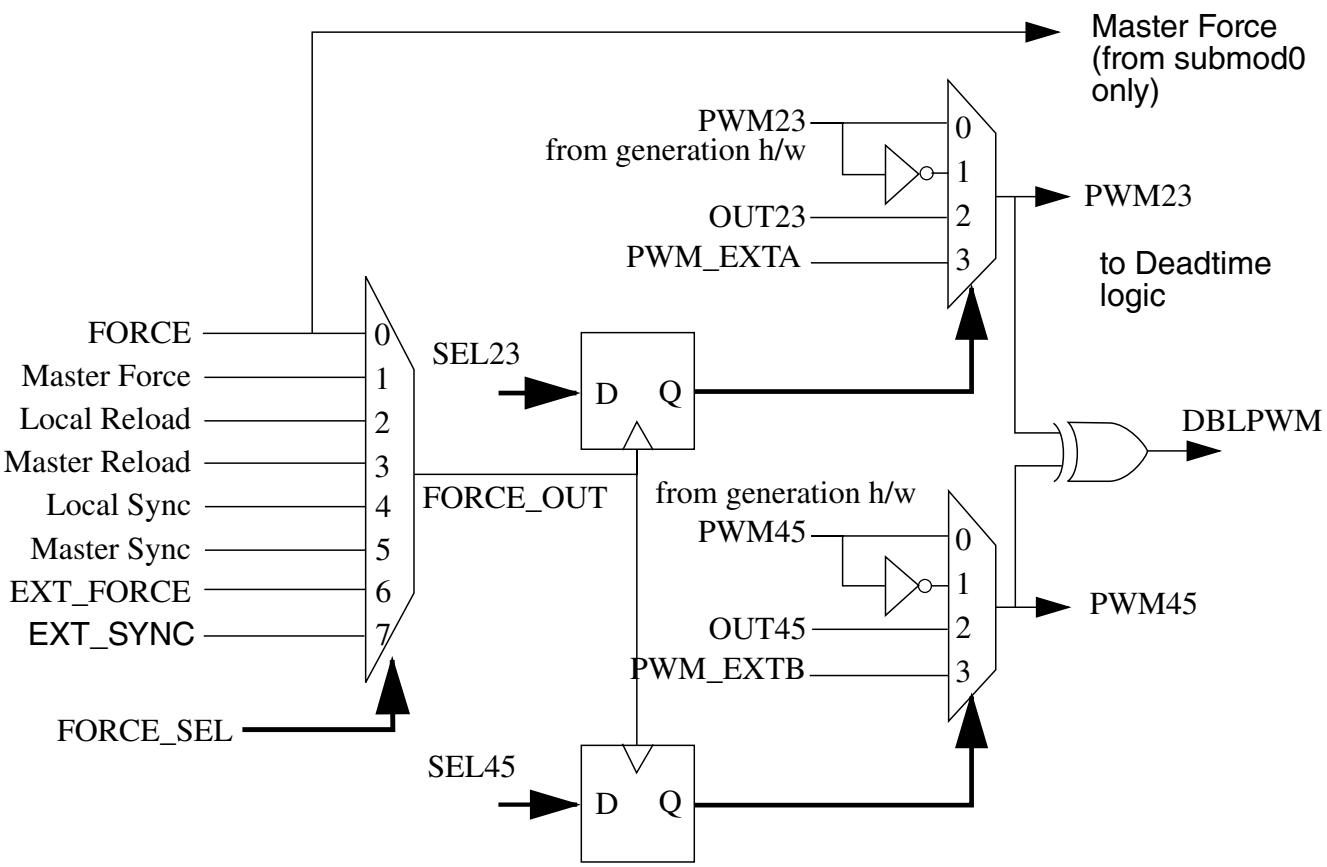


Figure 26-17. Force out logic for 1 submodule

The local CTRL2[FORCE] signal of submodule0 can be broadcast as the Master Force signal to other submodules. This feature allows the CTRL2[FORCE] of submodule0 to synchronously update all of the submodule outputs at the same time. The EXT_FORCE signal originates from outside the PWM module from a source such as a timer or digital comparators in the Analog-to-Digital Converter.

26.4.3.6 Independent or complementary channel operation

Writing a logic one to CTRL2[INDEP] configures the pair of PWM outputs as two independent PWM channels. Each PWM output is controlled by its own VALx pair operating independently of the other output.

Writing a logic zero to CTRL2[INDEP] configures the PWM output as a pair of complementary channels. The PWM pins are paired as shown in Figure 26-18 in complementary channel operation. The signal that is connected to the output pin (PWM23 or PWM45) is determined by MCTRL[IPOL].

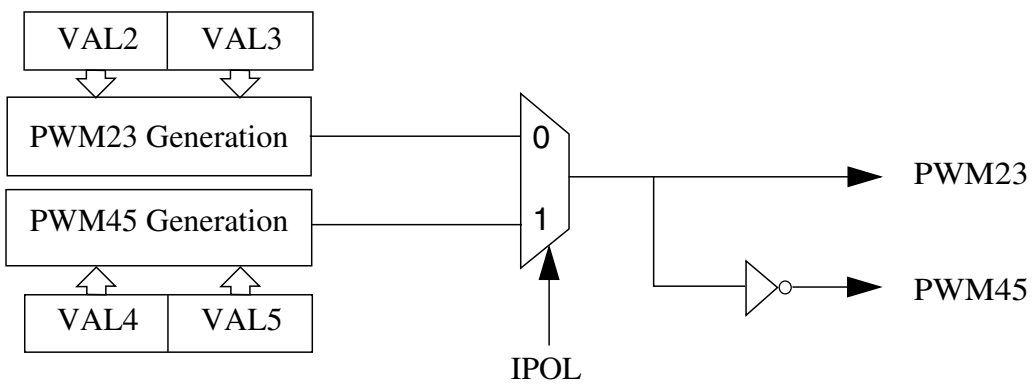


Figure 26-18. Complementary channel pair for 1 submodule

The complementary channel operation is for driving top and bottom transistors in a motor drive circuit, as shown in Figure 26-19.

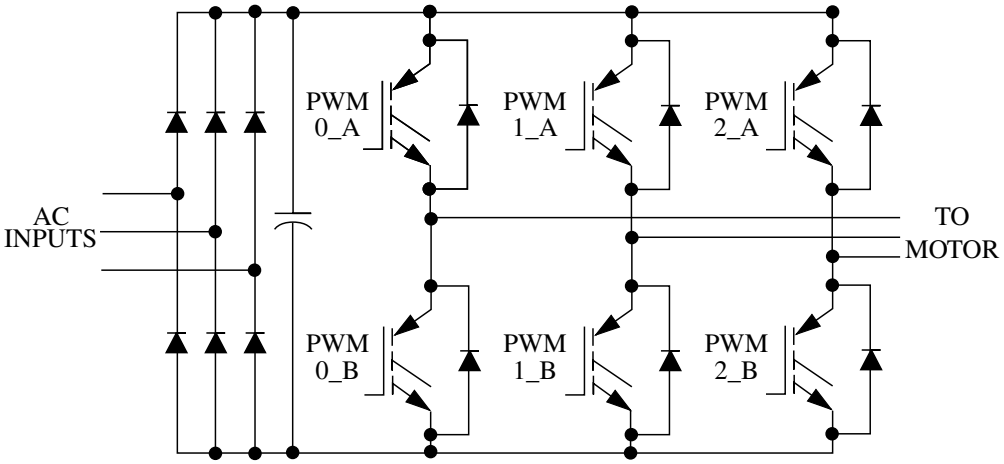


Figure 26-19. Typical 3 phase AC motor drive

The complementary operation allows the use of the deadtime insertion feature.

26.4.3.7 Deadtime insertion logic

The following figure shows the deadtime insertion logic of each submodule which is used to create non-overlapping complementary signals when not in independent mode.

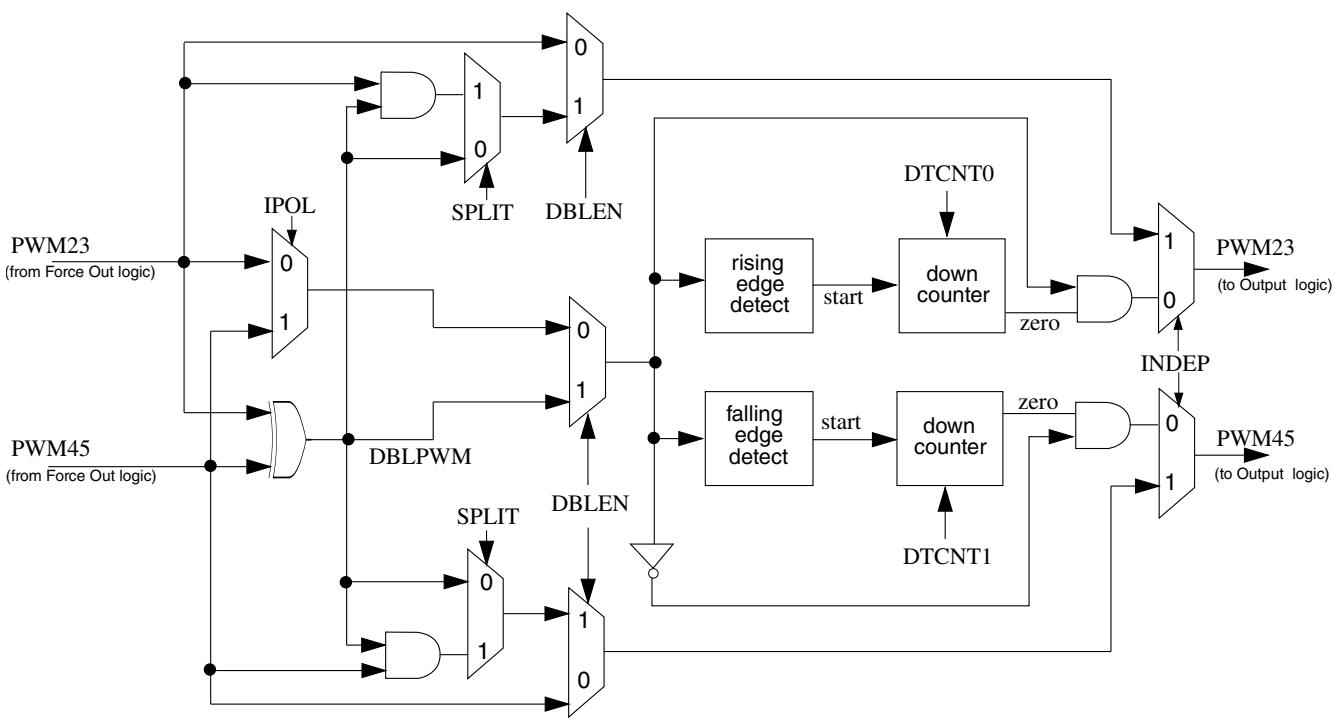


Figure 26-20. Deadtime Insertion Logic for 1 submodule

While in the complementary mode, a PWM pair can be used to drive top/bottom transistors, as shown in the figure. When the top PWM channel is active, the bottom PWM channel is inactive, and vice versa.

Note

To avoid short-circuiting the DC bus and endangering the transistor, there must be no overlap of conducting intervals between the top and bottom transistors. But the transistor's characteristics make its switching-off time longer than switching-on time. To avoid the conducting overlap of top and bottom transistors, deadtime needs to be inserted in the switching period, as shown in Figure 26-21.

The deadtime generators automatically insert software-selectable activation delays into the pair of PWM outputs. The deadtime registers (DTCNT0 and DTCNT1) specify the number of IPBus clock cycles to use for deadtime delay. Every time the deadtime generator inputs change state, deadtime is inserted. Deadtime forces both PWM outputs in the pair to the inactive state.

When deadtime is inserted in complementary PWM signals connected to an inverter driving an inductive load, the PWM waveform on the inverter output will have a different duty cycle than what appears on the output pins of the PWM module. This results in a distortion in the voltage applied to the load. A method of correcting this, adding to or subtracting from the PWM value used, is discussed next.

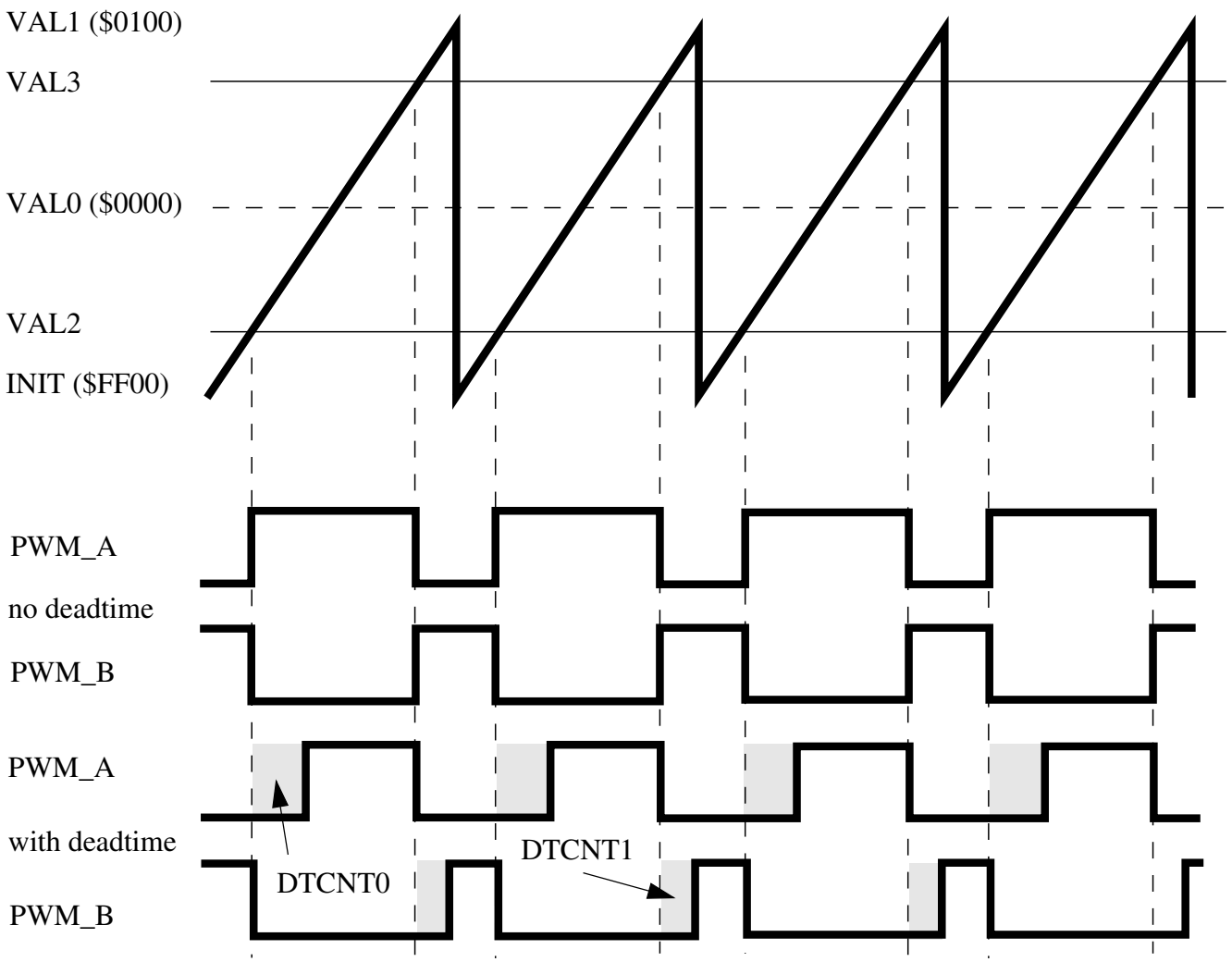


Figure 26-21. Deadtime insertion

26.4.3.7.1 Top/Bottom correction

In complementary mode, either the top or the bottom transistor controls the output voltage. However, deadtime has to be inserted to avoid overlap of conducting interval between the top and bottom transistor. Both transistors in complementary mode are off during deadtime, allowing the output voltage to be determined by the current status of load and introducing distortion in the output voltage, as shown in Figure 26-22. On AC induction motors running open-loop, the distortion typically manifests itself as poor low-speed performance, such as torque ripple and rough operation.

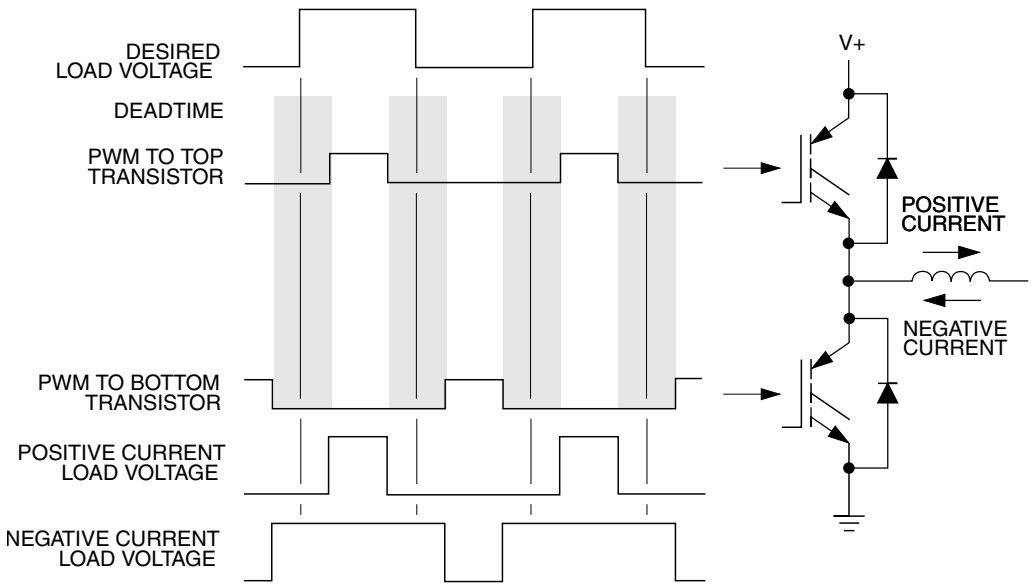


Figure 26-22. Deadtime distortion

During deadtime, load inductance distorts output voltage by keeping current flowing through the diodes. This deadtime current flow creates a load voltage that varies with the current direction. With a positive current flow, the load voltage during deadtime is equal to the bottom supply, putting the top transistor in control. With a negative current flow, the load voltage during deadtime is equal to the top supply putting the bottom transistor in control.

Remembering that the original PWM pulse widths were shortened by deadtime insertion, the averaged sinusoidal output is less than the desired value. However, when deadtime is inserted, it distorts in the motor current waveform inverter outputs. This distortion is aggravated by different turn-on and turn-off delays of each of the transistors. By giving the PWM module information on which transistor is controlling at a given time this distortion can be corrected.

For a typical circuit in complementary channel operation, only one of the transistors is effective in controlling the output voltage at any given time. This depends on the direction of the motor inverter current for that pair, as shown in Figure 26-22. To correct distortion one of two different factors must be added to the desired PWM value, depending on whether the top or bottom transistor is controlling the output voltage. Therefore, the software is responsible for calculating both compensated PWM values prior to placing them in the VALx registers. Either the VAL2/VAL3 or the VAL4/VAL5 register pair controls the pulse width at any given time. For a given PWM pair, whether the VAL2/VAL3 or VAL4/VAL5 pair is active depends on either:

- The state of the current status pin, PWMX, for that driver
- The state of the odd/even correction bit, MCTRL[IPOL], for that driver

To correct deadtime distortion, the software can decrease or increase the value in the appropriate VALx register.

- In edge-aligned operation, decreasing or increasing the PWM value by a correction value equal to the deadtime typically compensates for deadtime distortion.
- In center-aligned operation, decreasing or increasing the PWM value by a correction value equal to one-half the deadtime typically compensates for deadtime distortion.

26.4.3.7.2 Manual correction

To detect the current status, the voltage on each PWMX pin is sampled twice in a PWM period, at the end of each deadtime. The value is stored in CTRL[DT]. CTRL[DT] is a timing marker indicating when to toggle between PWM value registers. The software can then set MCTRL[IPOL] to switch between VAL2/VAL3 and VAL4/VAL5 register pairs according to CTRL[DT] values.

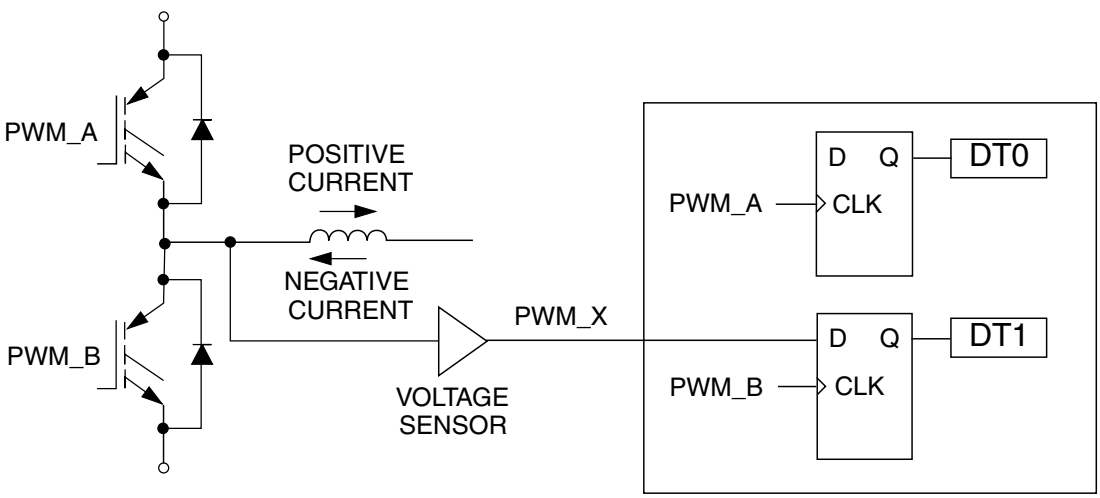


Figure 26-23. Current-status sense scheme for deadtime correction

Both D flip-flops latch low, CTRL[DT] = 00, during deadtime periods if the current is large and flowing out of the complementary circuit. See the preceding figure. Both D flip-flops latch the high, CTRL[DT] = 11, during deadtime periods if the current is also large and flowing into the complementary circuit.

However, under low-current, the output voltage of the complementary circuit during deadtime is between the high and low levels. The current cannot free-wheel through the opposition anti-body diode, regardless of polarity, giving additional distortion when the current crosses zero. Sampled results are CTRL[DT] = b10. Thus, the best time to change one PWM value register to another is just before the current zero crossing.

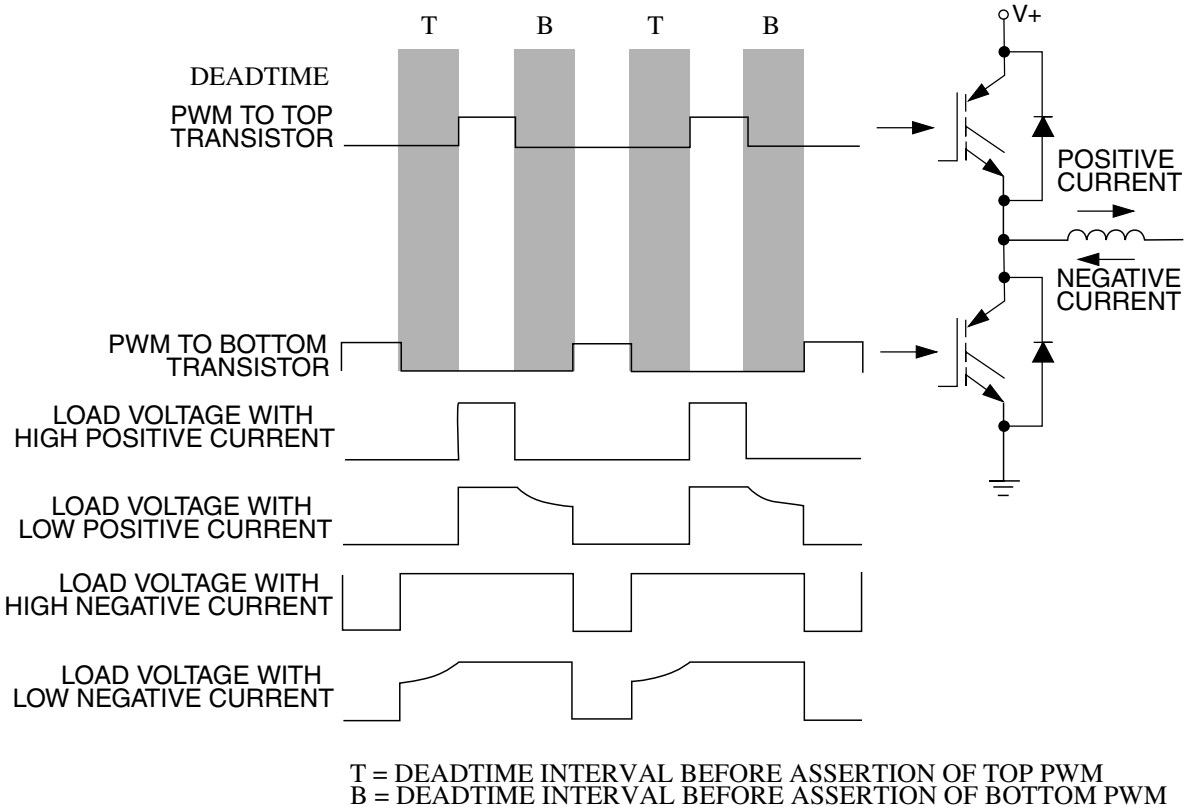


Figure 26-24. Output voltage waveforms

26.4.3.8 Fractional delay logic

For applications where more resolution than a single IPBus clock period is needed, the fractional delay logic can be used to achieve fine resolution on the rising and falling edges of the PWM_A and PWM_B outputs and fine resolution for the PWM period. Enable the use of the fractional delay logic by setting FRCTRL[FRACx_EN]. The FRACVALx registers act as a fractional clock cycle addition to the turn-on and turn-off count specified by the VAL2, VAL3, VAL4, or VAL5 registers. The FRACVAL1 register acts as a fractional increase in the PWM period as defined by VAL1. If FRACVAL1 is programmed to a non-zero value, then the largest value for the VAL1 register is 0xFFFE for unsigned usage or 0x7FFE for signed usage. This limit is required to avoid counter rollovers when accumulating the fractional additional period.

Both the fractional enables (1, 23, 45) and the fractional values (1-5) are double buffered and reloaded at the same time as the value registers.

Each PWM cycle, the value compare point is increased by 1 when there is overflow on the double buffered value of the fractional register plus the 5-bit accumulated fractional value. The accumulated fractional value starts at zero, so it is impossible to overflow the first PWM cycle.

At the end of each PWM cycle, if the corresponding fractional enable is set then the accumulated fractional value increments by the fractional value (double buffered value). The accumulated fractional value is 5-bits, so in the case of overflow only the remainder is kept. If the corresponding fractional enable is clear, then the accumulated fractional value is reset. This is the only way to reset the accumulated fractional value.

The accumulated fractional values are not accessible to software.

As an example, assume the following conditions:

- INIT = 0x0000
- VAL1 = 0x000F
- VAL2 = 0x0000
- VAL3 = 0x0007
- FRACVAL3 = 0x00

This would cause the PWM output to have a 50% duty cycle. It would be high from a count value of 0x0 to the count value of 0x7, at which point it would go low until the counter reaches the maximum value of 0xF. If FRACVAL3 was changed to a value of 0x17, then the time the PWM output is active would be 8 cycles + (23/32) cycle = 8.719 cycles, and for a high duty cycle of (8.719 cycles / 16 cycles) x 100% = 54.49%.

The results of the fractional delay logic depend on whether or not the PWM sub-module has an analog NanoEdge placer block available. With fractional delay enabled, PWM works in high-resolution mode: the value of FRACVALx is a fractional addition to the value of VALx for each PWM period.

26.4.3.8.1 Fractional delay logic with NanoEdge placement block

Using the NanoEdge placer block requires that the IP Bus clock to the PWM be set at a defined frequency. The NanoEdge placer is powered up by setting FRCTRL[FRAC_PU]. Enable fine edge control on the various PWM edges by setting FRCTRL[FRACx_EN]. The fractional values in the FRACVALx registers allow placing the PWM edge or PWM period to a granularity of 1/32 of the IP Bus clock period. For example, if you desire the rising edge of the PWM_A output to occur at a count of 12.25, then program VAL2 with 0x000C and FRACVAL2 with 0x4000. Using FRACVAL1 will adjust the PWM period with the same granularity of 1/32 of a clock period.

If the FRCTRL[FRAC_PU] bits in all of the submodules are clear, then the NanoEdge placer is powered down, and alternate clock frequencies can be used without the NanoEdge placement feature.

26.4.3.9 Output logic

The following figure shows the output logic of each submodule including how each PWM output has individual fault disabling, polarity control, and output enable. This allows for maximum flexibility when interfacing with the external circuitry.

The PWM23 and PWM45 signals which are output from the deadtime logic (as shown in Figure 26-25) are positive true signals. In other words, a high level on these signals should result in the corresponding transistor in the PWM inverter being turned ON. The voltage level required at the PWM output pin to turn the transistor ON or OFF is a function of the logic between the pin and the transistor. Therefore, it is imperative that the user program OCTRL[POLA] and OCTRL[POLB] before enabling the output pins. A fault condition can result in the PWM output being tristated, forced to a logic 1, or forced to a logic 0 depending on the values programmed into the OCTRL[PWMxFS] fields.

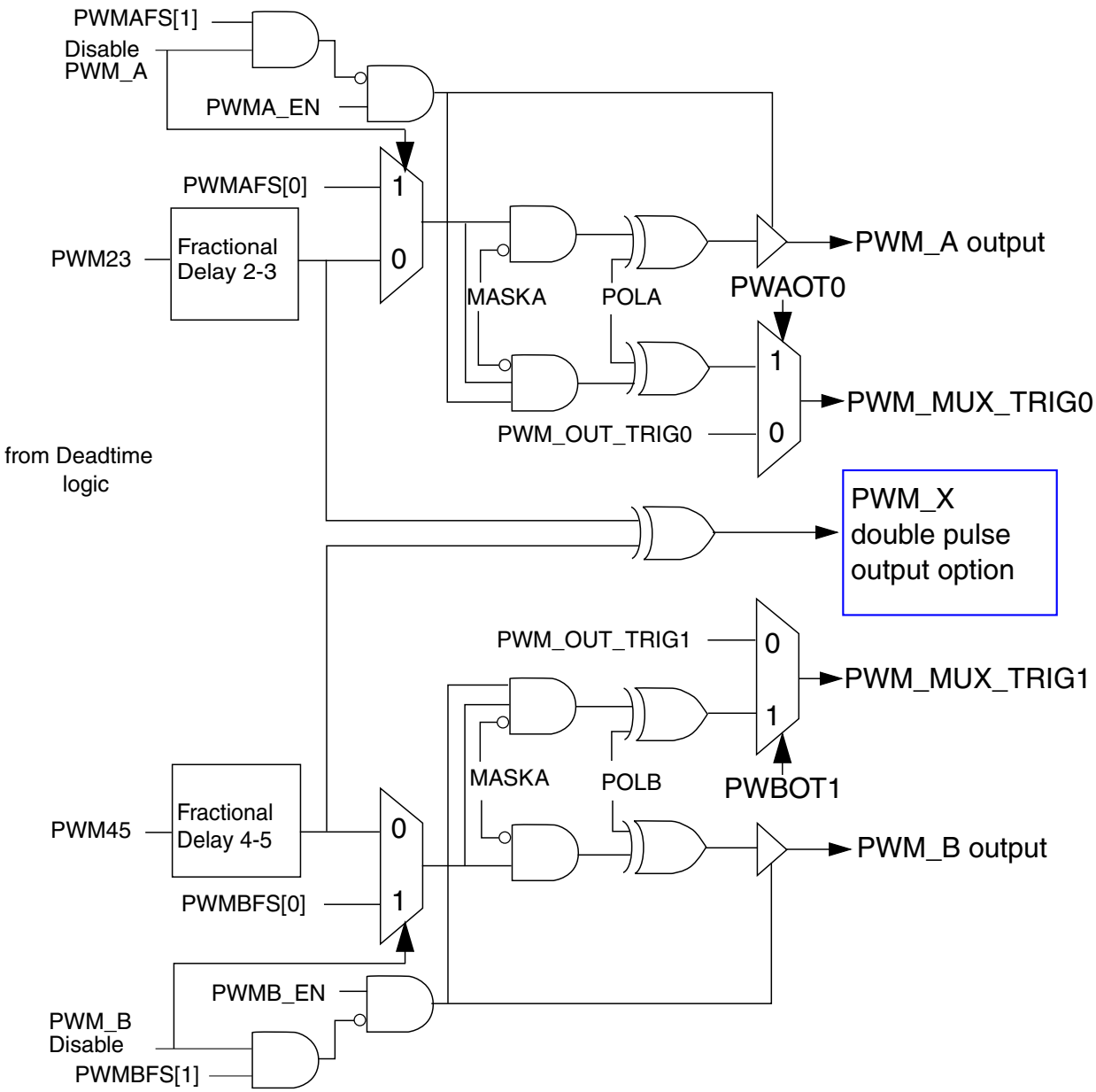


Figure 26-25. Output logic for 1 submodule

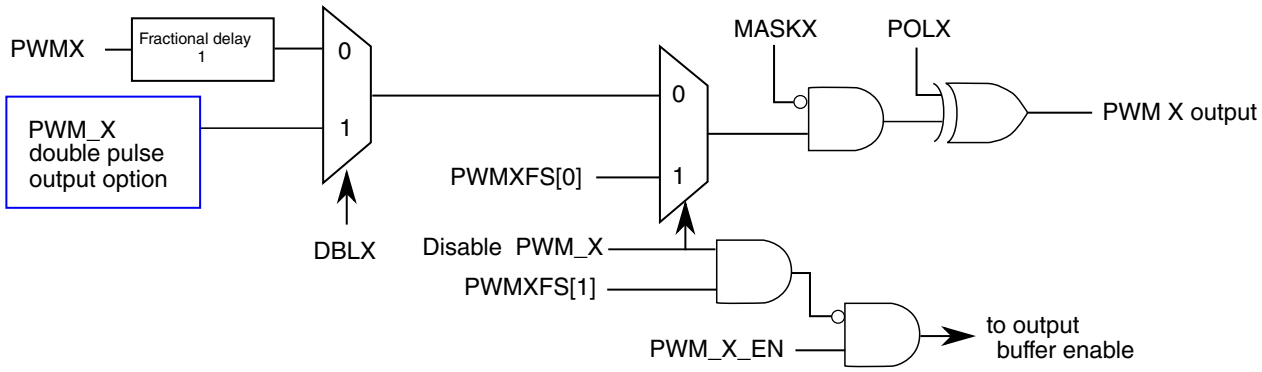


Figure 26-26. Output logic continued

26.4.3.10 E-Capture

The Enhanced Capture (E-Capture) logic is designed to measure both edges of an input signal. As a result, when a submodule pin is configured for input capture, the CVALx registers associated with that pin are used to record the edge values.

The following figure shows block diagram of the E-Capture circuit. Upon entering the pin input, the signal is split into two paths. One goes straight to a mux input where software can select to pass the signal directly to the capture logic for processing. The other path connects the signal to an 8-bit counter which counts both the rising and falling edges of the signal. The output of this counter is compared to an 8-bit value that is specified by the user (EDGCMPx) and when the two values are equal, the comparator generates a pulse that resets the counter. This pulse is also supplied to the mux input where software can select it to be processed by the capture logic. This feature permits the E-Capture circuit to count up to 256 edge events before initiating a capture event. This feature is useful for dividing down high frequency signals for capture processing so that capture interrupts do not overwhelm the CPU. Also, this feature can be used to generate an interrupt after "n" events have been counted.

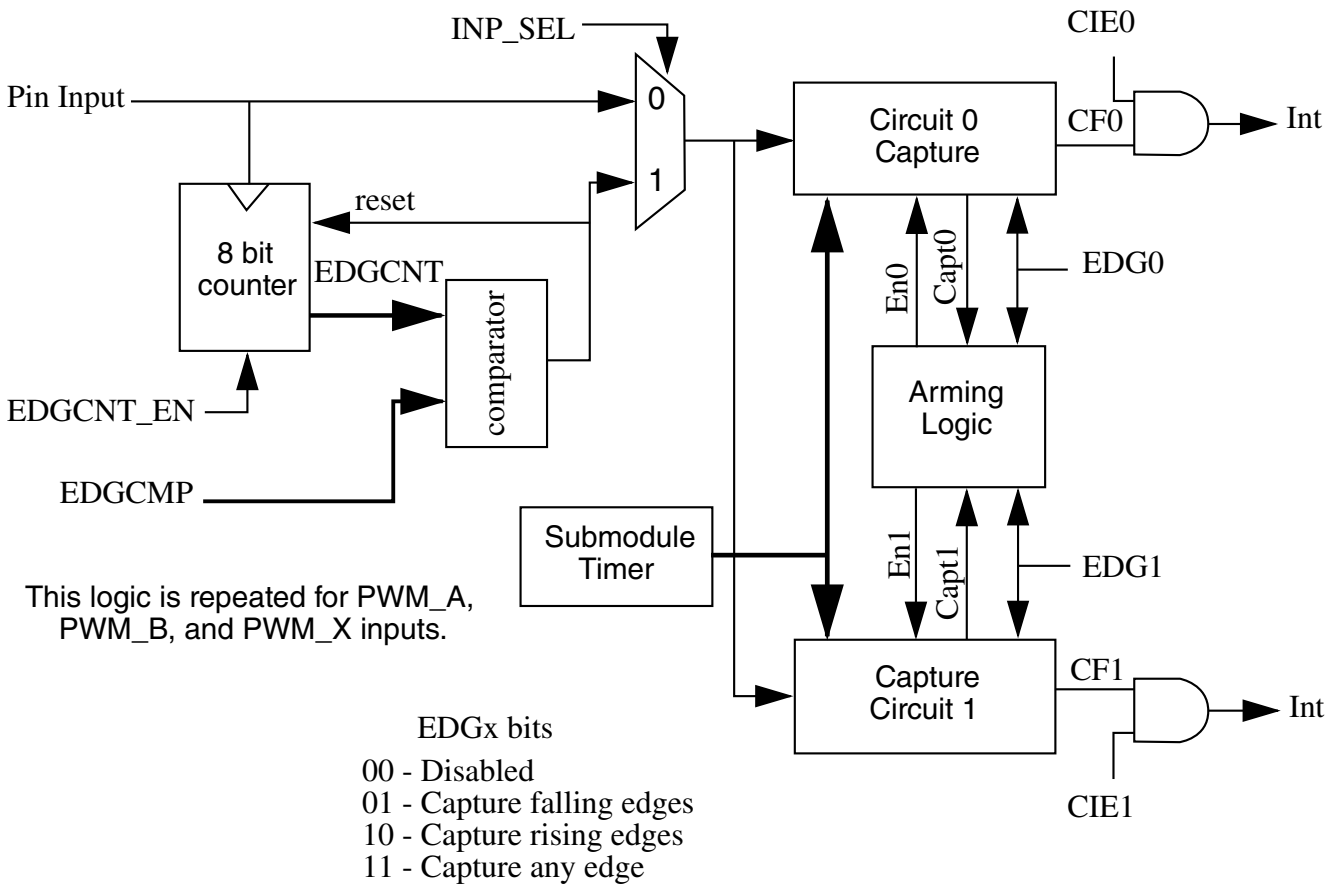


Figure 26-27. Enhanced capture (E-Capture) logic

Based on the mode selection, the mux selects either the pin input or the compare output from the count/compare circuit to be processed by the capture logic. The selected signal is routed to two separate capture circuits which work in tandem to capture sequential edges of the signal. The type of edge to be captured by each circuit is determined by CAPTCTRLx[EDGx1] and CAPTCTRLx[EDGx0], whose functionality is listed in the above figure. Also, controlling the operation of the capture circuits is the arming logic which allows captures to be performed in a free running (continuous) or one shot mode. In free running mode, the capture sequences will be performed indefinitely. If both capture circuits are enabled, they will work together in a ping-pong style where a capture event from one circuit leads to the arming of the other and vice versa. In one shot mode, only one capture sequence will be performed. If both capture circuits are enabled, capture circuit 0 is first armed and when a capture event occurs, capture circuit 1 is armed. Once the second capture occurs, further captures are disabled until another capture sequence is initiated. Both capture circuits are also capable of generating an interrupt to the CPU.

26.4.3.11 Fault protection

Fault protection can control any combination of PWM output pins. Faults are generated by a logic one on any of the FAULTx pins. This polarity can be changed via FCTRL[FLVL]. Each FAULTx pin can be mapped arbitrarily to any of the PWM outputs. When fault protection hardware disables PWM outputs, the PWM generator continues to run, only the output pins are forced to logic 0, logic 1, or high impedance depending on the values of OCTRL[PWMxFS].

The fault decoder disables PWM pins selected by the fault logic and the disable mapping (DISMAPn) registers. The following figure shows an example of the fault disable logic. Each bank of bits in DISMAPn control the mapping for a single PWM pin. See the following table.

The fault protection is enabled even when the PWM module is not enabled. Therefore, a fault is latched in and must be cleared to prevent an interrupt when the PWM is enabled.

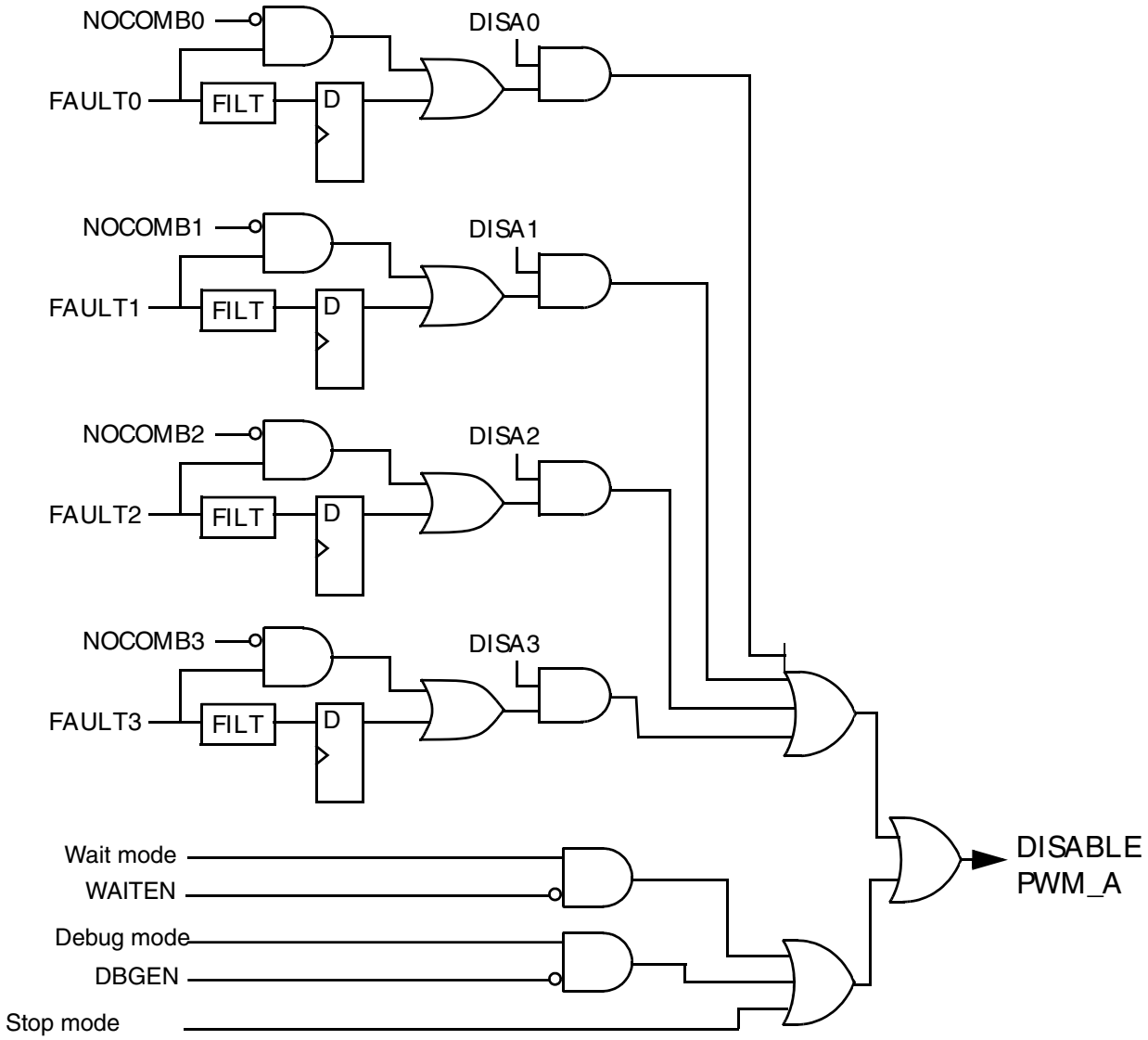


Figure 26-28. Fault decoder for PWM_A

Table 26-1. Fault Mapping

PWM Pin for 1 submodule	Controlling Register Bits
PWM_A	DISMAP0[DIS0A] and DISMAP1[DIS1A]
PWM_B	DISMAP0[DIS0B] and DISMAP1[DIS1B]
PWM_X	DISMAP0[DIS0X] and DISMAP1[DIS1X]

26.4.3.11.1 Fault pin filter

Each fault pin has a programmable filter that can be bypassed. The sampling period of the filter can be adjusted with FFILT[FILT_PER]. The number of consecutive samples that must agree before an input transition is recognized can be adjusted using FFILT[FILT_CNT]. Setting FFILT[FILT_PER] to all 0 disables the input filter for a given FAULTx pin.

Upon detecting a logic 0 on the filtered FAULTx pin (or a logic 1 if FCTRL[FLVLx] is set), the corresponding FSTS[FFPINx] and FSTS[FFLAGx] bits are set. FSTS[FFPINx] remains set as long as the filtered FAULTx pin is zero. Clear FSTS[FFLAGx] by writing a logic 1 to FSTS[FFLAGx].

If the FIEx, FAULTx pin interrupt enable bit is set, FSTS[FFLAGx] generates a CPU interrupt request. The interrupt request latch remains set until:

- Software clears FSTS[FFLAGx] by writing a logic one to the bit
- Software clears the FIEx bit by writing a logic zero to it
- A reset occurs

Even with the filter enabled, there is a combinational path from the FAULTx inputs to the PWM pins, which in turn bypasses the filter when FCTRL20[NOCOMBx] = 0. This logic is also capable of holding a fault condition in the event of loss of clock to the PWM module.

26.4.3.11.2 Automatic fault clearing

Setting an automatic clearing mode bit, FCTRL[FAUTOx] configures faults from the FAULTx pin for automatic clearing.

When FCTRL[FAUTOx] is set, disabled PWM pins are enabled when the FAULTx pin returns to logic one and a new PWM full or half cycle begins. See the following figure. If FSTS[FFULLx] is set and the fault condition on FAULTx disappears, then the disabled PWM pins are enabled at the start of next full cycle. If FSTS[FHALFx] is set, then the disabled PWM pins are enabled at the start of a half cycle. Clearing FSTS[FFLAGx] does not affect disabled PWM pins when FCTRL[FAUTOx] is set.

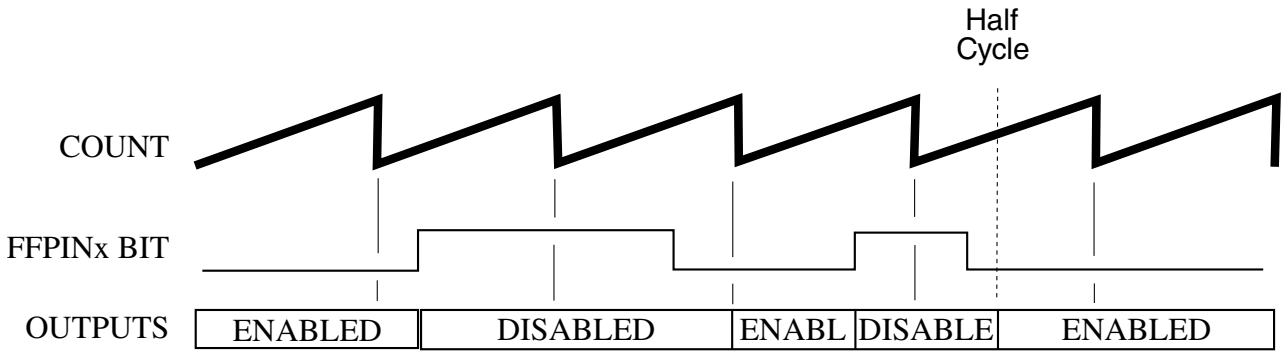


Figure 26-29. Automatic fault clearing

26.4.3.11.3 Manual fault clearing

Clearing the automatic clearing mode bit, FCTRL[FAUTOx], configures faults from the FAULTx pin for manual clearing:

- If the fault safety mode bits FCTRL[FSAFEx] are clear, then PWM pins disabled by the FAULTx pins are enabled when:
  - Software clears the corresponding FSTS[FFLAGx] flag
  - The pins are enabled when the next PWM full or half cycle begins regardless of the logic level detected by the filter at the FAULTx pin, as shown in Figure 26-30. If FSTS[FFULLx] is set, then the disabled PWM pins are enabled at the start of a full cycle. If FSTS[FHALFx] is set, then the disabled PWM pins are enabled at the start of a half cycle.
- If the fault safety mode bits FCTRL[FSAFEx] are set, then PWM pins disabled by the FAULTx pins are enabled when:
  - Software clears the corresponding FSTS[FFLAGx] flag
  - The filter detects a logic zero on the FAULTx pin at the start of the next PWM full or half cycle boundary, as shown in Figure 26-31. If FSTS[FFULLx] is set, then the disabled PWM pins are enabled at the start of a full cycle. If FSTS[FHALFx] is set, then the disabled PWM pins are enabled at the start of a half cycle.

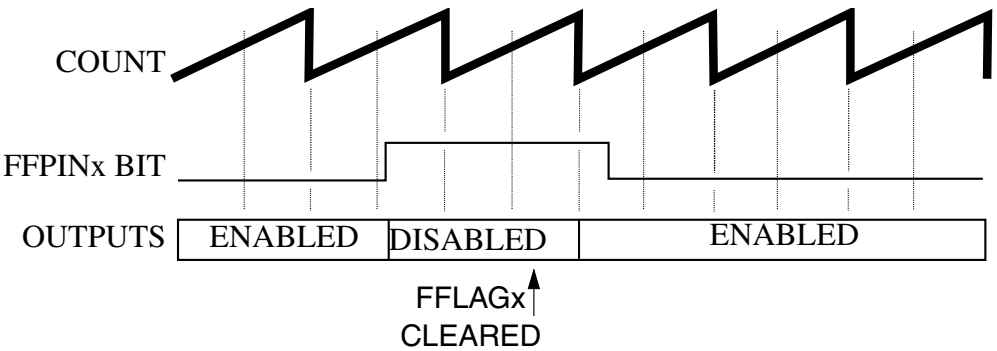


Figure 26-30. Manual fault clearing (FCTRL[FSAFEx] = 0, FSTS[FFULLx] = 1)

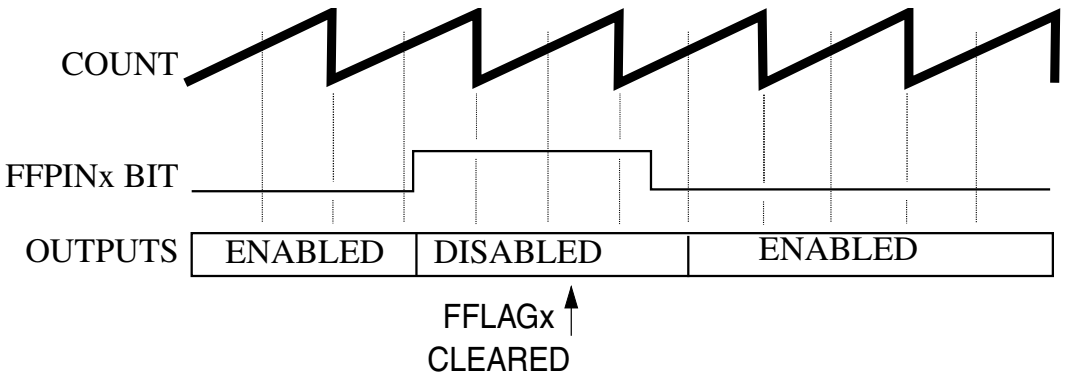


Figure 26-31. Manual fault clearing (FCTRL[FSAFEx] = 1, FSTS[FHALFx] = 1)

Note

Fault protection also applies during software output control when the SEL23 and SEL45 fields are set to select OUT23 and OUT45 bits or PWM_EXTA and PWM_EXTB. Fault clearing still occurs at half or full PWM cycle boundaries while the PWM generator is engaged, MCTRL[RUN] equals one. But the OUTx bits can control the PWM pins while the PWM generator is off, MCTRL[RUN] equals zero. Thus, fault clearing occurs at IPBus cycles while the PWM generator is off and at the start of PWM cycles when the generator is engaged.

26.4.3.11.4 Fault testing

FTST[FTEST] is used to simulate a fault condition on each of the fault inputs within that fault channel.

26.4.3.12 PWM generator loading

26.4.3.12.1 Load enable

MCTRL[LDOK] enables loading of the following PWM generator parameters.

- The prescaler divisor: from CTRL[PRSC]
- The PWM period and pulse width: from the INIT, FRACVALx, and VALx registers

MCTRL[LDOK] allows the software to finish calculating all of these PWM parameters so they can be synchronously updated. The CTRL[PRSC], INIT, and VALx registers are loaded by software into a set of outer buffers. When MCTRL[LDOK] is set, these values are transferred to an inner set of registers at the beginning of the next PWM reload cycle to be used by the PWM generator when CTRL[LDMOD] is cleared. These values can be transferred to the inner set of registers immediately upon setting MCTRL[LDOK] if CTRL[LDMOD] is set. After loading, MCTRL[LDOK] is automatically cleared.

26.4.3.12.2 Load frequency

CTRL[LDFQ] selects an integral loading frequency of one to 16 PWM reload opportunities. CTRL[LDFQ] takes effect at every PWM reload opportunity, regardless of the state of MCTRL[LDOK]. CTRL[HALF] and CTRL[FULL] control reload timing. If CTRL[FULL] is set, a reload opportunity occurs at the end of every PWM cycle when the count equals VAL1. If CTRL[HALF] is set, a reload opportunity occurs at the half cycle when the count equals VAL0. If both CTRL[HALF] and CTRL[FULL] are set, a reload opportunity occurs twice per PWM cycle when the count equals VAL1 and when it equals VAL0.

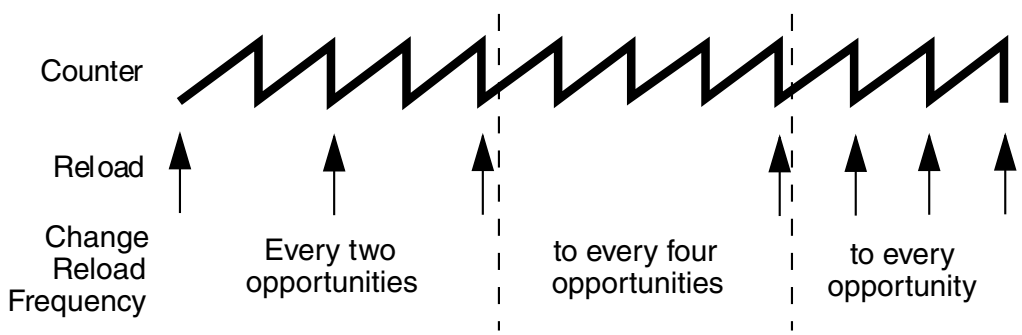


Figure 26-32. Full cycle reload frequency change

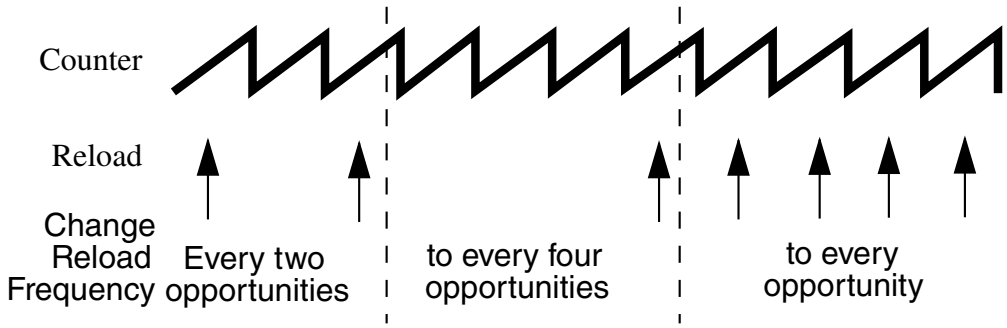


Figure 26-33. Half cycle reload frequency change

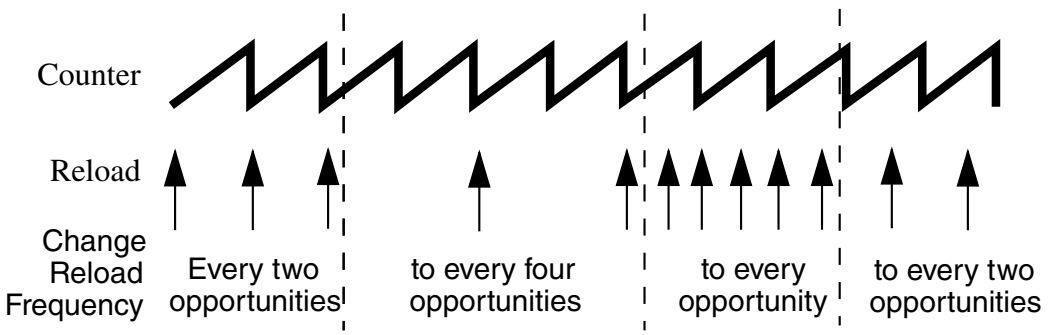


Figure 26-34. Full and half cycle reload frequency change

26.4.3.12.3 Reload flag

At every reload opportunity the PWM Reload Flag (STS[RF]) is set. Setting STS[RF] happens even if an actual reload is prevented by MCTRL[LDOK]. If the PWM reload interrupt enable bit INTEN[RIE] is set, the STS[RF] flag generates CPU interrupt requests allowing software to calculate new PWM parameters in real time. When INTEN[RIE] is not set, reloads still occur at the selected reload rate without generating CPU interrupt requests.

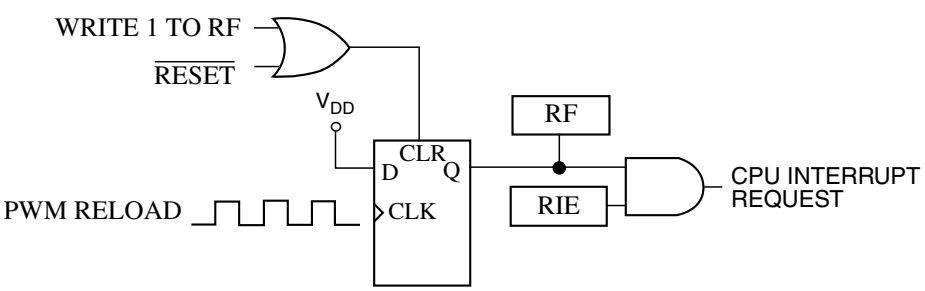


Figure 26-35. PWMF reload interrupt request

26.4.3.12.4 Reload errors

When one of the INIT, VALx, FRACVALx, or CTRL[PRSC] registers is updated (written by software), the STS[RUF] flag is set to indicate the data in the set of double buffered registers is not coherent. STS[RUF] is cleared by a successful reload which consists of the reload signal while MCTRL[LDOK] is set. If STS[RUF] is set and MCTRL[LDOK] is clear when the reload signal occurs, a reload error takes place and STS[REF] is set. If STS[RUF] is clear when a reload signal asserts, then the data is coherent and no error is flagged.

26.4.3.12.5 Initialization

Initialize all registers and then set MCTRL[LDOK] before setting MCTRL[RUN].

Note

If MCTRL[LDOK] is not set, setting MCTRL[RUN] also sets the STS[RF] flag. To prevent a CPU interrupt request, clear INTEN[RIE] before setting MCTRL[RUN].

The PWM generator uses the last values loaded if MCTRL[RUN] is cleared and then set while MCTRL[LDOK] equals zero.

When MCTRL[RUN] is cleared:

- The STS[RF] flag and pending CPU interrupt requests are not cleared
- All fault circuitry remains active
- Software/external output control remains active
- Deadtime insertion continues during software/external output control

26.4.4 Power modes

Be careful when using this module in Stop, Wait, and Debug operating modes.

CAUTION

Some applications require regular software updates for proper operation. Failure to provide regular software updates could result in destroying the hardware setup.

To accommodate this situation, PWM outputs are placed in their inactive states in Stop mode, and they can optionally be placed in inactive states in Wait and Debug modes. PWM outputs are reactivated (assuming they were active beforehand) when these modes are exited.

Table 26-2. Modes when PWM operation is restricted

Mode	Description
Stop	PWM outputs are inactive.
Wait	PWM outputs are driven or inactive as a function of CTRL2[WAITEN].
Debug	PWM outputs are driven or inactive as a function of CTRL2[DBGEN].

26.4.5 Clocking

Figure 26-36 shows the logic used to generate the main counter clock. Each submodule can select between three clock signals: the IPBus clock, EXT_CLK, and AUX_CLK. The EXT_CLK goes to all of the submodules. The AUX_CLK signal is broadcast from submodule0 and can be selected as the clock source by other submodules so that the 8-bit prescaler and MCTRL[RUN] from submodule0 can control all of the submodules.

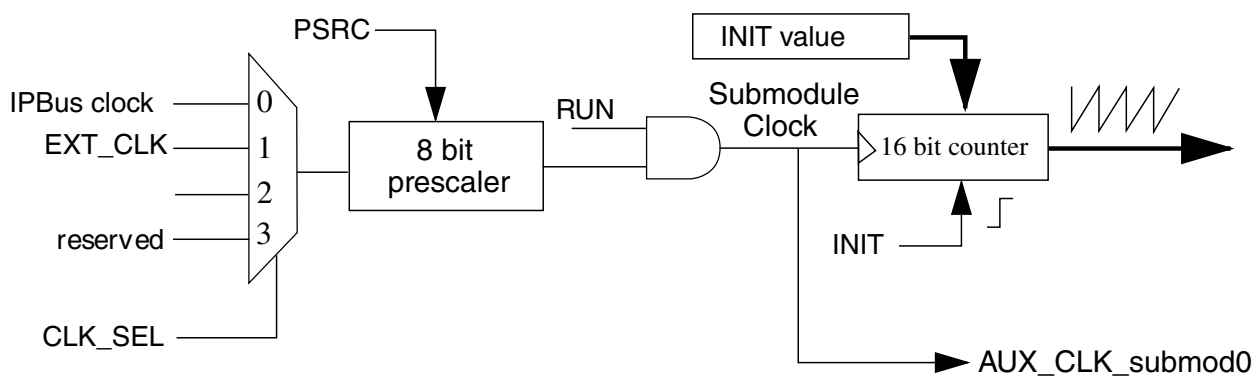


Figure 26-36. Clocking Block Diagram for PWM Submodule 0

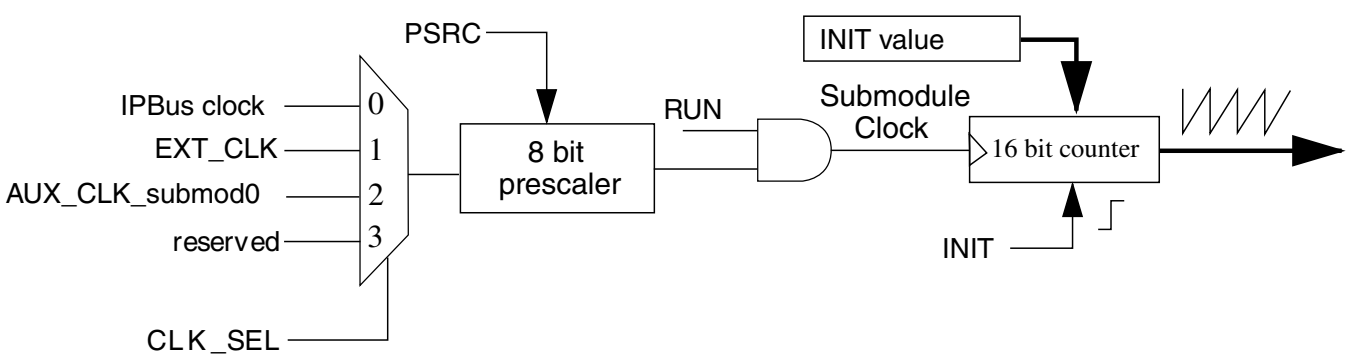


Figure 26-37. Clocking Block Diagram for PWM Submodule 1,2,3

To permit lower PWM frequencies, the prescaler produces the PWM clock frequency by dividing the IPBus clock frequency by 1-128. The prescaler bits, CTRL[PRSC], select the prescaler divisor. This prescaler is buffered and will not be used by the PWM generator until MCTRL[LDOK] is set and a new PWM reload cycle begins or CTRL[LDMOD] is set.

26.4.6 Resets

All PWM registers are reset to their default values upon any system reset.

The reset forces all registers to their reset states and tri-states the PWM outputs.

26.4.7 Interrupts

Each of the submodules within the eFlexPWM module can generate an interrupt from several sources. The fault logic can also generate interrupts. The interrupt service routine (ISR) must check the related interrupt enables and interrupt flags to determine the actual cause of the interrupt.

Table 26-3. Interrupt Summary

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
PWM_CMP0	SM0STS[CMPF]	SM0INTEN[CMPIE]	Submodule 0 compare interrupt	Compare event has occurred
PWM_CAP0	SM0STS[CFA1], SM0STS[CFA0], SM0STS[CFB1], SM0STS[CFB0], SM0STS[CFX1], SM0STS[CFX0]	SM0INTEN[CFA1IE], SM0INTEN[CFA0IE], SM0INTEN[CFB1IE], SM0INTEN[CFB0IE], SM0INTEN[CFX1IE], SM0INTEN[CFX0IE]	Submodule 0 input capture interrupt	Input capture event has occurred
PWM_RELOAD0	SM0STS[RF]	SM0INTEN[RIE]	Submodule 0 reload interrupt	Reload event has occurred
PWM_CMP1	SM1STS[CMPF]	SM1INTEN[CMPIE]	Submodule 1 compare interrupt	Compare event has occurred
PWM_CAP1	SM1STS[CFA1], SM1STS[CFA0], SM1STS[CFB1], SM1STS[CFB0], SM1STS[CFX1], SM1STS[CFX0]	SM1INTEN[CFA1IE], SM1INTEN[CFA0IE], SM1INTEN[CFB1IE], SM1INTEN[CFB0IE], SM1INTEN[CFX1IE], SM1INTEN[CFX0IE]	Submodule 1 input capture interrupt	Input capture event has occurred

Table continues on the next page...

Table 26-3. Interrupt Summary (continued)

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
PWM_RELOAD1	SM1STS[RF]	SM1INTEN[RIE]	Submodule 1 reload interrupt	Reload event has occurred
PWM_CMP2	SM2STS[CMPF]	SM2INTEN[CMPIE]	Submodule 2 compare interrupt	Compare event has occurred
PWM_CAP2	SM2STS[CFA1], SM2STS[CFA0], SM2STS[CFB1], SM2STS[CFB0], SM2STS[CFX1], SM2STS[CFX0]	SM2INTEN[CFA1IE], SM2INTEN[CFA0IE], SM2INTEN[CFB1IE], SM2INTEN[CFB0IE], SM2INTEN[CFX1IE], SM2INTEN[CFX0IE	Submodule 2 input capture interrupt	Input capture event has occurred
PWM_RELOAD2	SM2STS[RF]	SM2INTEN[RIE]	Submodule 2 reload interrupt	Reload event has occurred
PWM_CMP3	SM3STS[CMPF]	SM3INTEN[CMPIE]	Submodule 3 compare interrupt	Compare event has occurred
PWM_CAP3	SM3STS[CFA1], SM3STS[CFA0], SM3STS[CFB1], SM3STS[CFB0], SM3STS[CFX1], SM3STS[CFX0]	SM3INTEN[CFA1IE], SM3INTEN[CFA0IE], SM3INTEN[CFB1IE], SM3INTEN[CFB0IE], SM3INTEN[CFX1IE], SM3INTEN[CFX0IE	Submodule 3 input capture interrupt	Input capture event has occurred
PWM_RELOAD3	SM3STS[RF]	SM3INTEN[RIE]	Submodule 3 reload interrupt	Reload event has occurred
PWM_RERR	SM0STS[REF]	SM0INTEN[REIE]	Submodule 0 reload error interrupt	Reload error has occurred
	SM1STS[REF]	SM1INTEN[REIE]	Submodule 1 reload error interrupt	
	SM2STS[REF]	SM2INTEN[REIE]	Submodule 2 reload error interrupt	
	SM3STS[REF]	SM3INTEN[REIE]	Submodule 3 reload error interrupt	
PWM_FAULT	FSTS0[FFLAG], FSTS1[FFLAG]	FCTRL0[FIE], FCTRL1[FIE]	Fault input interrupt	Fault condition has been detected

26.4.8 DMA

Each submodule can request a DMA read access for its capture FIFOs and a DMA write request for its double buffered registers.

Table 26-4. DMA summary

DMA request	DMA enable	Name	Description
Submodule 0 read request	SM0DMAEN[CX0DE]	SM0 Capture FIFO X0 read request	SM0CVAL0 contains a value to be read
	SM0DMAEN[CX1DE]	SM0 Capture FIFO X1 read request	SM0CVAL1 contains a value to be read
	SM0DMAEN[CA0DE]	SM0 Capture FIFO A0 read request	SM0CVAL2 contains a value to be read
	SM0DMAEN[CA1DE]	SM0 Capture FIFO A1 read request	SM0CVAL3 contains a value to be read
	SM0DMAEN[CB0DE]	SM0 Capture FIFO B0 read request	SM0CVAL4 contains a value to be read
	SM0DMAEN[CB1DE]	SM0 Capture FIFO B1 read request	SM0CVAL5 contains a value to be read
	SM0DMAEN[CAPTDE]	SM0 Capture FIFO read request source select	Selects source of submodule0 read DMA request
Submodule 0 write request	SM0DMAEN[VALDE]	SM0VALx write request	SM0VALx and SM0FRACVALx registers need to be updated
Submodule 1 read request	SM1DMAEN[CX0DE]	SM1 Capture FIFO X0 read request	SM1CVAL0 contains a value to be read
	SM1DMAEN[CX1DE]	SM1 Capture FIFO X1 read request	SM1CVAL1 contains a value to be read
	SM1DMAEN[CA0DE]	SM1Capture FIFO A0 read request	SM1CVAL2 contains a value to be read
	SM1DMAEN[CA1DE]	SM1 Capture FIFO A1 read request	SM1CVAL3 contains a value to be read
	SM1DMAEN[CB0DE]	SM1 Capture FIFO B0 read request	SM1CVAL4 contains a value to be read
	SM1DMAEN[CB1DE]	SM1 Capture FIFO B1 read request	SM1CVAL5 contains a value to be read
	SM1DMAEN[CAPTDE]	SM1 Capture FIFO read request source select	Selects source of submodule1 read DMA request
Submodule 1 write request	SM1DMAEN[VALDE]	SM1VALx write request	SM1VALx and SM1FRACVALx registers need to be updated
Submodule 2 read request	SM2DMAEN[CX0DE]	SM2 Capture FIFO X0 read request	SM2CVAL0 contains a value to be read
	SM2DMAEN[CX1DE]	SM2 Capture FIFO X1 read request	SM2CVAL1 contains a value to be read
	SM2DMAEN[CA0DE]	SM2 Capture FIFO A0 read request	SM2CVAL2 contains a value to be read
	SM2DMAEN[CA1DE]	SM2 Capture FIFO A1 read request	SM2CVAL3 contains a value to be read
	SM2DMAEN[CB0DE]	SM2 Capture FIFO B0 read request	SM2CVAL4 contains a value to be read
	SM2DMAEN[CB1DE]	SM2 Capture FIFO B1 read request	SM2CVAL5 contains a value to be read

Table continues on the next page...

Table 26-4. DMA summary (continued)

DMA request	DMA enable	Name	Description
	SM2DMAEN[CAPTDE]	SM2 Capture FIFO read request source select	Selects source of submodule2 read DMA request
Submodule 2 write request	SM2DMAEN[VALDE]	SM2VALx write request	SM2VALx and SM2FRACVALx registers need to be updated
Submodule 3 read request	SM3DMAEN[CX0DE]	SM3 Capture FIFO X0 read request	SM3CVAL0 contains a value to be read
	SM3DMAEN[CX1DE]	SM3 Capture FIFO X1 read request	SM3CVAL1 contains a value to be read
	SM3DMAEN[CA0DE]	SM3 Capture FIFO A0 read request	SM3CVAL2 contains a value to be read
	SM3DMAEN[CA1DE]	SM3 Capture FIFO A1 read request	SM3CVAL3 contains a value to be read
	SM3DMAEN[CB0DE]	SM3 Capture FIFO B0 read request	SM3CVAL4 contains a value to be read
	SM3DMAEN[CB1DE]	SM3 Capture FIFO B1 read request	SM3CVAL5 contains a value to be read
	SM3DMAEN[CAPTDE]	SM3 Capture FIFO read request source select	Selects source of submodule3 read DMA request
Submodule 3 write request	SM3DMAEN[VALDE]	SM3VALx write request	SM3VALx and SM3FRACVALx registers need to be updated

26.5 External signals

The PWM has pins named PWM_An, PWM_Bn, PWM_Xn, FAULTn, EXT_SYNC, EXT_FORCE, PWMn_EXTA, and PWMn_EXTB. The PWM also has an on-chip input called EXT_CLK and output signals called PWMn_OUT_TRIGx and PWMn_MUX_TRIGx.

26.5.1 PWM_An and PWM_Bn - External PWM output pair

These pins are the output pins of the PWM channels. These pins can be independent PWM signals or a complementary pair. When not needed as an output, they can be used as inputs to the input capture circuitry.

26.5.2 PWM_Xn - Auxiliary PWM output signal

These pins are the auxiliary output pins of the PWM channels. They can be independent PWM signals. When not needed as an output, they can be used as inputs to the input capture circuitry or used to detect the polarity of the current flowing through the complementary circuit at deadtime correction.

26.5.3 FAULTn - Fault Inputs

These are input pins for disabling selected PWM outputs.

26.5.4 EXT_SYNC - External synchronization signal

These input signals allow a source external to the PWM to initialize the PWM counter. Therefore, the PWM can be synchronized to external circuitry.

26.5.5 EXT_FORCE - External output force signal

This input signal allows a source external to the PWM to force an update of the PWM outputs. Therefore, the PWM can be synchronized to external circuitry.

26.5.6 PWMn_EXTA and PWMn_EXTB - Alternate PWM control signals

These pins allow an alternate source to control the PWM_An and PWM_Bn outputs. Typically, either the PWMn_EXTA or PWMn_EXTB input (depending on the state of MCTRL[IPOL]) is used for the generation of a complementary pair. Typical control signals include ADC conversion high/low limits, TMR outputs, GPIO inputs, and comparator outputs.

For pin input details, see chip-specific eFlexPWM information.

26.5.7 PWMn_OUT_TRIG0 and PWMn_OUT_TRIG1 - Output triggers

These outputs allow the PWM submodules to control timing of ADC conversions. See the description of the SMnTCTRL[OUT_TRIG_EN] for information about how to enable these outputs and how the compare registers match up to the output triggers.

26.5.8 PWM[n]_MUX_TRIG0 and PWM[n]_MUX_TRIG1 - Output triggers

These outputs can be either PWMn_OUT_TRIG0/PWMn_OUT_TRIG1 signals or PWM_A/PWM_B signals from output logic. See the description of the PWAOT0 and PWBOT1 bits in the Output Trigger Control Register for information about how to enable these outputs.

26.5.9 EXT_CLK - External clock signal

This signal allows a source external to the PWM (typically a timer or an off-chip source) to control the PWM clocking. Therefore, the PWM can be synchronized to the timer, or multiple chips can be synchronized to each other.

Chapter 27
Quad Timer (TMR)

27.1 Overview

Each timer module (TMR) contains four identical counter/timer groups. Each 16-bit counter/timer group contains a prescaler, a counter, a load register, a hold register, a capture register, two compare registers, two status and control registers, and one control register. All of the registers except the prescaler are read/writable.

NOTE

This document uses the terms "Timer" and "Counter" interchangeably because the counter/timers may perform either or both tasks.

The load register provides the initialization value to the counter when the counter's terminal value has been reached.

The hold register captures the counter's value when other counters are being read. This feature supports the reading of cascaded counters.

The capture register enables an external signal to take a "snap shot" of the counter's current value.

The COMP1 and COMP2 registers provide the values to which the counter is compared. If a match occurs, the OFLAG (TMR Output signal) can be set, cleared, or toggled. At match time, an interrupt is generated if enabled, and the new compare value is loaded into the COMP1 or COMP2 registers from CMPLD1 and CMPLD2 if enabled.

The prescaler provides different time bases useful for clocking the counter/timer.

The counter provides the ability to count internal or external events.

Within a timer module (set of four timer/counters), the input pins are shareable.

27.2 Features

The TMR module design includes these distinctive features:

- Four 16-bit counters/timers
- Count up/down
- Counters are cascadable
- Programmable count modulo
- Max count rate equals peripheral clock/2 for external clocks
- Max count rate equals peripheral clock for internal clocks
- Count once or repeatedly
- Counters are preloadable
- Compare registers are preloadable (available with compare load feature)
- Counters can share available input pins
- Separate prescaler for each counter
- Each counter has capture and compare capability
- Programmable operation during debug mode
- Inputs may act as fault inputs
- Programmable input filter
- Counting start can be synchronized across counters

27.3 Modes of Operation

The TMR module design operates in only a single mode of operation: Functional Mode. The various counting modes are detailed in the Functional Description.

27.4 Block Diagram

Each of the timer/counter groups within the quad-timer are shown in this figure.

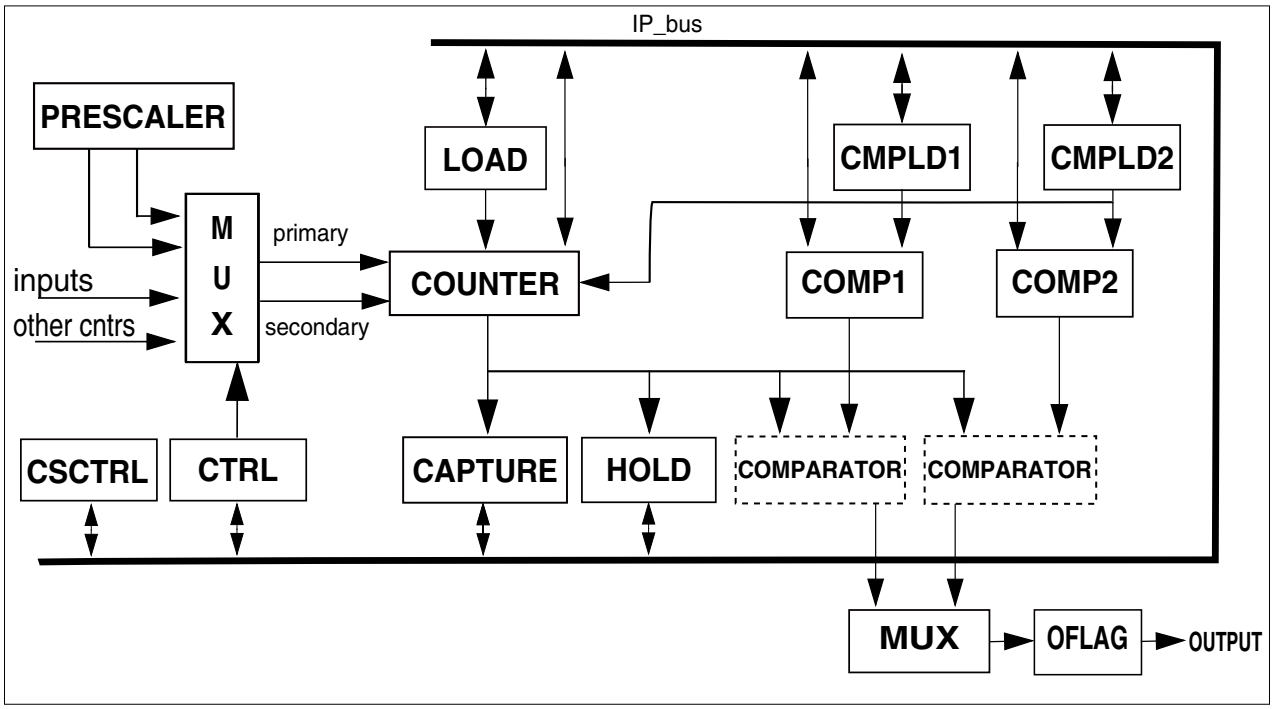


Figure 27-1. Quad Timer Block Diagram

27.5 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 and the address offset is defined at the module level. Make certain to check which quad timer is available on the chip being used, and which timer channels have external I/O.

TMR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E140	Timer Channel Compare Register 1 (TMRA_COMP10)	16	R/W	0000h	27.5.1/827
E141	Timer Channel Compare Register 2 (TMRA_COMP20)	16	R/W	0000h	27.5.2/828
E142	Timer Channel Capture Register (TMRA_CAPT0)	16	R/W	0000h	27.5.3/828
E143	Timer Channel Load Register (TMRA_LOAD0)	16	R/W	0000h	27.5.4/828
E144	Timer Channel Hold Register (TMRA_HOLD0)	16	R/W	0000h	27.5.5/829
E145	Timer Channel Counter Register (TMRA_CNTR0)	16	R/W	0000h	27.5.6/829
E146	Timer Channel Control Register (TMRA_CTRL0)	16	R/W	0000h	27.5.7/829
E147	Timer Channel Status and Control Register (TMRA_SCTRL0)	16	R/W	0000h	27.5.8/832

Table continues on the next page...

TMR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E148	Timer Channel Comparator Load Register 1 (TMRA_CMPLD10)	16	R/W	0000h	27.5.9/833
E149	Timer Channel Comparator Load Register 2 (TMRA_CMPLD20)	16	R/W	0000h	27.5.10/834
E14A	Timer Channel Comparator Status and Control Register (TMRA_CSCTRL0)	16	R/W	0000h	27.5.11/834
E14B	Timer Channel Input Filter Register (TMRA_FILT0)	16	R/W	0000h	27.5.12/836
E14C	Timer Channel DMA Enable Register (TMRA_DMA0)	16	R/W	0000h	27.5.13/837
E14F	Timer Channel Enable Register (TMRA_ENBL)	16	R/W	000Fh	27.5.14/838
E150	Timer Channel Compare Register 1 (TMRA_COMP11)	16	R/W	0000h	27.5.1/827
E151	Timer Channel Compare Register 2 (TMRA_COMP21)	16	R/W	0000h	27.5.2/828
E152	Timer Channel Capture Register (TMRA_CAPT1)	16	R/W	0000h	27.5.3/828
E153	Timer Channel Load Register (TMRA_LOAD1)	16	R/W	0000h	27.5.4/828
E154	Timer Channel Hold Register (TMRA_HOLD1)	16	R/W	0000h	27.5.5/829
E155	Timer Channel Counter Register (TMRA_CNTR1)	16	R/W	0000h	27.5.6/829
E156	Timer Channel Control Register (TMRA_CTRL1)	16	R/W	0000h	27.5.7/829
E157	Timer Channel Status and Control Register (TMRA_SCTRL1)	16	R/W	0000h	27.5.8/832
E158	Timer Channel Comparator Load Register 1 (TMRA_CMPLD11)	16	R/W	0000h	27.5.9/833
E159	Timer Channel Comparator Load Register 2 (TMRA_CMPLD21)	16	R/W	0000h	27.5.10/834
E15A	Timer Channel Comparator Status and Control Register (TMRA_CSCTRL1)	16	R/W	0000h	27.5.11/834
E15B	Timer Channel Input Filter Register (TMRA_FILT1)	16	R/W	0000h	27.5.12/836
E15C	Timer Channel DMA Enable Register (TMRA_DMA1)	16	R/W	0000h	27.5.13/837
E160	Timer Channel Compare Register 1 (TMRA_COMP12)	16	R/W	0000h	27.5.1/827
E161	Timer Channel Compare Register 2 (TMRA_COMP22)	16	R/W	0000h	27.5.2/828
E162	Timer Channel Capture Register (TMRA_CAPT2)	16	R/W	0000h	27.5.3/828
E163	Timer Channel Load Register (TMRA_LOAD2)	16	R/W	0000h	27.5.4/828
E164	Timer Channel Hold Register (TMRA_HOLD2)	16	R/W	0000h	27.5.5/829
E165	Timer Channel Counter Register (TMRA_CNTR2)	16	R/W	0000h	27.5.6/829
E166	Timer Channel Control Register (TMRA_CTRL2)	16	R/W	0000h	27.5.7/829
E167	Timer Channel Status and Control Register (TMRA_SCTRL2)	16	R/W	0000h	27.5.8/832
E168	Timer Channel Comparator Load Register 1 (TMRA_CMPLD12)	16	R/W	0000h	27.5.9/833
E169	Timer Channel Comparator Load Register 2 (TMRA_CMPLD22)	16	R/W	0000h	27.5.10/834
E16A	Timer Channel Comparator Status and Control Register (TMRA_CSCTRL2)	16	R/W	0000h	27.5.11/834

Table continues on the next page...

TMR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E16B	Timer Channel Input Filter Register (TMRA_FILT2)	16	R/W	0000h	27.5.12/836
E16C	Timer Channel DMA Enable Register (TMRA_DMA2)	16	R/W	0000h	27.5.13/837
E170	Timer Channel Compare Register 1 (TMRA_COMP13)	16	R/W	0000h	27.5.1/827
E171	Timer Channel Compare Register 2 (TMRA_COMP23)	16	R/W	0000h	27.5.2/828
E172	Timer Channel Capture Register (TMRA_CAPT3)	16	R/W	0000h	27.5.3/828
E173	Timer Channel Load Register (TMRA_LOAD3)	16	R/W	0000h	27.5.4/828
E174	Timer Channel Hold Register (TMRA_HOLD3)	16	R/W	0000h	27.5.5/829
E175	Timer Channel Counter Register (TMRA_CNTR3)	16	R/W	0000h	27.5.6/829
E176	Timer Channel Control Register (TMRA_CTRL3)	16	R/W	0000h	27.5.7/829
E177	Timer Channel Status and Control Register (TMRA_SCTRL3)	16	R/W	0000h	27.5.8/832
E178	Timer Channel Comparator Load Register 1 (TMRA_CMPLD13)	16	R/W	0000h	27.5.9/833
E179	Timer Channel Comparator Load Register 2 (TMRA_CMPLD23)	16	R/W	0000h	27.5.10/834
E17A	Timer Channel Comparator Status and Control Register (TMRA_CSCTRL3)	16	R/W	0000h	27.5.11/834
E17B	Timer Channel Input Filter Register (TMRA_FILT3)	16	R/W	0000h	27.5.12/836
E17C	Timer Channel DMA Enable Register (TMRA_DMA3)	16	R/W	0000h	27.5.13/837

27.5.1 Timer Channel Compare Register 1 (TMRx_COMP1*n*)

Address: E140h base + 0h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	COMPARISON_1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMRx_COMP1*n* field descriptions

Field	Description
COMPARISON_1	Comparison Value 1 This read/write register stores the value used for comparison with the counter value in count up mode.

27.5.2 Timer Channel Compare Register 2 (TMR*x*_COMP2*n*)

Address: E140h base + 1h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	COMPARISON_2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR*x*_COMP2*n* field descriptions

Field	Description
COMPARISON_2	Comparison Value 2 This read/write register stores the value used for comparison with the counter value in count down mode or alternating compare mode.

27.5.3 Timer Channel Capture Register (TMR*x*_CAPT*n*)

Address: E140h base + 2h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	CAPTURE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR*x*_CAPT*n* field descriptions

Field	Description
CAPTURE	Capture Value This read/write register stores the value captured from the counter.

27.5.4 Timer Channel Load Register (TMR*x*_LOAD*n*)

Address: E140h base + 3h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	LOAD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR*x*_LOAD*n* field descriptions

Field	Description
LOAD	Timer Load Register This read/write register stores the value used to initialize the counter after counter compare or re-initialization.

27.5.5 Timer Channel Hold Register (TMR*x*_HOLD*n*)

Address: E140h base + 4h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	HOLD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR*x*_HOLD*n* field descriptions

Field	Description
HOLD	This read/write register stores the counter's values of specific channels whenever any of the four counters within a module is read.

27.5.6 Timer Channel Counter Register (TMR*x*_CNTR*n*)

Address: E140h base + 5h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	COUNTER															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR*x*_CNTR*n* field descriptions

Field	Description
COUNTER	This read/write register is the counter for the corresponding channel in a timer module.

27.5.7 Timer Channel Control Register (TMR*x*_CTRL*n*)

Address: E140h base + 6h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read Write	CM			PCS				SCS
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	SCS	ONCE	LENGTH	DIR	COINIT	OUTMODE		
Reset	0	0	0	0	0	0	0	0

TMR*x*_CTRL*n* field descriptions

Field	Description
15–13 CM	Count Mode These bits control the basic counting and behavior of the counter.

Table continues on the next page...

TMRx_CTRL*n* field descriptions (continued)

Field	Description
	000 No operation 001 Count rising edges of primary source[1] 010 Count rising and falling edges of primary source[2] 011 Count rising edges of primary source while secondary input high active 100 Quadrature count mode, uses primary and secondary sources 101 Count rising edges of primary source; secondary source specifies direction[3] 110 Edge of secondary source triggers primary count until compare 111 Cascaded counter mode (up/down)[4]
12–9 PCS	Primary Count Source These bits select the primary count source. **NOTE:** A timer selecting its own output for input is not a legal choice. The result is no counting. 0000 Counter 0 input pin 0001 Counter 1 input pin 0010 Counter 2 input pin 0011 Counter 3 input pin 0100 Counter 0 output 0101 Counter 1 output 0110 Counter 2 output 0111 Counter 3 output 1000 IP bus clock divide by 1 prescaler 1001 IP bus clock divide by 2 prescaler 1010 IP bus clock divide by 4 prescaler 1011 IP bus clock divide by 8 prescaler 1100 IP bus clock divide by 16 prescaler 1101 IP bus clock divide by 32 prescaler 1110 IP bus clock divide by 64 prescaler 1111 IP bus clock divide by 128 prescaler
8–7 SCS	Secondary Count Source These bits identify the external input pin to be used as a count command or timer command. The selected input can trigger the timer to capture the current value of CNTR . The selected input can also be used to specify the count direction. The selected signal can also be used as a fault input when CSCTRL[FAULT] is set. The polarity of the signal can be inverted by SCTRL[IPS]. 00 Counter 0 input pin 01 Counter 1 input pin 10 Counter 2 input pin 11 Counter 3 input pin
6 ONCE	Count Once This bit selects continuous or one shot counting mode. 0 Count repeatedly. 1 Count until compare and then stop. If counting up, a successful compare occurs when the counter reaches a COMP1 value. If counting down, a successful compare occurs when the counter reaches a COMP2 value. When output mode $4 is used, the counter re-initializes after reaching the COMP1 value, continues to count to the COMP2 value, and then stops.

Table continues on the next page...

TMRx_CTRL*n* field descriptions (continued)

Field	Description
5 LENGTH	Count Length This bit determines whether the counter: • counts to the compare value and then re-initializes itself to the value specified in the LOAD (or CMPLD2) register, or • continues counting past the compare value to the binary roll over. 0 Count until roll over at $FFFF and then continue by re-initializing the counter from the LOAD register. 1 Count until compare, then re-initialize using the LOAD regsiter. If counting up, a successful compare occurs when the counter reaches a COMP1 value. If counting down, a successful compare occurs when the counter reaches a COMP2 value. When output mode $4 is used, alternating values of COMP1 and COMP2 are used to generate successful comparisons. For example, the counter counts until a COMP1 value is reached, re-initializes, counts until COMP2 value is reached, re-initializes, counts until COMP1 value is reached, and so on.
4 DIR	Count Direction This bit selects either the normal count direction up, or the reverse direction, down. 0 Count up. 1 Count down.
3 COINIT	Co-Channel Initialization This bit enables another counter/timer within the module to force the re-initialization of this counter/timer when it has an active compare event. 0 Co-channel counter/timers cannot force a re-initialization of this counter/timer 1 Co-channel counter/timers may force a re-initialization of this counter/timer
OUTMODE	Output Mode These bits determine the mode of operation for the OFLAG output signal. 000 Asserted while counter is active 001 Clear OFLAG output on successful compare 010 Set OFLAG output on successful compare 011 Toggle OFLAG output on successful compare 100 Toggle OFLAG output using alternating compare registers 101 Set on compare, cleared on secondary source input edge 110 Set on compare, cleared on counter rollover 111 Enable gated clock output while counter is active

1. Rising edges are counted only when SCTRL[IPS] = 0. Falling edges are counted when SCTRL[IPS] = 1. If the primary count source is IP bus clock divide by 1, only rising edges are counted regardless of the value of SCTRL[IPS].
2. IP bus clock divide by 1 cannot be used as a primary count source in edge count mode.
3. Rising edges are counted only when SCTRL[IPS] = 0. Falling edges are counted when SCTRL[IPS] = 1.
4. The primary count source must be set to one of the counter outputs.

27.5.8 Timer Channel Status and Control Register (TMRx_SCTRL*n*)

Address: E140h base + 7h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	CAPTURE_MODE		MSTR	EEOF	VAL	0	OPS	OEN
Write						FORCE		
Reset	0	0	0	0	0	0	0	0

TMRx_SCTRL*n* field descriptions

Field	Description
15 TCF	Timer Compare Flag This bit is set when a successful compare occurs. This bit is cleared by writing a zero to this bit location.
14 TCFIE	Timer Compare Flag Interrupt Enable This bit (when set) enables interrupts when TCF is set.
13 TOF	Timer Overflow Flag This bit is set when the counter rolls over its maximum value $FFFF or $0000 (depending on count direction). This bit is cleared by writing a zero to this bit location.
12 TOFIE	Timer Overflow Flag Interrupt Enable This bit (when set) enables interrupts when TOF is set.
11 IEF	Input Edge Flag This bit is set when CAPTMODE is enabled and a proper input transition occurs (on an input selected as a secondary count source) while the count mode does not equal 000. This bit is cleared by writing a zero to this bit position. This bit can also be cleared automatically by a read of CAPT when DMA[IEFDE] is set. **NOTE:** Setting the input polarity select bit (IPS) changes the edge to be detected. Also, the control register's secondary count source determines which external input pin is monitored by the detection circuitry.
10 IEFIE	Input Edge Flag Interrupt Enable This bit (when set) enables interrupts when IEF is set. **Restriction:** Do not set both this bit and DMA[IEFDE].
9 IPS	Input Polarity Select This bit (when set) inverts the input signal polarity.
8 INPUT	External Input Signal This read-only bit reflects the current state of the external input pin selected via the secondary count source after application of IPS and filtering.

Table continues on the next page...

TMR*x*_SCTRL*n* field descriptions (continued)

Field	Description
7–6 CAPTURE_ MODE	Input Capture Mode These bits specify the operation of the capture register as well as the operation of the input edge flag. The input source is the secondary count source. 00 Capture function is disabled 01 Load capture register on rising edge (when IPS=0) or falling edge (when IPS=1) of input 10 Load capture register on falling edge (when IPS=0) or rising edge (when IPS=1) of input 11 Load capture register on both edges of input
5 MSTR	Master Mode This bit (when set) enables the compare function's output to be broadcasted to the other counters/timers in the module. This signal then can be used to re-initialize the other counters and/or force their OFLAG signal outputs.
4 EEOF	Enable External OFLAG Force This bit (when set) enables the compare from another counter/timer within the same module to force the state of this counter's OFLAG output signal.
3 VAL	Forced OFLAG Value This bit determines the value of the OFLAG output signal when software triggers a FORCE command.
2 FORCE	Force OFLAG Output This write only bit forces the current value of VAL to be written to the OFLAG output. This bit always reads as a zero. VAL and FORCE can be written simultaneously in a single write operation. Write to FORCE only if the counter is disabled. Setting this bit while the counter is enabled may yield unpredictable results.
1 OPS	Output Polarity Select This bit determines the polarity of the OFLAG output signal. 0 True polarity. 1 Inverted polarity.
0 OEN	Output Enable This bit determines the direction of the external pin. 0 The external pin is configured as an input. 1 The OFLAG output signal is driven on the external pin. Other timer groups using this external pin as their input see the driven value. The polarity of the signal is determined by OPS.

27.5.9 Timer Channel Comparator Load Register 1 (TMR*x*_CMPLD1*n*)

Address: E140h base + 8h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	COMPARATOR_LOAD_1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMRx_CMPLD1*n* field descriptions

Field	Description
COMPARATOR_LOAD_1	This read/write register is the comparator 1 preload value for the COMP1 register for the corresponding channel in a timer module.

27.5.10 Timer Channel Comparator Load Register 2 (TMRx_CMPLD2*n*)

Address: E140h base + 9h offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	COMPARATOR_LOAD_2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMRx_CMPLD2*n* field descriptions

Field	Description
COMPARATOR_LOAD_2	This read/write register is the comparator 2 preload value for the COMP2 register for the corresponding channel in a timer module.

27.5.11 Timer Channel Comparator Status and Control Register (TMRx_CSCTRL*n*)

Address: E140h base + Ah offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read	DBG_EN		FAULT	ALT_LOAD	ROC	TCI	UP	OFLAG
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	TCF2EN	TCF1EN	TCF2	TCF1	CL2		CL1	
Write								
Reset	0	0	0	0	0	0	0	0

TMRx_CSCTRL*n* field descriptions

Field	Description
15–14 DBG_EN	Debug Actions Enable These bits allow the TMR module to perform certain actions in response to the chip entering debug mode. 00 Continue with normal operation during debug mode. (default) 01 Halt TMR counter during debug mode.

Table continues on the next page...

TMRx_CSCTRLn field descriptions (continued)

Field	Description
	10 Force TMR output to logic 0 (prior to consideration of SCTRL[OPS]). 11 Both halt counter and force output to 0 during debug mode.
13 FAULT	Fault Enable The selected secondary input acts as a fault signal so that the timer OFLAG is cleared when the secondary input is set. When the secondary input is used in this mode, there is no resynchronization of the input so that there is a combinational path to clear the OFLAG. Fault inputs less than two clock periods wide will not be latched. Latched faults will be cleared the next time that the counter logic sets the OFLAG. 0 Fault function disabled. 1 Fault function enabled.
12 ALT_LOAD	Alternative Load Enable This bit allows for an alternative method for loading the counter during modulo counting. Normally, the counter can be loaded only with the value from the LOAD register. When this bit is set, the counter is loaded from the LOAD register when counting up and a match with COMP1 occurs, or the counter is loaded from the CMPLD2 register when counting down and a match with COMP2 occurs. 0 Counter can be re-initialized only with the LOAD register. 1 Counter can be re-initialized with the LOAD or CMPLD2 registers depending on count direction.
11 ROC	Reload on Capture This bit enables the capture function to cause the counter to be reloaded from the LOAD register. 0 Do not reload the counter on a capture event. 1 Reload the counter on a capture event.
10 TCI	Triggered Count Initialization Control This bit is used during triggered count mode, CTRL[CM] = 110, to enable the counter to be re-initialized when a second trigger occurs while the counter is still counting. Normally, the second trigger causes the counting to stop/pause until a third trigger occurs. With this bit set, a second trigger event causes the counter to re-initialize and continue counting. 0 Stop counter upon receiving a second trigger event while still counting from the first trigger event. 1 Reload the counter upon receiving a second trigger event while still counting from the first trigger event.
9 UP	Counting Direction Indicator This read-only bit is used during quadrature count mode, CTRL[CM] = 100, to read the direction of the last count. CTRL[DIR] reverses the sense of this bit. 0 The last count was in the DOWN direction. 1 The last count was in the UP direction.
8 OFLAG	Output flag This read only bit shows the state of the Timer's internal OFLAG signal prior to consideration of polarity or debug mode.
7 TCF2EN	Timer Compare 2 Interrupt Enable An interrupt is issued when both this bit and TCF2 are set.
6 TCF1EN	Timer Compare 1 Interrupt Enable An interrupt is issued when both this bit and TCF1 are set.

Table continues on the next page...

TMRx_CSCTRL*n* field descriptions (continued)

Field	Description
5 TCF2	Timer Compare 2 Interrupt Flag When set, this bit indicates a successful comparison of the timer and the the COMP2 register has occurred. This bit is sticky, and will remain set until explicitly cleared by writing a zero to this bit location.
4 TCF1	Timer Compare 1 Interrupt Flag When set, this bit indicates a successful comparison of the timer and the the COMP1 register has occurred. This bit is sticky, and will remain set until explicitly cleared by writing a zero to this bit location.
3–2 CL2	Compare Load Control 2 These bits control when COMP2 is preloaded with the value from CMPLD2. 00 Never preload 01 Load upon successful compare with the value in COMP1 10 Load upon successful compare with the value in COMP2 11 Reserved
CL1	Compare Load Control 1 These bits control when COMP1 is preloaded with the value from CMPLD1. 00 Never preload 01 Load upon successful compare with the value in COMP1 10 Load upon successful compare with the value in COMP2 11 Reserved

27.5.12 Timer Channel Input Filter Register (TMRx_FILT*n*)

The FILT register programs the values for the filtering of the corresponding input without regard for the fact that any timer channel can use the input as a count source.

Input filter considerations:

- Set the FILT_PER value such that the sampling period is larger than the period of the expected noise. In this way, a noise spike will corrupt only one sample. Choose the FILT_CNT value to reduce the probability that noisy samples cause an incorrect transition to be recognized. The probability of an incorrect transistion is defined as the probability of an incorrect sample raised to the power of (FILT_CNT + 3).
- The values of FILT_PER and FILT_CNT must also be balanced against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting FILT_PER to a non-zero value) introduces a latency of (((FILT_CNT + 3) x FILT_PER) + 2) IP bus clock periods.

Address: E140h base + Bh offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0					FILT_CNT			FILT_PER							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR*x*_FILT*n* 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 FILT_CNT	Input Filter Sample Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. A value of 0x0 represents 3 samples. A value of 0x7 represents 10 samples. The value of FILT_CNT affects the input latency.
FILT_PER	Input Filter Sample Period These bits represent the sampling period (in IP bus clock cycles) of the TMR input signals. Each input is sampled multiple times at the rate specified by this field. If FILT_PER is 0x00 (default), then the input filter is bypassed. The value of FILT_PER affects the input latency. When changing values for FILT_PER from one non-zero value to another non-zero value, write a value of zero first to clear the filter.

27.5.13 Timer Channel DMA Enable Register (TMR*x*_DMA*n*)

Address: E140h base + Ch offset + (16d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0					CMPLD2DE	CMPLD1DE	IEFDE
Write								
Reset	0	0	0	0	0	0	0	0

TMR*x*_DMA*n* field descriptions

Field	Description
15–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 CMPLD2DE	Comparator Preload Register 2 DMA Enable Setting this bit enables DMA write requests for CMPLD2 whenever data is transferred out of the CMPLD2 register into the CNTR or COMP2 registers.
1 CMPLD1DE	Comparator Preload Register 1 DMA Enable Setting this bit enables DMA write requests for CMPLD1 whenever data is transferred out of the CMPLD1 register into the COMP1 register.

Table continues on the next page...

TMRx_DMA*n* field descriptions (continued)

Field	Description
0 IEFDE	Input Edge Flag DMA Enable Setting this bit enables DMA read requests for CAPT when SCTRL[IEF] is set. **Restriction:** Do not set both this bit and SCTRL[IEFIE].

27.5.14 Timer Channel Enable Register (TMRx_ENBL)

Address: E140h base + Fh offset = E14Fh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0												ENBL			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

TMRx_ENBL field descriptions

Field	Description
15–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ENBL	Timer Channel Enable These bits enable the prescaler (if it is being used) and counter in each channel. Multiple ENBL bits can be set at the same time to synchronize the start of separate counters. If an ENBL bit is set, then the corresponding channel starts its counter as soon as the CTRL[CM] field has a value other than 0. When an ENBL bit is clear, the corresponding counter maintains its current value. 0 Timer channel is disabled. 1 Timer channel is enabled. (default)

27.6 Functional Description

27.6.1 General

The counter/timer has two basic modes of operation: it can count internal or external events, or it can count an internal clock source while an external input signal is asserted, thus timing the width of the external input signal.

- The counter can count the rising, falling, or both edges of the selected input pin.
- The counter can decode and count quadrature encoded input signals.

- The counter can count up and down using dual inputs in a "count with direction" format.
- The counter's terminal count value (modulo) is programmable.
  - The value that is loaded into the counter after reaching its terminal count is programmable.
- The counter can count repeatedly, or it can stop after completing one count cycle.
- The counter can be programmed to count to a programmed value and then immediately reinitialize, or it can count through the compare value until the count "rolls over" to zero.

The external inputs to each counter/timer are shareable among each of the four counter/timers within the module. The external inputs can be used as:

- Count commands
- Timer commands
- They can trigger the current counter value to be "captured"
- They can be used to generate interrupt requests

The polarity of the external inputs are selectable.

The primary output of each timer/counter is the output signal OFLAG. The OFLAG output signal can be:

- Set, cleared, or toggled when the counter reaches the programmed value.
- The OFLAG output signal may be output to an external pin instead of having that pin serve as a timer input.
- The OFLAG output signal enables each counter to generate square waves, PWM, or pulse stream outputs.
- The polarity of the OFLAG output signal is selectable.

Any counter/timer can be assigned as a master. A master's compare signal can be broadcast to the other counter/timers within the module. The other counters can be configured to reinitialize their counters and/or force their OFLAG output signals to predetermined values when a master's counter/timer compare event occurs.

27.6.2 Usage of Compare Registers

The dual compare registers (COMP1 and COMP2) provide a bidirectional modulo count capability. The COMP1 register is used when the counter is *counting up*, and the COMP2 register is used when the counter is *counting down*. Alternating compare mode is the only exception.

The COMP1 register should be set to the desired maximum count value or FFFFh to indicate the maximum unsigned value prior to roll-over, and the COMP2 register should be set to the minimum count value or 0000h to indicate the minimum unsigned value prior to roll-under.

If CTRL[OUTMODE] is set to 100, the OFLAG will toggle while using alternating compare registers. In this variable frequency PWM mode, the COMP2 value defines the desired pulse width of the on time, and the COMP1 register defines the off time.

Use caution when changing COMP1 and COMP2 while the counter is active. If the counter has already passed the new value, it will count to FFFFh or 0000h, roll over, then begin counting toward the new value. The check is: Count=CMPx, *not* Count > COMP1 or Count < COMP2.

The use of the CMPLD1 and CMPLD2 registers to compare values will help to minimize this problem.

27.6.3 Usage of Compare Load Registers

The CMPLD1, CMPLD2, and CSCTRL registers offer a high degree of flexibility for loading compare registers with user-defined values on different compare events. To ensure correct functionality while using these registers we strongly suggest using the following method described in this section.

The purpose of the compare load feature is to allow quicker updating of the compare registers. In the past, a compare register could be updated using interrupts. However, because of the latency between an interrupt event occurring and the service of that interrupt, there was the possibility that the counter may have already counted past the new compare value by the time the compare register was updated by the interrupt service routine. The counter would then continue counting until it rolled over and reached the new compare value.

To address this, the compare registers are now updated in hardware in the same way the counter register is re-initialized to the value stored in the load register. The compare load feature allows the user to calculate new compare values and store them in to the

comparator load registers. When a compare event occurs, the new compare values in the comparator load registers are written to the compare registers eliminating the use of software to do this.

The compare load feature is intended to be used in variable frequency PWM mode. The COMP1 register determines the pulse width for the logic low part of OFLAG and COMP2 determines the pulse width for the logic high part of OFLAG. The period of the waveform is determined by the COMP1 and COMP2 values and the frequency of the primary clock source. See the following figure.

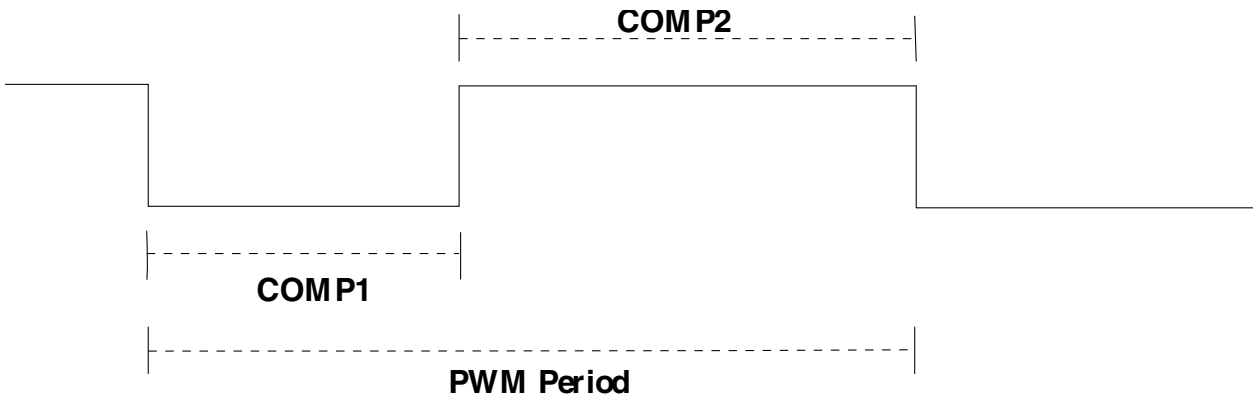


Figure 27-2. Variable PWM Waveform

Should we desire to update the duty cycle or period of the above waveform, we would need to update the COMP1 and COMP2 values using the compare load feature.

27.6.4 Usage of the Capture Register

The capture register stores a copy of the counter's value when an input edge (positive, negative, or both) is detected. After a capture event occurs, no further updating of the capture register will occur until the SCTRL[IEF] (input edge flag) is cleared by writing a zero to the SCTRL[IEF].

27.6.5 Functional Modes

The selected external count signals are sampled at the TMR's base clock rate and then run through a transition detector. The maximum count rate is one-half of the TMR's base clock rate. Internal clock sources can be used to clock the counters at the TMR's base clock rate.

If a counter is programmed to count to a specific value and then stop, the CTRL[CM] field is cleared when the count terminates.

27.6.5.1 Stop Mode

If CTRL[CM] is set to '000', the counter is inert. No counting will occur. Stop mode will also disable the interrupts caused by input transitions on a selected input pin.

27.6.5.2 Count Mode

If CTRL[CM] is set to '001', the counter will count the rising edges of the selected clock source. This mode is useful for generating periodic interrupts for timing purposes, or counting external events such as "widgets" on a conveyor belt passing a sensor. If the selected input is inverted by setting SCTRL[IPS] (input polarity select), then the negative edge of the selected external input signal is counted.

Example: 27.6.5.2.1 Count Pulses from External Source

```
//        (See Processor Expert PulseAccumulator bean.)
//    This example uses TMR1 to count pulse (actually counts rising edges of the pulse)
//      from an external source (QT3).
//
void Pulse_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=3,SCS=0,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR1_CTRL,0x0600);           /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x00);
  setReg(TMR1_CNTR,0x00);             /* Reset counter register */
  setReg(TMR1_LOAD,0x00);             /* Reset load register */
  setRegBitGroup(TMR1_CTRL,CM,0x01);  /* Run counter */
}
```

Example: 27.6.5.2.2 Generate Periodic Interrupt By Counting Internal Clocks

```
//         (See Processor Expert TimerInt bean.)
//   This example generates an interrupt every 100ms,
//     assuming the chip is operating at 60 MHz.
//
// It does this by using the IP_bus_clk divided by 128 as the counter clock source.
// The counter then counts to 46874 where it matches the COMP1 value.
// At that time an interrupt is generated, the counter is reloaded and
// the next COMP1 value is loaded from CMPLD1.
//
void TimerInt_Init(void)
{
  /* TMR0_CTRL: CM=0,PCS=0,SCS=0,ONCE=0,LENGTH=1,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR0_CTRL,0x20);             /* Stop all functions of the timer */
```

```
  /* TMR0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR0_SCTRL,0x00);
  setReg(TMR0_LOAD,0x00);             /* Reset load register */
  setReg(TMR0_COMP1,46874);           /* Set up compare 1 register */
  setReg(TMR0_CMPLD1,46874);          /* Also set the compare preload register */
  /* TMR0_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,TCF2EN=0,TCF1EN=1,
                  TCF2=0,TCF1=0,CL2=0,CL1=1 */
  setReg(TMR0_CSCTRL,0x41);           /* Enable compare 1 interrupt and */
                                       /*   compare 1 preload */
  setRegBitGroup(TMR0_CTRL,PCS,0xF); /* Primary Count Source to IP_bus_clk / 128 */
  setReg(TMR0_CNTR,0x00);             /* Reset counter register */
  setRegBitGroup(TMR0_CTRL,CM,0x01); /* Run counter */
}
```

27.6.5.3 Edge-Count Mode

If CTRL[CM] is set to '010', the counter will count both edges of the selected external clock source. This mode is useful for counting the changes in the external environment, such as a simple encoder wheel.

Example: 27.6.5.3.1 Count Both Edges of External Source Signal

```
//       (See Processor Expert PulseAccumulator bean.)
//  This example uses TMR1 to count pulse (actually counts both edges of the pulse)
//    from an external source (QT3).
//
void Pulse_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=3,SCS=0,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR1_CTRL,0x0600);           /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x00);
  setReg(TMR1_CNTR,0x00);             /* Reset counter register */
  setReg(TMR1_LOAD,0x00);             /* Reset load register */
  setRegBitGroup(TMR1_CTRL,CM,0x02);  /* Run counter */
}
```

27.6.5.4 Gated-Count Mode

If CTRL[CM] is set to '011', the counter will count while the selected secondary input signal is high. This mode is used to time the duration of external events. If the selected input is inverted by setting SCTRL[IPS] (input polarity select), then the counter will count while the selected secondary input is low.

Example: 27.6.5.4.1 Capture Duration of External Pulse

```
//       (See Processor Expert PulseAccumulator bean.)
//   This example uses TMR1 to determine the duration of an external pulse.
//
//   The IP_bus clock is used as the primary counter. If the duration of the
//     external pulse is longer than 0.001 seconds one of the other IP_bus clock
//     dividers can be used. If the pulse duration is longer than 0.128 seconds
//     an external clock source will have to be used as the primary clock source.
//
void Pulse1_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=8,SCS=1,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR1_CTRL,0x1080);             /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x00);
  setReg(TMR1_CNTR,0x00);              /* Reset counter register */
  setReg(TMR1_LOAD,0x00);              /* Reset load register */
  setRegBitGroup(TMR1_CTRL,CM,0x03);  /* Run counter */
}
```

27.6.5.5 Quadrature-Count Mode

If CTRL[CM] is set to '100', the counter will decode the primary and secondary external inputs as quadrature encoded signals. Quadrature signals are usually generated by rotary or linear sensors used to monitor movement of motor shafts or mechanical equipment. The quadrature signals are square waves that are 90 degrees out of phase. The decoding of quadrature signal provides both count and direction information.

This figure shows a timing diagram illustrating the basic operation of a quadrature incremental position encoder.

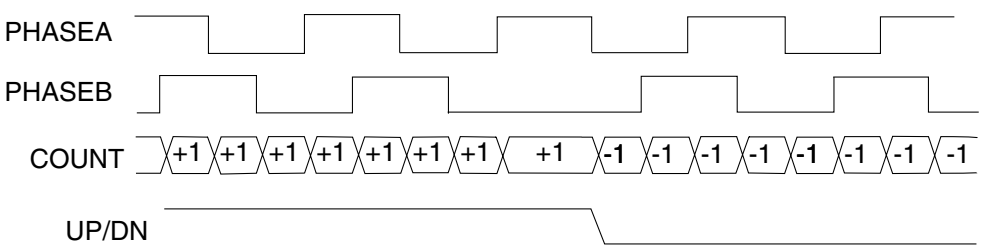


Figure 27-3. Quadrature Incremental Position Encoder

Example: 27.6.5.5.1 Quadrature Count Mode Example

```
//       (See Processor Expert PulseAccumulator bean.)
//  This example uses TMR0 for counting states of a quadrature position encoder.
//
//  Timer input 0 is used as the primary count source (PHASEA).
//  Timer input 1 is used as the secondary count source (PHASEB).
//
void Pulse_Init(void)
{
```

```
  /* TMR0_CTRL: CM=0,PCS=0,SCS=1,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMRC0_CTRL,0x80);              /* Set up mode */
  /* TMR0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=1 */
  setReg(TMR0_SCTRL,0x00);
  setReg(TMR0_CNTR,0x00);             /* Reset counter register */
  setReg(TMR0_LOAD,0x00);             /* Reset load register */
  setReg(TMR0_COMP1,0xFFFF);           /* Set up compare 1 register */
  setReg(TMR0_COMP2,0x00);             /* Set up compare 2 register */
  /* TMR0_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR0_CSCTRL,0x00);
  setRegBitGroup(TMR0_CTRL,CM,0x04);  /* Run counter */
}
```

27.6.5.6 Quadrature-Count Mode with Index Input

As an extension to the quadrature count mode discussed in the previous paragraph, some rotary shafts have a HOME or INDEX indicator. This would be a third input to the timer that is used to reset the timer's counter.

In this example, channel 0 is used to decode the quadrature inputs, but it doesn't actually count. Because its upper and lower limits are both set to 0, its output is cascaded count up and count down signals each time the quadrature inputs indicate a change in count. Channel 1 works in cascaded count mode receiving its counting instructions from channel 0. When an input capture event occurs, channel 1 is programmed to reset its counter value. The channel 1 counter contains the position value for the shaft.

Example: 27.6.5.6.1 Quadrature Count Mode with Index Input Example

```
//       (See Processor Expert PulseAccumulator bean.)
//  This example uses TMR0and TMR1 for counting states of a quadrature position encoder.
//
// Timer input 0 is used as the primary count source (PHASEA).
// Timer input 1 is used as the secondary count source (PHASEB).
// Timer input 2 is used as the index input source (INDEX).
//
void Pulse_Init(void)
{
  /* TMR0_CTRL: CM=0,PCS=0,SCS=1,ONCE=0,LENGTH=1,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR0_CTRL,0xA0);              /* Set up mode */
  /* TMR0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR0_SCTRL,0x00);
  setReg(TMR0_CNTR,0x00);             /* Reset counter register */
  setReg(TMR0_LOAD,0x00);             /* Reset load register */
  setReg(TMR0_COMP1,0x00);           /* Set up compare 1 register */
  setReg(TMR0_COMP2,0x00);             /* Set up compare 2 register */
  /* TMR0_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR0_CSCTRL,0x00);
/* TMR1_CTRL: CM=7,PCS=100,SCS=2,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR1_CTRL,0xEA00);            /* Set up capture edge */
/* TMR1_SCTRL: Capture_Mode=10 */
```

```
  setReg(TMR1_SCTRL,0x0080);
/* TMR1_CSTRL: ROC=1 */
  setReg(TMR1_CTRL,0x0800);           /* Set up reload on capture */
setRegBitGroup(TMR0_CTRL,CM,0x04);  /* Run counter */
}
```

27.6.5.7 Signed-Count Mode

If CTRL[CM] is set to '101', the counter counts the primary clock source while the selected secondary source provides the selected count direction (up/down).

Example: 27.6.5.7.1 Signed Count Mode Example

```
//       (See Processor Expert PulseAccumulator bean.)
//  This example uses TMR0 for signed mode counting.
//
//  Timer input 2 is used as the primary count source.
//  Timer input 1 is used to determine the count direction.
//
void Pulse_Init(void)
{
  /* TMR0_CTRL: CM=0,PCS=2,SCS=1,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR0_CTRL,0x0480);             /* Set up mode */
  /* TMR0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR0_SCTRL,0x1000);
  setReg(TMR0_CNTR,0x00);             /* Reset counter register */
  setReg(TMR0_LOAD,0x00);             /* Reset load register */
  setRegBitGroup(TMR0_CTRL,CM,0x05);  /* Run counter */
}
```

27.6.5.8 Triggered-Count Mode 1

If CSCTRL[TCI] is clear and CTRL[CM] is set to '110', the counter will begin counting the primary clock source after a positive transition (negative edge if SCTRL[IPS]=1) of the secondary input occurs. The counting will continue until a compare event occurs or another positive input transition is detected. If a second input transition occurs before a terminal count is reached, counting will stop and SCTRL[TCF] (timer compare flag) will be set. Subsequent secondary input transitions will continue to restart and stop the counting until a compare event occurs.

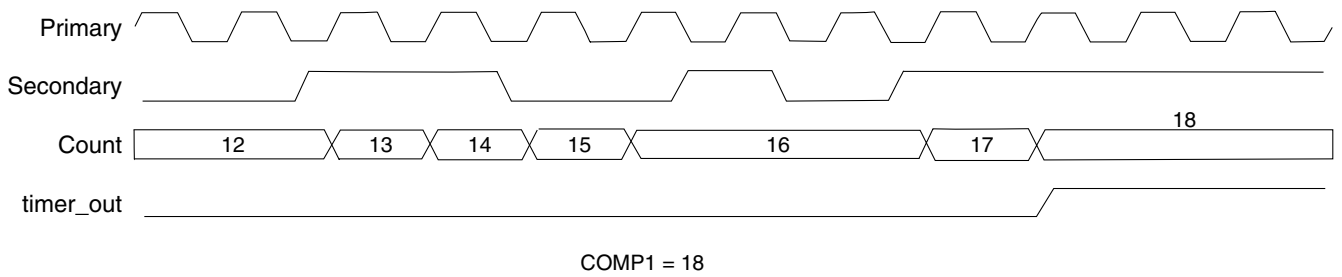


Figure 27-4. Triggered Count Mode 1 (CTRL[LENGTH]=0)

Example: 27.6.5.8.1 Triggered Count Mode 1 Example

```
//        (See Processor Expert PulseAccumulator bean.)
//   This example uses TMR1 for triggered mode counting.
//
//   Timer input 3 is used as the primary count source.
//   Timer input 2 is used for the trigger input.
//
void Pulse_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=3,SCS=2,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR1_CTRL,0x0700);            /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x1000);
  setReg(TMR1_CNTR,0x00);              /* Reset counter register */
  setReg(TMR1_LOAD,0x00);              /* Reset load register */
  setReg(TMR1_COMP1,0x0012);            /* Set up compare 1 register */
  /* TMR1_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR1_CSCTRL,0x00);
  setRegBitGroup(TMR1_CTRL,CM,0x06);    /* Run counter */
}
```

27.6.5.9 Triggered-Count Mode 2

If CSCTRL[TCI] is set and CTRL[CM] is set to '110', the counter will begin counting the primary clock source after a positive transition (negative edge if SCTRL[IPS]=1) of the secondary input occurs. The counting will continue until a compare event occurs or another positive input transition is detected. If a second input transition occurs before a terminal count was reached, the counter will reload and continue counting. When CSCTRL[TCI] is set, the OFLAG output mode, CTRL[OUTMODE], should probably be set to '101' (cleared on init, set on compare) to ensure the output will be in a known state after the second input transition and subsequent reload takes place.

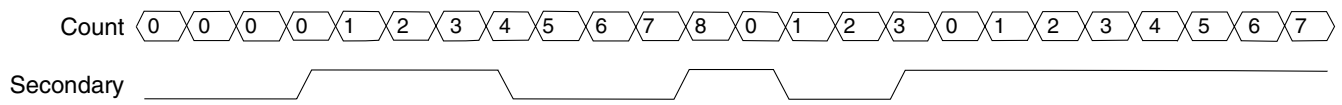


Figure 27-5. Triggered Count Mode 2 (CTRL[LENGTH]=0)

Example: 27.6.5.9.1 Triggered Count Mode 2Example

```
//        (See Processor Expert PulseAccumulator bean.)
//   This example uses TMR1 for triggered mode counting.
//
//   Timer input 3 is used as the primary count source.
//   Timer input 2 is used for the trigger input.
//
void Pulse_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=3,SCS=2,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR1_CTRL,0x0700);           /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x1000);

  setReg(TMR1_CNTR,0x00);             /* Reset counter register */
  setReg(TMR1_LOAD,0x00);             /* Reset load register */
  setReg(TMR1_COMP1,0x0012);           /* Set up compare 1 register */
  /* TMR1_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR1_CSCTRL,0x00);
  setRegBitGroup(TMR1_CTRL,CM,0x06); /* Run counter */
}
```

27.6.5.10 One-Shot Mode

If CTRL[CM] is set to '110', and the counter is set to reinitialize at a compare event (CTRL[LENGTH]=1),and CTRL[OUTMODE] is set to '101' (cleared on init, set on compare), the counter works in a one-shot mode. An external event causes the counter to count, and when the terminal count is reached, the output is asserted. This delayed output can be used to provide timing delays.

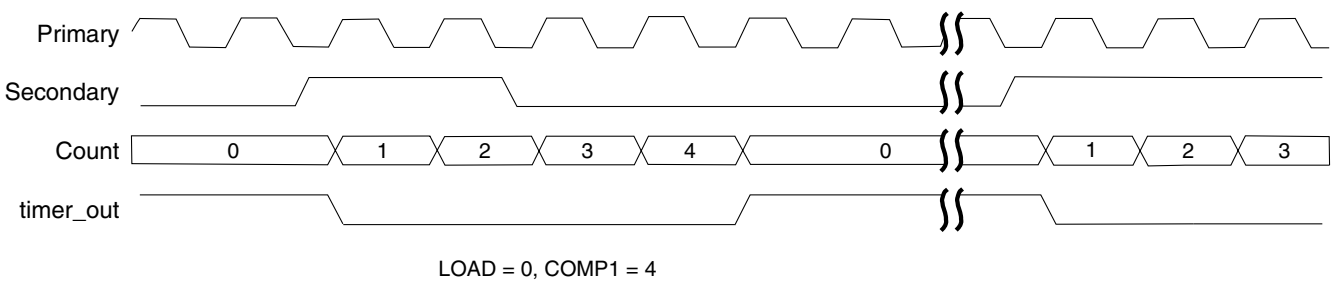


Figure 27-6. One-Shot Mode (CTRL[LENGTH]=1)

Example: 27.6.5.10.1 One-Shot Mode Example

```
//   (See Processor Expert PulseAccumulator bean.)
//  This example uses TMR1 for one-shot mode counting.
//
//  Timer input 3 is used as the primary count source.
//  Timer input 2 is used for the trigger input.
//
void Pulse_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=3,SCS=2,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=5 */
  setReg(TMR1_CTRL,0x0725);           /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
              Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x1000);
  setReg(TMR1_CNTR,0x00);             /* Reset counter register */
  setReg(TMR1_LOAD,0x00);             /* Reset load register */
  setReg(TMR1_COMP1,0x0004);           /* Set up compare 1 register */
  /* TMR1_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR1_CSCTRL,0x00);
  setRegBitGroup(TMR1_CTRL,CM,0x06);  /* Run counter */
}
```

27.6.5.11 Cascade-Count Mode

If CTRL[CM] is set to '111', the counter's input is connected to the output of another selected counter. The counter will count up and down as compare events occur in the selected source counter. This cascade or daisy-chained mode enables multiple counters to be cascaded to yield longer counter lengths. When operating in cascade mode, a special high-speed signal path is used between modules rather than the OFLAG output signal. If the selected source counter is counting up and it experiences a compare event, the counter will be incremented. If the selected source counter is counting down and it experiences a compare event, the counter will be decremented.

Up to four counters may be cascaded to create a 64-bit wide synchronous counter. Check the data sheet to see if there are any frequency limits for cascaded counting mode.

Whenever any counter is read within a counter module, all of the counters' values within the module are captured in their respective hold registers. This action supports the reading of a cascaded counter chain. First read any counter of a cascaded counter chain, then read the hold registers of the other counters in the chain. The cascaded counter mode is synchronous.

Note

It is possible to connect counters together by using the other (non-cascade) counter modes and selecting the outputs of other counters as a clock source. In this case, the counters are operating in a ripple mode, where higher order counters will transition a clock later than a purely synchronous design.

Example: 27.6.5.11.1 Generate Periodic Interrupt Cascading Two Counters

```
//    (See Processor Expert TimerInt bean.)
// This example generates an interrupt every 30 seconds,
//    assuming the chip is operating at 60 MHz.
//
// To do this, counter 2 is used to count 60,000 IP_bus clocks, which means it
//    will compare and reload every 0.001 seconds.
// Counter 3 is cascaded and used to count the 0.001 second ticks and
//    generate the desired interrupt interval.
//
void TimerInt_Init(void)
{
// Set counter 2 to count IP_bus clocks
   /* TMR2_CTRL: CM=0,PCS=8,SCS=0,ONCE=0,LENGTH=1,DIR=0,COINIT=0,OUTMODE=0 */
   setReg(TMR2_CTRL,0x1020);           /* Stop all functions of the timer */
// Set counter 3 as cascaded and to count counter 2 outputs
  /* TMR3_CTRL: CM=7,PCS=6,SCS=0,ONCE=0,LENGTH=1,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR3_CTRL,0xEC20);            /* Set up cascade counter mode */
  /* TMR3_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR3_SCTRL,0x00);
  /* TMR2_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR2_SCTRL,0x00);
  setReg(TMR3_CNTR,0x00);             /* Reset counter register */
  setReg(TMR2_CNTR,0x00);
  setReg(TMR3_LOAD,0x00);             /* Reset load register */
  setReg(TMR2_LOAD,0x00);
  setReg(TMR3_COMP1, 30000);           /* milliseconds in 30 seconds */
  setReg(TMR3_CMPLD1,30000);
  setReg(TMR2_COMP1, 60000);          // Set to cycle every milisecond
  setReg(TMR2_CMPLD1,60000);
  /* TMR3_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                  TCF2EN=0,TCF1EN=1,TCF2=0,TCF1=0,CL2=0,CL1=1 */
  setReg(TMR3_CSCTRL,0x41);           /* Enable compare 1 interrupt and */
                                       /* compare 1 preload             */
  /* TMR2_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=1 */
  setReg(TMR2_CSCTRL,0x01);           /* Enable Compare 1 preload */
  setRegBitGroup(TMR2_CTRL,CM,0x01);  /* Run counter */
}
```

27.6.5.12 Pulse-Output Mode

If CTRL[CM]=001, and CTRL[OUTMODE] is set to 111' (gated clock output), and CTRL[ONCE] is set, then the counter will output a pulse stream of pulses that has the same frequency of the selected clock source, and the number of output pulses is equal to the compare value minus the init value. This mode is useful for driving step motor systems.

Note

This does not work if CTRL[PCS] is set to 1000 (IP_bus/1).

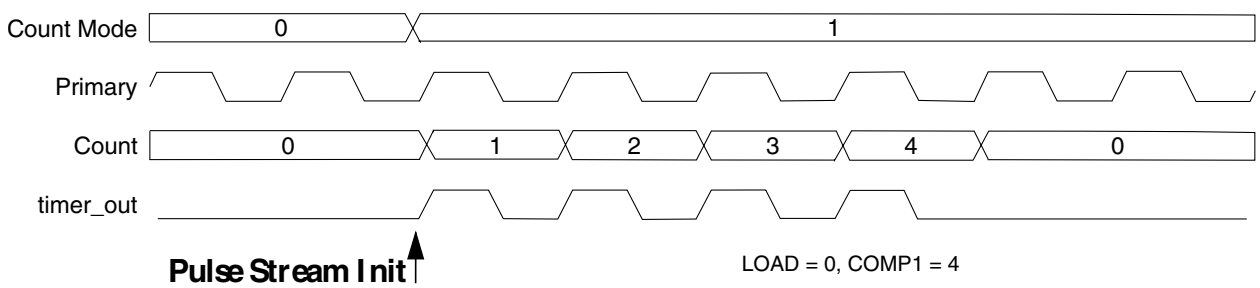


Figure 27-7. Pulse Output Mode

Example: 27.6.5.12.1 Pulse Outputs Using Two Counters

```
//     (See Processor Expert PulseStream bean.)
// This example generates six 10ms pulses, from QT1 output.
// Assuming the chip is operating at 60 MHz.
//
// To do this, timer 3 is used to generate a clock with a period of 10ms.
//
// Timer 1 is used to gate these clocks and count the number of pulses that have
//  been generated.
//
void PulseStream_Init(void)
{
// Select IP_bus_clk/16 as the clock source for Timer 3
  /* TMR3_CTRL: CM=0,PCS=0x0C,SCS=0,ONCE=0,LENGTH=1,DIR=0,COINIT=0,OUTMODE=3 */
   setReg(TMR3_CTRL,0x1823);             /* Set up mode */
  /* TMR3_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
   setReg(TMR3_SCTRL,0x00);
   setReg(TMR3_LOAD,0x00);              /* Reset load register */
   setReg(TMR3_COMP1,37500);            /* (16 * 37500 ) / 60e6 = 0.01 sec */
 /* TMR3_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR3_CSCTRL,0x00);             /* Set up comparator control register */



// Timer 3 output is the clock source for this timer.
  /* TMR1_CTRL: CM=0,PCS=7,SCS=0,ONCE=1,LENGTH=1,DIR=0,COINIT=0,OUTMODE=7 */
  setReg(TMR1_CTRL,0x0E67);             /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=1 */
```

```
   setReg(TMR1_SCTRL,0x01);
   setReg(TMR1_CNTR,0x00);             /* Reset counter register */
   setReg(TMR1_LOAD,0x00);             /* Reset load register */
   setReg(TMR1_COMP1,0x04);             /* Set up compare 1 register */

// set to interrupt after the last pulse
  /* TMR1_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                  TCF2EN=0,TCF1EN=1,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR1_CSCTRL,0x40);             /* Set up comparator control register */
// Finally, start the counters running
  setReg(TMR3_CNTR,0);                  /* Reset counter */
  setRegBitGroup(TMR3_CTRL,CM,0x01);    /* Run source clock counter */
  setRegBitGroup(TMR1_CTRL,CM,0x01);    /* Run counter */
}
```

27.6.5.13 Fixed-Frequency PWM Mode

If CTRL[CM]=001, count through roll-over (CTRL[LENGTH]=0), continuous count (CTRL[ONCE]=0) and CTRL[OUTMODE] is '110' (set on compare, cleared on counter roll-over), then the counter output yields a pulse-width modulated (PWM) signal with a frequency equal to the count clock frequency divided by 65,536 and a pulse-width duty cycle equal to the compare value divided by 65,536. This mode of operation is often used to drive PWM amplifiers used to power motors and inverters.

Example: 27.6.5.13.1 Fixed-Frequency PWM Mode Example

```
//    (See Processor Expert PWM bean.)
// This example uses TMR0 for Fixed-Frequency PWM mode timing.
//
// The timer will count IP_bus clocks continuously until it rolls over.
// This results in a PWM period of 65536 / 60e6 = 1092.267 usec
//
// Initially, an output pulse width of 25 usec is generated ( 1500 / 60e6 )
//   giving a PWM ratio of 1500 / 65536 = 2.289%
// This pulse width can be changed by changing the COMP1 register value (using CMPLD1).
//
void PWM1_Init(void)
{
  setReg(TMR0_CNTR,0);                 /* Reset counter */
  /* TMR0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=1,OPS=0,OEN=1 */
  setReg(TMR0_SCTRL,0x05);              /* Enable output */
  setReg(TMR0_COMP1,1500);          /* Store initial value to the duty-compare register */
  /* TMR0_CTRL: CM=1,PCS=8,SCS=0,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=6 */
  setReg(TMR0_CTRL,0x3006);      /* Run counter */
}
```

27.6.5.14 Variable-Frequency PWM Mode

If CTRL[CM]=001, count until compare (CTRL[LENGTH]=1), continuous count (CTRL[ONCE] = 0) and CTRL[OUTMODE] is '100' (toggle OFLAG and alternate compare registers), then the counter output yields a pulse-width modulated (PWM) signal whose frequency and pulse width is determined by the values programmed into the COMP1 and COMP2 registers, and the input clock frequency. This method of PWM generation has the advantage of allowing almost any desired PWM frequency and/or constant on or off periods. This mode of operation is often used to drive PWM amplifiers used to power motors and inverters. The CMPLD1 and CMPLD2 registers are especially useful for this mode, as they allow the programmer time to calculate values for the next PWM cycle while the PWM current cycle is underway.

To set up the TMR to run in variable frequency PWM mode with compare preload please use the following setup for the specific counter you would like to use. When performing the setup, update the TMR_CTRL register last because the counter will start counting if the count mode is changed to any value other than 000 (assuming the primary count source is already active).

Timer Control Register (CTRL)

- CM=001 (count rising edges of primary source)
- PCS=1000 (IP bus clock for best granularity for waveform timing)
- SCS=Any (ignored in this mode)
- ONCE=0 (want to count repeatedly)
- LENGTH=1 (want to count until compare value is reached and re-initialize counter register)
- DIR=Any (user's choice. The compare register values must be chosen carefully to account for things like roll-under, etc.)
- COINIT=0 (user can set this if they need this function)
- OUTMODE=100 (toggle OFLAG output using alternating compare registers)

Timer Status and Control Register (SCTRL)

- OEN = 1 (output enable to allow OFLAG output to be put on an external pin. Set this bit as needed.)

- OPS = Any (user's choice)
- Make sure the rest of the bits are cleared for this register. We will enable interrupts in the comparator status and control register instead of in this register.

Comparator Status and Control Register (CSCTRL)

- TCF2EN=1 (allow interrupt to be issued when CSCTRL[TCF2] is set)
- TCF1EN=0 (do not allow interrupt to be issued when CSCTRL[TCF1] is set)
- TCF1=0 (clear timer compare 1 interrupt source flag. This is set when counter register equals compare register 1 value and OFLAG is low)
- TCF2=0 (clear timer compare 2 interrupt source flag. This is set when counter register equals compare register 2 value and OFLAG is high)
- CL1=10 (load compare register when CSCTRL[TCF2] is asserted)
- CL2=01 (load compare register when CSCTRL[TCF1] is asserted)

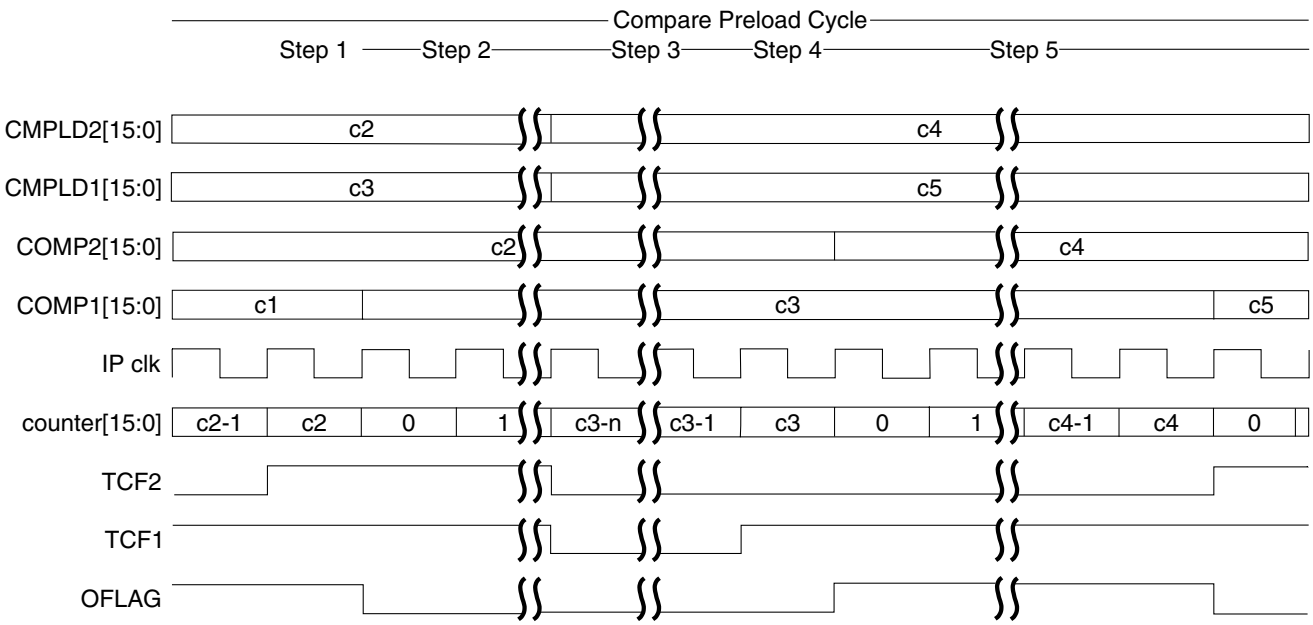
Interrupt Service Routines

To service the CSCTRL[TCF2] interrupts generated by the timer, the interrupt controller must be configured to enable the interrupts for the particular timer being used. Additionally the user will need to write an interrupt service routine to do at a minimum the following:

- Clear CSCTRL[TCF2] and CSCTRL[TCF1] flags.
- Calculate and write new values for both CMPLD1 and CMPLD2.

Timing

This figure contains the timing for using the compare preload feature. The compare preload cycle begins with a compare event on COMP2 causing CSCTRL[TCF2] to be asserted. COMP1 is loaded with the value in the CMPLD1 (c3) one IP bus clock later. In addition an interrupt is asserted by the timer and the interrupt service routine is executed during which both comparator load registers are updated with new values (c4 and c5). When CSCTRL[TCF1] is asserted, COMP2 is loaded with the value in CMPLD2 (c4). And on the subsequent CSCTRL[TCF2] event, COMP1 is loaded with the value in CMPLD1 (c5). The cycle starts over again as an interrupt is asserted and the interrupt service routine clears CSCTRL[TCF1] and CSCTRL[TCF2] and calculates new values for CMPLD1 and CMPLD2.



Step 1-- CNTR matches COMP2 value. CSCTRL[TCF2] is asserted and an interrupt request is generated.

Step 2-- One clock later, OFLAG toggles, CMPLD1 is copied to COMP1, LOAD is copied to CNTR, the counter starts counting.

Step 3-- The interrupt service routine clears CSCTRL[TCF1] and CSCTRL[TCF2] and the ISR loads CMPLD1 and CMPLD2 with the values for the next cycle. The counter continues counting until CNTR matches COMP1.

Step 4-- CSCTRL[TCF1] is asserted. One clock later, OFLAG toggles, CMPLD2 is copied to COMP2, LOAD is copied to CNTR and the counter starts counting.

Step 5--The counter continues counting until CNTR matches COMP2.

Figure 27-8. Compare Load Timing

Example: 27.6.5.14.1 Variable Frequency PWM Mode

```
//    (See Processor Expert PPG [Programmable Pulse Generator] bean.)
// This example starts with an 11 msec with a 31 msec cycle.
// Assuming the chip is operating at 60 MHz, the timer use IP_bus_clk/32 as its
//   clock source.
//
// Initial pulse period: 60e6/32 clocks/sec * 31 ms = 58125 total clocks in period
// Initial pulse width: 60e6/32 clocks/sec * 11 ms = 20625 clocks in pulse
//
//
// Once the initial values of COMP1/CMPLD1 and COMP2/CMPLD2 are set the pulse width
//   can be varied by load new values of CMPLD1 and CMPLD2 on each compare interrupt.
//    (See Usage of Compare Load Registers.)
//
void PPG1_Init(void)
{
  setReg(TMR0_LOAD,0);                /* Clear load register */
```

```
  setReg(TMR0_CNTR,0);                /* Clear counter */
  /* TMR0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=1,OPS=0,OEN=1 */
  setReg(TMR0_SCTRL,5);                /* Set Status and Control Register */
// Set compare preload operation and enable an interrupt on compare2 events.
  /* TMR0_CSCTRL: TCF2EN=1,TCF1EN=0,TCF2=0,TCF1=0,CL21=0,CL20=1,CL11=1,CL10=0 */
  setReg(TMR0_CSCTRL,0x86);          /* Set Comparator Status and Control Register */

  setReg(TMR0_COMP1,20625);            /* Set the pulse width of the off time */
  setReg(TMR0_CMPLD1,20625);          /* Set the pulse width of the off time */
  setReg(TMR0_COMP2,58125-20625);      /* Set the pulse width of the on time */
  setReg(TMR0_CMPLD2,58125-20625);    /* Set the pulse width of the on time */
  /* TMR0_CTRL: CM=1,PCS=0xD,SCS=0,ONCE=0,LENGTH=1,DIR=0,COINIT=0,OUTMODE=4 */
  setRegBits(TMR0_CTRL,0x3A24);        /* Set variable PWM mode and run counter */
}
```

27.7 Resets

27.7.1 General

The TMR module can be reset only by the RST_B signal. This forces all registers to their reset state and clears the OFLAG signal if it is asserted. The counter will be turned off until the settings in the control register are changed.

Table 27-1. Reset Summary

Reset	Priority	Source	Characteristics
RST_B	n/a	Hardware Reset	Full System Reset

27.8 Clocks

27.8.1 General

The timer only receives the IP bus clock .

27.9 Interrupts

27.9.1 General

The TMR module can generate 20 interrupts, Five for each of the four counters/channels.

Table 27-2. Interrupt Summary

Core Interrupt	Interrupt	Description
TMR Channel 0	TMR0_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 0
	TMR0_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 0
	TMR0_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 0
	TMR0_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 0
	TMR0_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 0
TMR Channel 1	TMR1_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 1
	TMR1_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 1
	TMR1_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 1
	TMR1_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 1
	TMR1_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 1
TMR Channel 2	TMR2_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 2
	TMR2_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 2
	TMR2_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 2
	TMR2_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 2
	TMR2_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 2
TMR Channel 3	TMR3_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 3
	TMR3_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 3
	TMR3_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 3
	TMR3_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 3
	TMR3_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 3

27.9.2 Description of Interrupt Operation

27.9.2.1 Timer Compare Interrupts

These interrupts are generated when a successful compare occurs between a counter and it's compare registers while SCTRL[TCFIE] is set. These interrupts are cleared by writing a zero to the appropriate SCTRL[TCF].

When a timer compare interrupt is set in TMR_SCTRL and the Compare Load registers are available, one of the following two interrupts will also be asserted.

27.9.2.1.1 Timer Compare 1 Interrupts (Available with Compare Load Feature)

These interrupts are generated when a successful compare occurs between a counter and its COMP1 register while CSCTRL[TCF1EN] is set. These interrupts are cleared by writing a zero to the appropriate CSCTRL[TCF1].

27.9.2.1.2 Timer Compare 2 Interrupts (Available with Compare Load Feature)

These interrupts are generated when a successful compare occurs between a counter and its COMP2 register while CSCTRL[TCF2EN] is set. These interrupts are cleared by writing a zero to the appropriate CSCTRL[TCF2].

27.9.2.2 Timer Overflow Interrupts

These interrupts are generated when a counter rolls over its maximum value while SCTRL[TOFIE] is set. These interrupts are cleared by writing zero to the appropriate SCTRL[TOF].

27.9.2.3 Timer Input Edge Interrupts

These interrupts are generated by a transition of the input signal (either positive or negative depending on IPS setting) while SCTRL[IEFIE] is set. These interrupts are cleared by writing a zero to the appropriate SCTRL[IEF].

27.10 DMA

The TMR module can generate twelve DMA requests: three for each of the four counters/channels. Refer to the following table.

Table 27-3. DMA Summary

DMA Request	DMA Enable	Description
Channels 0-3	DMA[IEFDE]	CAPT contains a value
	DMA[CMPLD1DE]	CMPLD1 needs an update
	DMA[CMPLD2DE]	CMPLD2 needs an update

Chapter 28
Periodic Interrupt Timer (PIT)

28.1 Chip-specific information for this module

28.1.1 PIT low power clocks

Each PIT module's CTRL[CLKSEL] bitfield selects among the options for that PIT's counter clock source. For each value of the bitfield, the following table correlates between the module-level and chip-level names of the clock options.

Table 28-1. PIT counter clock selection

PITx_CTRL[CLKSEL] value	Clock input	Clock source
00b	IP bus clock	IPS Bus Clock
01b	Alternate clock 1	Crystal Oscillator
10b	Alternate clock 2	8 MHz / 2 MHz IRC
11b	Alternate clock 3	200 KHz IRC (Fosc200KHz)

28.1.2 PIT master/slave selection

Either PIT0 or PIT1 can be the master of the other PIT module instance. Use the SIM's MISC0[PIT_MSTR] bit to select between the two master/slave configurations. The master PIT module instance generates the count_enable signal to synchronize both instances of the PIT.

28.2 Introduction

The programmable interval timer module (PIT) contains clock select logic, a 32-bit up counter, a modulo register, and a control register. The modulo and control registers are read/writable. The counter is read only.

The modulo register is loaded with a value to count to and the prescaler is set to determine the counting rate. When enabled, the counter counts up to the modulo value and set a flag (and an interrupt request if enabled), reset to 0x0, and resume counting.

28.2.1 Features

The PIT module design includes these distinctive features:

- 32-bit counter/timer.
- Programmable count modulo.
- Up to 4 selectable clock sources.
- Maximum count rate equal to clocking rate.
- Slave mode allows synchronization of multiple PIT count enables.

28.2.2 Modes of Operation

The PIT module design operates in only a single mode of operation: functional mode.

28.2.3 Block Diagram

The following figure shows the PIT block diagram.

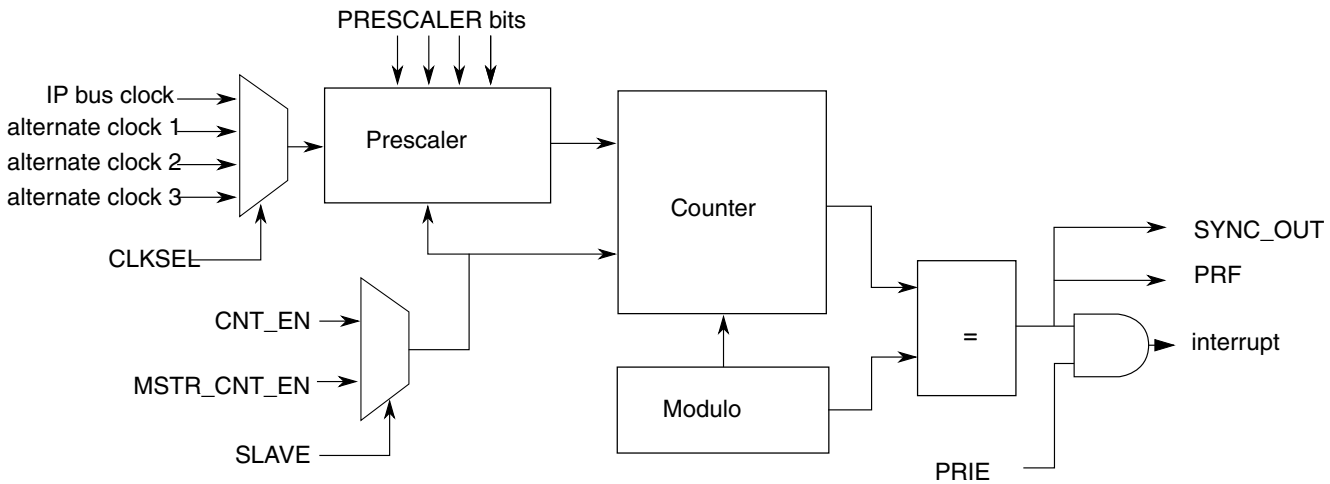


Figure 28-1. Programmable Interval Timer Block Diagram

28.3 Memory Map and Registers

The base address of the PIT module differs from chip to chip. The following descriptions identify the locations of memory mapped registers in relation to the base address.

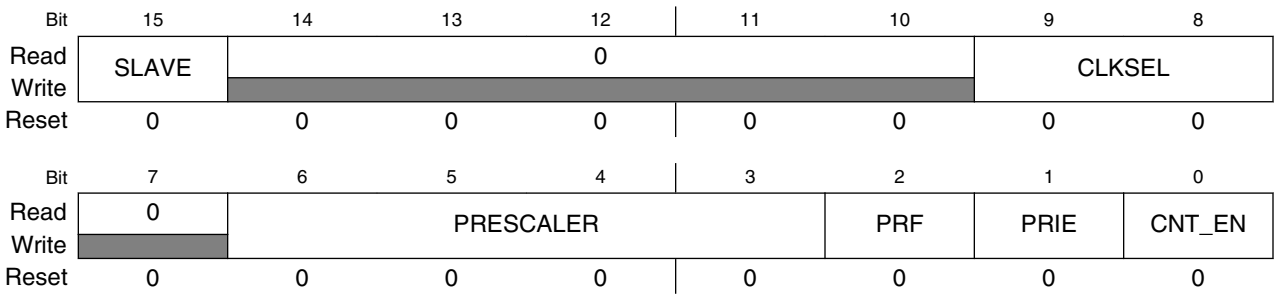
The address of a register is the sum of a base address and an address offset. The base address is defined at the DSC core level and the address offset is defined at the module level.

PIT memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E100	PIT Control Register (PIT0_CTRL)	16	R/W	0000h	28.3.1/862
E101	PIT Modulo Register Low Half Word (PIT0_MOD_L)	16	R/W	0000h	28.3.2/863
E102	PIT Modulo Register High Half Word (PIT0_MOD_H)	16	R/W	0000h	28.3.3/864
E103	PIT Counter low half word Register (PIT0_CNTR_L)	16	R	0000h	28.3.4/864
E104	PIT Counter high half word Register (PIT0_CNTR_H)	16	R	0000h	28.3.5/864
E110	PIT Control Register (PIT1_CTRL)	16	R/W	0000h	28.3.1/862
E111	PIT Modulo Register Low Half Word (PIT1_MOD_L)	16	R/W	0000h	28.3.2/863
E112	PIT Modulo Register High Half Word (PIT1_MOD_H)	16	R/W	0000h	28.3.3/864
E113	PIT Counter low half word Register (PIT1_CNTR_L)	16	R	0000h	28.3.4/864
E114	PIT Counter high half word Register (PIT1_CNTR_H)	16	R	0000h	28.3.5/864

28.3.1 PIT Control Register (PITx_CTRL)

Address: Base address + 0h offset



PITx_CTRL field descriptions

Field	Description
15 SLAVE	This field is used to place this PIT module in slave mode. This means that the CNT_EN field is ignored and instead this PIT uses the master count enable signal broadcast from the PIT0 module. This bit allows synchronization of the counts across multiple PIT modules. This bit is only useful in designs with multiple PIT modules. Setting this bit in the (master) PIT0 module has no effect as its own CNT_EN field is also the master count enable. 0 CNT_EN from this PIT is used to control operation (default). 1 CNT_EN from master PIT is used to control operation.
14–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–8 CLKSEL	This field is used to select the source of the clocking for the counter (PIT clock). This field should not be changed when CNT_EN is set. The default selection is the IPBus clock. 00 Selects IPBus clock 01 Selects alternate clock 1 10 Selects alternate clock 2 11 Selects alternate clock 3
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–3 PRESCALER	This field is used to select the prescaling of the selected clock to determine the counting rate of the PIT clock. 0000 Clock 0001 Clock divided by 2 0010 Clock divided by 4 0011 Clock divided by 8 0100 Clock divided by 16 0101 Clock divided by 32 0110 Clock divided by 64 0111 Clock divided by 128 1000 Clock divided by 256 1001 Clock divided by 512 1010 Clock divided by 1024 1011 Clock divided by 2048

Table continues on the next page...

PITx_CTRL field descriptions (continued)

Field	Description
	1100 Clock divided by 4096 1101 Clock divided by 8192 1110 Clock divided by 16384 1111 Clock divided by 32768
2 PRF	PIT Roll-Over Flag. This bit is set when the counter rolls over to 0x0000 after matching the value in the PIT compare register. This bit is cleared by reading the CTRL register with PRF set and then writing a zero to this bit position. Due to resynchronization timing for this method of clearing PRF, allow two IPBus clock periods before the cleared value of PRF is shown in the CTRL register. This bit can also be cleared by setting the CNT_EN bit to 0. Due to resynchronization timing for this method of clearing PRF, allow three PIT clock periods and two IPBus clock periods before the cleared PRF bit is shown in the CTRL register. Writing a one to the PRF bit position has no effect. 0 PIT counter has not reached the modulo value. (default) 1 PIT counter has reached the modulo value.
1 PRIE	PIT Roll-Over Interrupt Enable. This bit enables the PIT roll-over interrupt when the PRF bit becomes set. 0 PIT roll-over interrupt disabled (default). 1 PIT roll-over interrupt enabled.
0 CNT_EN	Count Enable This bit enables the PIT prescaler and counter. When this bit is clear, the counter remains at or returns to a 0x0000 value. The PRF bit is also reset when CNT_EN is clear. Due to resynchronization timing for this method of clearing PRF, allow three PIT clock periods and two IPBus clock periods before the cleared PRF bit is shown in the CTRL register. This field is ignored when the SLAVE bit is set and the count enable signal from the master PIT is used instead. 0 PIT counter reset (default). 1 PIT counter active.

28.3.2 PIT Modulo Register Low Half Word (PITx_MOD_L)

Address: Base address + 1h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	MODULO_VALUE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PITx_MOD_L field descriptions

Field	Description
MODULO_ VALUE	This read/write register stores the lower 16-bit modulo value for the PIT counter.

28.3.3 PIT Modulo Register High Half Word (PITx_MOD_H)

Address: Base address + 2h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	MODULO_VALUE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PITx_MOD_H field descriptions

Field	Description
MODULO_ VALUE	This read/write register stores the high 16-bit modulo value for the PIT counter.

28.3.4 PIT Counter low half word Register (PITx_CNTR_L)

Address: Base address + 3h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COUNTER_VALUE															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PITx_CNTR_L field descriptions

Field	Description
COUNTER_ VALUE	This read only register contains the low 16-bit current value of the PIT counter. Clearing CNT_EN resets the counter to 0x0000. When the PIT counter rolls over the modulo value, the PRF bit becomes set and the PIT counter resumes counting from 0x0000. If the selected counting rate is faster than the IPBus clock, then this count value cannot be read.

28.3.5 PIT Counter high half word Register (PITx_CNTR_H)

Address: Base address + 4h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COUNTER_VALUE															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PITx_CNTR_H field descriptions

Field	Description
COUNTER_ VALUE	This read only register contains the high 16-bit current value of the PIT counter. Clearing CNT_EN resets the counter to 0x0000. When the PIT counter rolls over the modulo value, the PRF bit becomes set and

PITx_CNTR_H field descriptions (continued)

Field	Description
	the PIT counter resumes counting from 0x0000. If the selected counting rate is faster than the IPBus clock, then this count value cannot be read.

28.4 Functional Description

The purpose of the PIT is to create a repeated interrupt request at a programmable time interval. The periodic rate is determined based on the peripheral clock rate, the prescaler value, and the modulo value as shown in the following equation:

```
interrupt rate = peripheral clock rate / ((2^prescaler) * modulo value)
```

When using the PIT, set the clock select, prescaler, and modulo values prior to setting CNT_EN. Changing these settings with CNT_EN set may cause unexpected operation.

The roll-over flag is set upon rolling over the modulo value back to 0x0. See the following figure for an example of PRF timing using a prescaler of 0x2 (IP bus clock divided by 4).

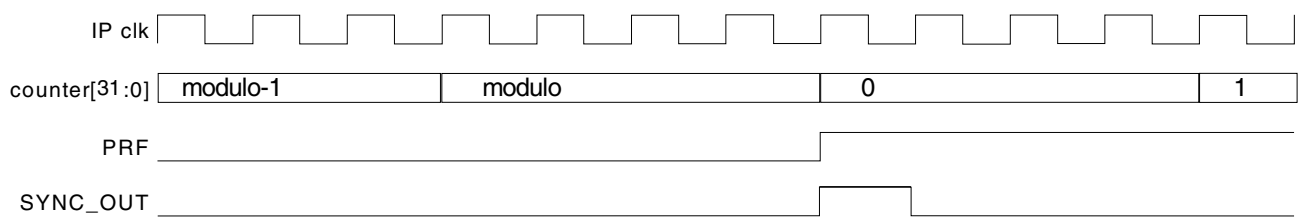


Figure 28-2. Example POF Timing

The roll-over flag can be cleared by writing a zero to the PRF bit position or by clearing CNT_EN. Due to resynchronization timing when clearing CNT_EN, allow three PIT clock periods and two IPBus clock periods before the cleared PRF bit is shown in the CTRL register. Due to resynchronization timing when writing a 0 to PRF, allow two IPBus clock periods before the cleared PRF bit is shown in the CTRL register.

28.4.1 Slave Mode

When using slave mode to synchronize several PIT modules only the CNT_EN signal is shared not the clocking for the counters. This means that each slave PIT must have their CLKSEL field, PRESCALER field, and PIT_MOD registers programmed prior to setting CNT_EN in the master PIT. The following figure shows the connection between the master PIT (PIT0) and the possible slave PITs (PIT1–PITn).

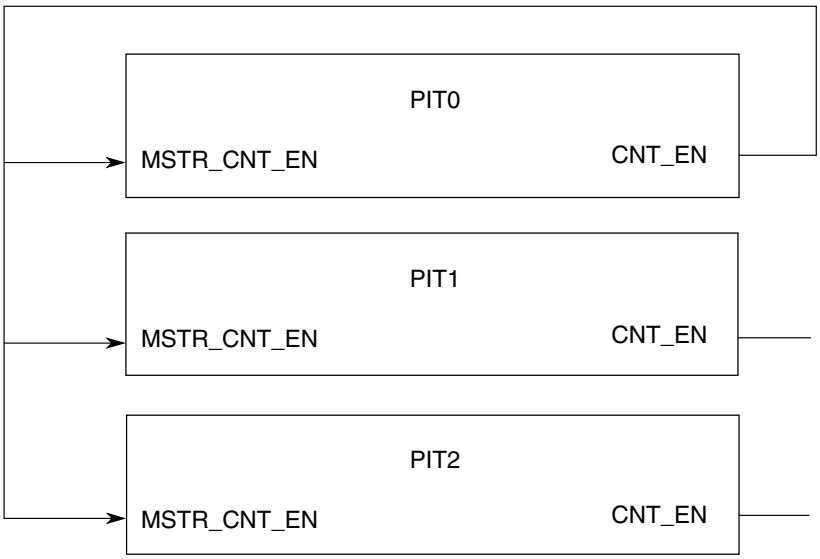


Figure 28-3. CNT_EN Connection Between Multiple PITs

28.4.2 Low Power Modes

This section describes the PIT low power modes.

28.4.2.1 Wait Mode

If the CNT_EN bit is set prior to entering wait mode, then the PIT continues to count and can wake the chip by asserting its interrupt upon reaching the modulo value.

28.4.2.2 Stop Mode

Stop mode operation depends on whether the system integration module (SIM) is set to allow the PIT to be clocked in stop mode. If not, the PIT counter does not operate during stop mode, but does retain its current settings. If CNT_EN is set, then the counter resumes counting upon exit of stop mode assuming the exit isn't caused by a reset. If the PIT does receive clocks while the chip is in stop mode, then operation continues normally.

28.4.2.3 Debug Mode

If the CNT_EN bit is set prior to the chip entering debug mode, then the PIT continues to count during debug mode.

28.5 Interrupts

The PIT module can generate a single interrupt when the counter reaches the modulo value.

28.6 Read process of the counter registers

Since the 32-bit counter registers are divided into lower register and higher register respectively, the read process is as follows:

- Read lower register first: if the first read is PIT_CNTR_L, then the present counter value has been latched. Then if the next read is PIT_CNTR_H, the total 32-bit counter value read process has been finished. If the next read is still PIT_CNTR_L, the new present counter value is latched, waiting for the according PIT_CNTR_H read to finish the complete count read.
- Read higher register first: if the first read is PIT_CNTR_H, then the present counter value has been latched. Then if the next read is PIT_CNTR_L, the total 32-bit couter value read process has been finished. If the next read is still PIT_CNTR_H, then a new present counter value is latched, waiting for the according PIT_CNTR_L read to finish the complete count read.

Chapter 29
Queued Serial Communications Interface (QSCI)

29.1 Introduction

The SCI allows asynchronous serial communications with peripheral devices.

29.1.1 Features

- Full-duplex or single-wire operation
- Standard mark/space non-return-to-zero (NRZ) format
- 16-bit integer and 3-bit fractional baud rate selection
- Programmable 8-bit or 9-bit data format
- Separately enabled transmitter and receiver
- Separate receiver and transmitter DSC core interrupt requests
- Programmable polarity for transmitter and receiver
- Two receiver wake-up methods: idle line or address mark
- Clockless receiver wake-up on active input edge
- Interrupt-driven operation with multiple flags:
    - Transmitter empty
    - Transmitter idle
    - Receiver full
    - Receiver overrun
    - Receiver idle

- Receiver input edge
- Noise error
- Framing error
- Parity error

- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection
- Supports Local Interconnect Network (LIN)

29.1.2 SCI Block Diagram

The following figure shows the SCI block diagram.

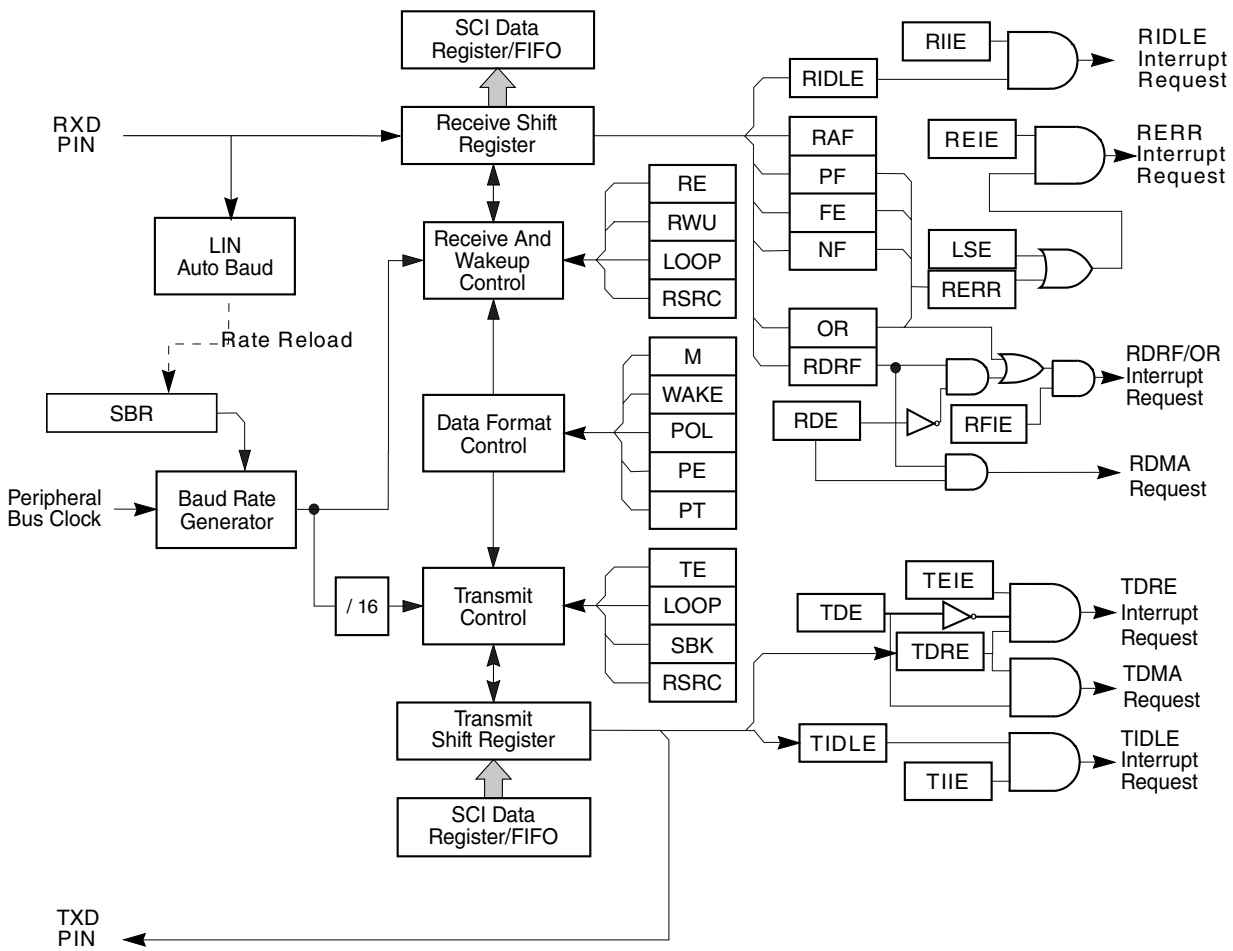


Figure 29-1. SCI Block Diagram with DMA

29.2 External Signal Descriptions

Table 29-1. Signal Properties

Name	I/O Type	Function	Reset State
TXD	O	Transmit Data Pin	1
RXD	I	Receive Data Pin	—

29.2.1 TXD —Transmit Data

The Transmit Data Pin (TXD) is the SCI transmitter pin. TXD is available for general purpose I/O when it is not configured for transmitter operation, such as when CTRL1[TE] = 0.

29.2.2 RXD —Receiver Data

The Receiver Data Pin (RXD) is the SCI receiver pin. RXD is available for general-purpose I/O when it is not configured for receiver operation, such as when CTRL1[RE] = 0.

29.3 Memory Map and Registers

QSCI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E080	QSCI Baud Rate Register (QSCI0_RATE)	16	R/W	0200h	29.3.1/872
E081	QSCI Control Register 1 (QSCI0_CTRL1)	16	R/W	0000h	29.3.2/872
E082	QSCI Control Register 2 (QSCI0_CTRL2)	16	R/W	0000h	29.3.3/875
E083	QSCI Status Register (QSCI0_STAT)	16	R/W	C000h	29.3.4/877
E084	QSCI Data Register (QSCI0_DATA)	16	R/W	0000h	29.3.5/880
E085	QSCI Control Register 3 (QSCI0_CTRL3)	16	R/W	0000h	29.3.6/881
E090	QSCI Baud Rate Register (QSCI1_RATE)	16	R/W	0200h	29.3.1/872
E091	QSCI Control Register 1 (QSCI1_CTRL1)	16	R/W	0000h	29.3.2/872
E092	QSCI Control Register 2 (QSCI1_CTRL2)	16	R/W	0000h	29.3.3/875
E093	QSCI Status Register (QSCI1_STAT)	16	R/W	C000h	29.3.4/877

Table continues on the next page...

QSCI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E094	QSCI Data Register (QSCI1_DATA)	16	R/W	0000h	29.3.5/880
E095	QSCI Control Register 3 (QSCI1_CTRL3)	16	R/W	0000h	29.3.6/881

29.3.1 QSCI Baud Rate Register (QSCIx_RATE)

Read: anytime

Write: anytime

Address: Base address + 0h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SBRL													FRAC_SBR		
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

QSCIx_RATE field descriptions

Field	Description
15–3 SBRL	Low order bits of SCI Baud Rate divider, which combine with the CTRL3[SBRH] field to form a value from 1 to 65535 Refer to the description of the FRAC_SBR bitfield.
FRAC_SBR	Fractional SCI Baud Rate divider, a value from 0 to 7 that is divided by 8 The SBRL, SBRH, and FRAC_SBR fields combine to form the divider to determine the baud rate of the SCI. The integer portion of the baud rate divider is represented by SBRL and SBRH, and the fractional portion is represented by FRAC_SBR. The FRAC_SBR field can only be used when the integer portion is greater than 1. Therefore, the value of the divider can be 1.000 or (with fractional values) in the range from 2.000 to 65535.875. The formula for calculating the baud rate is: `Baud rate = peripheral bus clock / (16 * ({SBRH, SBRL} + (FRAC_SBR / 8)))` **NOTE:** The baud rate generator is disabled until CTRL1[TE] or CTRL1[RE] is set for the first time after reset. The baud rate generator is disabled when RATE[SBRL], CTRL3[SBRH], and RATE[FRAC_SBR] equal 0. **NOTE:** If CTRL2[LINMODE] is set, the value of this register is automatically adjusted to match the data rate of the LIN master device. Reading this register yields the auto-baud value set.

29.3.2 QSCI Control Register 1 (QSCIx_CTRL1)

Read: anytime

Write: anytime

Address: Base address + 1h offset

Bit	15	14	13	12	11	10	9	8
Read Write	LOOP	SWAI	RSRC	M	WAKE	POL	PE	PT
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	TEIE	TIIE	RFIE	REIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0

QSCIx_CTRL1 field descriptions

Field	Description
15 LOOP	Loop Select This bit enables loop operation. In loop operation the RXD pin is disconnected from the SCI, and the transmitter output goes into the receiver input. Both the transmitter and the receiver must be enabled to use the internal loop function as opposed to single wire operation, which requires only one or the other to be enabled. The receiver input is determined by CTRL1[RSRC]. The transmitter output is controlled by CTRL1[TE]. If CTRL1[TE] is set and CTRL1[LOOP]=1, the transmitter output appears on the TXD pin. If CTRL1[TE] is clear and CTRL1[LOOP]=1, the TXD pin is high-impedance. 0 Normal operation, regardless of the value of RSRC 1 When RSRC = 0: Loop mode with internal TXD fed back to RXD 1 When RSRC = 1: Single-wire mode with TXD output fed back to RXD
14 SWAI	Stop in Wait Mode This bit disables the SCI in wait mode 0 SCI enabled in wait mode 1 SCI disabled in wait mode
13 RSRC	Receiver Source When CTRL1[LOOP]=1, CTRL1[RSRC] determines the internal feedback path for the receiver. 0 Receiver input internally connected to transmitter output 1 Receiver input connected to TXD pin
12 M	Data Format Mode This bit determines whether data characters are eight or nine bits long. 0 One start bit, eight data bits, one stop bit 1 One start bit, nine data bits, one stop bit
11 WAKE	Wake-up Condition This bit determines which condition wakes the SCI: a one (address mark) in the most significant bit position of a received data character or an idle condition on the RXD pin. 0 Idle line wake-up 1 Address mark wake-up
10 POL	Polarity This bit determines whether to invert the data as it goes from the transmitter to the TXD pin and from the RXD pin to the receiver. All bits (start, data, and stop) are inverted as they leave the transmit shift register and before they enter the receive shift register.

Table continues on the next page...

QSCIx_CTRL1 field descriptions (continued)

Field	Description
	NOTE: It is recommended that CTRL1[POL] be toggled only when CTRL1[TE]=0 and CTRL1[RE]=0. 0 Don't invert transmit and receive data bits (normal mode) 1 Invert transmit and receive data bits (inverted mode)
9 PE	Parity Enable This bit enables the parity function. When enabled, the parity function replaces the most significant bit of the data character with a parity bit. 0 Parity function disabled 1 Parity function enabled
8 PT	Parity Type This bit determines whether the SCI generates and checks for even parity or odd parity of the data bits. With even parity, an even number of ones clears the parity bit and an odd number of ones sets the parity bit. With odd parity, an odd number of ones clears the parity bit and an even number of ones sets the parity bit. 0 Even parity 1 Odd parity
7 TEIE	Transmitter Empty Interrupt Enable This bit enables the transmit data register empty flag, STAT[TDRE], to generate interrupt requests. 0 STAT[TDRE] interrupt requests disabled 1 STAT[TDRE] interrupt requests enabled
6 TIIE	Transmitter Idle Interrupt Enable This bit enables the transmitter idle flag, STAT[TIDLE], to generate interrupt requests. 0 STAT[TIDLE] interrupt requests disabled 1 STAT[TIDLE] interrupt requests enabled
5 RFIE	Receiver Full Interrupt Enable This bit enables the receive data register full flag, STAT[RDRF], or the overrun flag, STAT[OR], to generate interrupt requests. 0 STAT[RDRF] and STAT[OR] interrupt requests disabled 1 STAT[RDRF] and STAT[OR] interrupt requests enabled
4 REIE	Receive Error Interrupt Enable This bit enables the receive error flags (STAT[NF], STAT[PF], STAT[FE], STAT[LSE], and STAT[OR]) to generate interrupt requests. 0 Error interrupt requests disabled 1 Error interrupt requests enabled
3 TE	Transmitter Enable This bit enables the SCI transmitter and configures the TXD pin as the SCI transmitter output. CTRL1[TE] can be used to queue an idle preamble. 0 Transmitter disabled 1 Transmitter enabled

Table continues on the next page...

QSCIx_CTRL1 field descriptions (continued)

Field	Description
2 RE	Receiver Enable This bit enables the SCI receiver. 0 Receiver disabled 1 Receiver enabled
1 RWU	Receiver Wake-up This bit enables the wake-up function and inhibits further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing CTRL1[RWU]. 0 Normal operation 1 Standby state
0 SBK	Send Break Toggling this bit sends one break character (10 or 11 zeroes). As long as this bit is set, the transmitter sends zeroes. 0 No break characters 1 Transmit break characters

29.3.3 QSCI Control Register 2 (QSCIx_CTRL2)

Read: anytime

Write: anytime

Address: Base address + 2h offset

Bit	15	14	13	12	11	10	9	8
Read	TFCNT			TFWM		RFCNT		
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	RFWM		FIFO_EN	RIEIE	LINMODE	RIIE	TDE	RDE
Write								
Reset	0	0	0	0	0	0	0	0

QSCIx_CTRL2 field descriptions

Field	Description
15–13 TFCNT	Transmit FIFO Count These read only bits show how many words are used in the TX FIFO. Writes to DATA cause CTRL2[TFCNT] to increment. As words are pulled for transmission, CTRL2[TFCNT] decrements. Attempts to write new data to DATA are ignored when CTRL2[TFCNT] indicates the FIFO is full (4 words). 000 0 words in Tx FIFO

Table continues on the next page...

QSCIx_CTRL2 field descriptions (continued)

Field	Description
	001 1 word in Tx FIFO 010 2 words in Tx FIFO 011 3 words in Tx FIFO 100 4 words in Tx FIFO 101 Reserved 110 Reserved 111 Reserved
12–11 TFWM	Transmit FIFO Empty Water Mark These bits set the TX FIFO word count level at which STAT[TDRE] is set. If CTRL2[FIFO_EN] is clear (FIFO disabled), then this field is forced to 00 and STAT[TDRE] is set when there is no word in the transmit buffer. 00 TDRE is set when 0 words are in the FIFO 01 TDRE is set when 1 or fewer words are in the FIFO 10 TDRE is set when 2 or fewer words are in the FIFO 11 TDRE is set when 3 or fewer words are in the FIFO
10–8 RFCNT	Receive FIFO Count These read only bits show how many words are used in the RX FIFO. As words are received, CTRL2[RFCNT] is incremented. As words are read from DATA the value of CTRL2[RFCNT] decrements. There is one word time to read DATA between when STAT[RDRF] is set (interrupt asserted), due to the RX FIFO being full, and when an overflow condition is flagged. 000 0 words in RX FIFO 001 1 word in RX FIFO 010 2 words in RX FIFO 011 3 words in RX FIFO 100 4 words in RX FIFO 101 Reserved 110 Reserved 111 Reserved
7–6 RFWM	Receive FIFO Full Water Mark These bits set the RX FIFO word count level at which STAT[RDRF] is set. If CTRL2[FIFO_EN] is clear (FIFO disabled), then this field is forced to 00 and STAT[RDRF] is set if there is a word in the receive buffer. 00 RDRF is set when at least 1 word is in the FIFO 01 RDRF is set when at least 2 words are in the FIFO 10 RDRF is set when at least 3 words are in the FIFO 11 RDRF is set when at least 4 words are in the FIFO
5 FIFO_EN	FIFO Enable This read/write bit enables the 4-word-deep TX and RX FIFOs. Change this bit only when the SCI is idle. 0 FIFOs are disabled. 1 FIFOs are enabled.
4 RIEIE	Receiver Input Edge Interrupt Enable This bit is used to enable the receiver input active edge interrupt request with the STAT[RIEF] bit.

Table continues on the next page...

QSCIx_CTRL2 field descriptions (continued)

Field	Description
	0 Receiver input edge interrupt request disabled. 1 Receiver input edge interrupt request enabled.
3 LINMODE	Enable LIN Slave Mode This bit should be used only in Local Interconnect Network (LIN) applications. **NOTE:** During initialization, the RATE register should be loaded to a value that is within 14% of the actual master data rate; otherwise, 0x00 data might be misinterpreted as a break. **NOTE:** If the first character following a break is not the LIN sync character (0x55), the RATE register is not adjusted and STAT[LSE] is set. 0 The LIN auto baud feature is disabled and the RATE register maintains whatever value the processor writes to it. 1 Enable LIN slave functionality. This includes a search for the break character followed by a sync character (0x55) from the master LIN device. When the break is detected (11 consecutive samples of zero), the subsequent sync character is used to measure the baud rate of the transmitting master, and the RATE register is automatically reloaded with the value needed to "match" that baud rate.
2 RIIE	Receiver Idle Interrupt Enable This bit is used to let the processor know a message has completed when using DMA in a wake-up mode of operation. The receiver is configured normally until a message is detected. The processor then determines if the message is for it. If so, DMA is enabled for the max message size, and the STAT[RIDLE] interrupt is enabled to tell the core when the message has completed. The receive DMA operation would be halted by the interrupt service routine at this time.
1 TDE	Transmitter DMA Enable This bit is used to enable DMA mode transfer of data to the DATA register. For transmit DMA operation this bit must be set and a DMA channel must be enabled to use the request. 0 Transmit DMA disabled 1 Transmit DMA enabled
0 RDE	Receiver DMA Enable This bit is used to enable DMA mode transfer of data from the DATA register. For receive DMA operation this bit must be set and a DMA channel must be enabled to utilize the request. 0 Receive DMA disabled 1 Receive DMA enabled

29.3.4 QSCI Status Register (QSCIx_STAT)

Read: anytime

Write: this register is writable only for clearing some flags

Address: Base address + 3h offset

Bit	15	14	13	12	11	10	9	8
Read	TDRE	TIDLE	RDRF	RIDLE	OR	NF	FE	PF
Write								
Reset	1	1	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0			RIEF	LSE	TDMA	RDMA	RAF
Write				w1c				
Reset	0	0	0	0	0	0	0	0

QSCIx_STAT field descriptions

Field	Description
15 TDRE	Transmit Data Register Empty Flag This bit is set when the TX FIFO word count (CTRL2[TFCNT]) falls to the watermark level (CTRL2[TXWM]). Clear TDRE by reading STAT with TDRE set and then writing to the SCI data register until CTRL2[TFCNT] is above CTRL2[TFWM]. If CTRL2[FIFO_EN] or CTRL2[TDE] is set, then you can clear TDRE by writing to the SCI data register without first reading STAT with TDRE set. 0 TX FIFO word count is above watermark 1 TX FIFO word count is at or below watermark
14 TIDLE	Transmitter Idle Flag This bit is set when the TX FIFO is empty and no data, preamble, or break character is being transmitted. When TIDLE is set, the TXD pin becomes idle (1). Clear TIDLE by reading STAT with TIDLE set and then writing to the SCI data register. TIDLE is not generated when a data character, a preamble, or a break is queued and ready to be sent. 0 Transmission in progress 1 No transmission in progress
13 RDRF	Receive Data Register Full Flag This bit is set when the RX FIFO word count (CTRL2[RFCNT]) rises above the watermark (CTRL2[RXWM]). Clear RDRF by reading STAT with RDRF set and then reading the SCI data register until CTRL2[RFCNT] is no longer above CTRL2[RFWM]. If CTRL2[FIFO_EN] or CTRL2[RDE] is set, then you can clear RDRF by reading the SCI data register without first reading STAT with RDRF set. **NOTE:** When you are using the CodeWarrior debugger, RDRF may be erased when a breakpoint is reached. If a memory window that includes the SCI registers is open when a breakpoint is reached, these memory addresses are read to update the memory window. If RDRF is set at this time, these reads satisfy the requirements for clearing RDRF because the status register is read with RDRF set and then the data register is read, which causes RDRF to clear. 0 RX FIFO word count is at or below watermark 1 RX FIFO word count is above watermark
12 RIDLE	Receiver Idle Line Flag This bit is set when 10 consecutive ones (if CTRL1[M]=0) or 11 consecutive ones (if CTRL1[M]=1) appear on the receiver input. The RIDLE flag is cleared by reading STAT with RIDLE set and then reading the SCI data register. Once RIDLE is cleared, a valid frame must be received before an idle condition can set RIDLE.

Table continues on the next page...

QSCIx_STAT field descriptions (continued)

Field	Description
	When the receiver wake-up bit (CTRL1[RWU]) is 0, the user can poll RIDLE to detect when the receiver is idle. **NOTE:** When the receiver wake-up bit (CTRL1[RWU]) is set to 1, an idle line condition does not set RIDLE. 0 Receiver input is either active now or has never become active since RIDLE was last cleared 1 Receiver input has become idle (after receiving a valid frame)
11 OR	Overrun Flag This bit is set when software fails to read the SCI data register when the RX FIFO is full before the receive shift register receives the next frame. The data in the shift register is lost, but the data already in the SCI data register/FIFO is not affected. Clear OR by reading STAT with OR set and then writing the SCI status register with any value. **NOTE:** When the Overrun Flag is set and remains set, other flags cannot become set. The receive FIFO is full and cannot receive any more words, so the arriving data is ignored and cannot set any other flags. 0 No overrun 1 Overrun
10 NF	Noise Flag This bit is set when the SCI detects noise on the receiver input. NF is set during the same cycle as RDRF but is not set in the case of an overrun. Clear NF by reading STAT and then writing the SCI status register with any value. 0 No noise 1 Noise
9 FE	Framing Error Flag This bit is set when a 0 is accepted as the stop bit. FE is set during the same cycle as RDRF but is not set in the case of an overrun. Clear FE by reading STAT with FE set and then writing the SCI status register with any value. 0 No framing error 1 Framing error
8 PF	Parity Error Flag This bit is set when the parity enable bit (CTRL1[PE]) is set and the parity of the received data does not match its parity bit. Clear PF by reading STAT and then writing the SCI status register with any value. 0 No parity error 1 Parity error
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 RIEF	Receiver Input Edge Flag This flag is set when an active edge is seen on the RXD input pin. An active edge is defined as a 1 to 0 transition when CTRL[POL] is 0 or as a 0 to 1 transition when CTRL[POL] is 1. This flag can be used in stop mode (clocks turned off) to create an interrupt to wake the CPU when the leading edge of a start bit is detected. As to whether the SOC can start up clocking fast enough for the SCI to properly detect the start bit and receive the data word is a function of SOC architecture and SCI baud rate. Clear RIEF by writing a 1 to this bit position.

Table continues on the next page...

QSCIx_STAT field descriptions (continued)

Field	Description
	0 No active edge on the receive pin has occured. 1 An active edge on the receive pin has occured.
3 LSE	LIN Sync Error This bit is active only when CTRL2[LINMODE] is set. When LSE is set, a Receive Error interrupt occurs if CTRL1[REIE] is set. LSE is set when a LIN sync search detects a non-sync character (anything other than 0x55). When set, this bit indicates either that a protocol error was detected from the Master LIN device or there is a gross mismatch in data rates. This bit is cleared by reading STAT with LSE set and then writing the SCI status register with any value. 0 No error occurred since CTRL2[LINMODE] was enabled or the bit was last cleared 1 A sync error prevented loading of the RATE register with a revised value after the break was detected.
2 TDMA	Transmit DMA Request When set, this bit indicates that the SCI is currently requesting a DMA data transfer for transmit data. This bit is cleared when the DMA channel writes enough data to the DATA register to raise CTRL2[TFCNT] to its maximum value. 0 Either CTRL2[TDE] is cleared or CTRL2[TDE] is set and CTRL2[TFCNT] is at its maximum value. 1 CTRL2[TDE] is set and CTRL2[TFCNT] is currently below its maximum value.
1 RDMA	Receive DMA Request When set, this bit indicates that the SCI is currently requesting a DMA data transfer for received data. This bit is cleared when the DMA channel reads enough data from the DATA register to lower CTRL2[RFCNT] to 0. 0 Either CTRL2[RDE] is cleared or CTRL2[RDE] is set and CTRL2[RFCNT] is 0. 1 CTRL2[RDE] is set and CTRL2[RFCNT] is currently above 0.
0 RAF	Receiver Active Flag This bit is set when the receiver detects a 0 during the RT1 time period of the start bit search. RAF is cleared when the receiver detects false start bits (usually from noise or baud rate mismatch) or when the receiver detects a preamble. 0 No reception in progress 1 Reception in progress

29.3.5 QSCI Data Register (QSCIx_DATA)

Read: anytime. Reading accesses the SCI receive data register.

Write: anytime. Writing accesses the SCI transmit data register.

Address: Base address + 4h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								RECEIVE_TRANSMIT_DATA							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QSCIx_DATA field descriptions

Field	Description
15–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RECEIVE_ TRANSMIT_ DATA	RECEIVE_DATA: Received data (when reading this register) TRANSMIT_DATA: Data to be transmitted (when writing this register) If STAT[TDRE] is set, writes to the transmit data field are ignored unless STAT has been read with STAT[TDRE] set. However, when CTRL2[FIFO_EN] or CTRL2[TDE] are set, you can write to the SCI data register without first reading STAT with STAT[TDRE] set. If STAT[RDRF] is set, reads from the receive data field are ignored unless STAT has been read with STAT[RDRF] set. However, when CTRL2[FIFO_EN] or CTRL2[RDE] are set, you can read from the SCI data register without first reading STAT with STAT[RDRF] set. This register is a 4-deep FIFO. **NOTE:** When the data format is configured for 8-bit data, bits 7 to 0 contain the data.

29.3.6 QSCI Control Register 3 (QSCIx_CTRL3)

Read: anytime.

Write: anytime.

Address: Base address + 5h offset

Bit	15	14	13	12	11	10	9	8
Read	SBRH				0			
Write								
Reset	0	0	0	0	0	0	0	0

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

QSCIx_CTRL3 field descriptions

Field	Description
15–13 SBRH	High order bits of SCI Baud Rate divider, which combine with the RATE[SBRL] field to form a value from 1 to 65535 Refer to the description of the RATE[FRAC_SBR] bitfield.
12–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 SHEN	Stop mode entry hold off Setting this bit allows the SCI to hold off chip level stop mode entry while the transmitter is not idle or while the transmit data register is not empty. It will also hold off stop mode entry while the receiver is not idle. Clearing this bit means that stop mode entry is not delyed and may occur while the SCI is actively transmitting/receiving.

Table continues on the next page...

QSCIx_CTRL3 field descriptions (continued)

Field	Description
	Under no cases will this bit affect wake up from stop mode once stop mode has been entered. **NOTE:** Not all chips support stop mode hold off. 0 Stop mode hold off is disabled. 1 Stop mode holdoff is enabled.

29.4 Functional Description

The SCI block diagram shows the structure of the SCI module. The SCI allows full duplex, asynchronous, NRZ serial communication between the DSP and remote devices, including other DSPs. The SCI transmitter and receiver operate independently, although they use the same baud rate generator. The DSC core monitors the status of the SCI, writes the data to be transmitted, and processes received data.

When initializing the SCI, be sure to set the proper peripheral enable bits in the GPIO as well as any pullup enables.

29.4.1 Data Frame Format

The SCI uses the standard NRZ mark/space data frame format illustrated in the following figure.

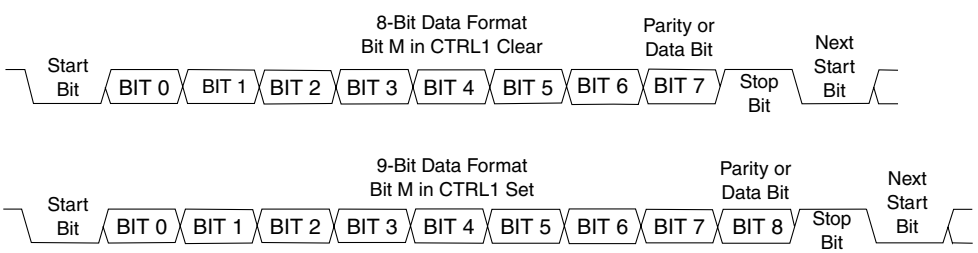


Figure 29-2. SCI Data Frame Formats

Each data character is contained in a frame that includes a start bit, eight or nine data bits, and a stop bit. Clearing CTRL1[M] configures the SCI for 8-bit data characters. A frame with eight data bits has a total of 10 bits, with formats as shown in the following table.

Table 29-2. Example 8-Bit Data Frame Formats

Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
1	8	0	0	1

Table continues on the next page...

Table 29-2. Example 8-Bit Data Frame Formats (continued)

Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
1	7	0	1	1
1	7	1[1]	0	1

1. The address bit identifies the frame as an address character.

Setting CTRL1[M] configures the SCI for 9-bit data characters. A frame with nine data bits has a total of 11 bits with formats as shown in the following table.

Table 29-3. Example 9-Bit Data Frame Formats

Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
1	9	0	0	1
1	8	0	0	2
1	8	0	1	1
1	8	1[1]	0	1

1. The address bit identifies the frame as an address character.

29.4.2 Baud-Rate Generation

A 16-bit modulus counter in the baud-rate generator derives the baud rate for both the receiver and the transmitter. The value written to the RATE[SBRL], CTRL3[SBRH], and RATE[FRAC_SBR] bits determines the peripheral bus clock divisor. The baud rate clock is synchronized with the IP Bus clock and drives the receiver. The baud rate clock divided by 16 drives the transmitter. 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 peripheral bus clock may not give the exact target frequency.
- Synchronization with the bus clock can cause phase shift.

The following table lists some examples of achieving target baud rates with a peripheral bus clock frequency of 60 MHz.

Table 29-4. Example Baud Rates (Peripheral Bus Clock = 60 MHz)

SBR Bits	Receiver Clock (Hz)	Transmitter Clock (Hz)	Target Baud Rate	Error (%)
32.5	1,846,154	115,385	115,200	0.16
65.125	921,305	57,582	57,600	-0.03

Table continues on the next page...

Table 29-4. Example Baud Rates (Peripheral Bus Clock = 60 MHz) (continued)

SBR Bits	Receiver Clock (Hz)	Transmitter Clock (Hz)	Target Baud Rate	Error (%)
97.625	614,597	38,412	38,400	0.03
195.25	307,298	19,206	19,200	0.03
390.625	153,600	9,600	9600	0.00
781.25	76,800	4,800	4800	0.00
1562.5	38,400	2,400	2400	0.00
3125	19,200	1,200.0	1200	0.00
6250	9,600	600.0	600	0.00

The following table lists some examples of achieving target baud rates with a peripheral bus clock frequency of 32 MHz.

Table 29-5. Example Baud Rates (Peripheral Bus Clock = 32 MHz)

SBR Bits	Receiver Clock (Hz)	Transmitter Clock (Hz)	Target Baud Rate	Error (%)
17.375	1,841,727	115,108	115,200	-0.08
34.75	920,863	57,554	57,600	-0.08
52.125	613,909	38,369	38,400	-0.08
104.125	307,323	19,208	19,200	0.04
208.375	153,569	9,598	9,600	-0.02
416.625	76,808	4,800	4,800	0.01
833.375	38,398	2,400	2,400	-0.01
1666.625	19,200	1,200	1,200	0.00
3333.375	9,600	600	600	0.00

Note

Maximum baud rate is peripheral bus clock rate divided by 16. System overhead may preclude processing the data at this speed.

29.4.3 Transmitter

The following figure is a block diagram of the transmitter functions.

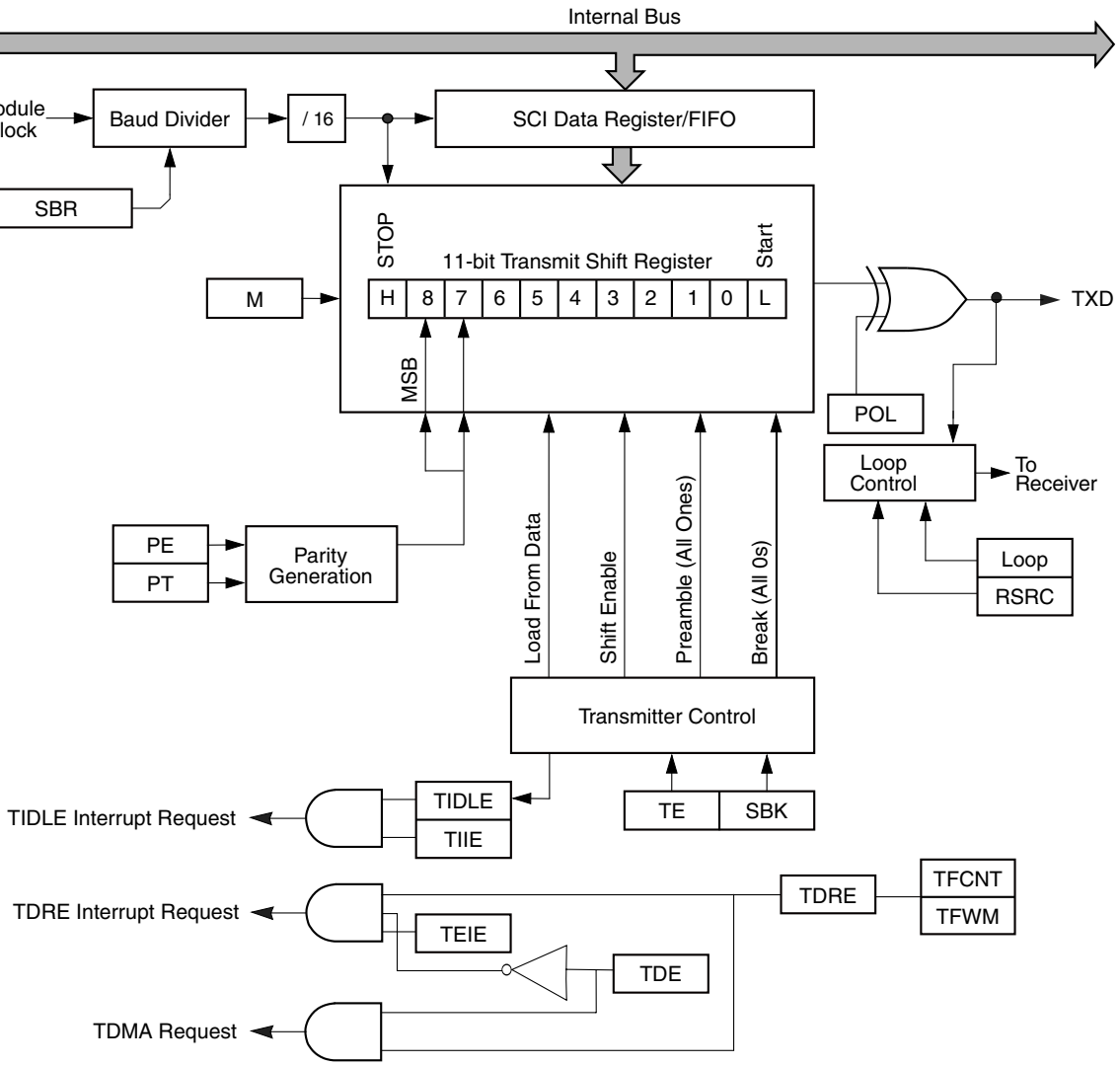


Figure 29-3. SCI Transmitter Block Diagram with DMA

29.4.3.1 Character Length

The SCI transmitter can accommodate either 8-bit or 9-bit data characters. The state of CTRL1[M] determines the length of data characters.

29.4.3.2 Character Transmission

During an SCI transmission, the transmit shift register shifts a frame out to the TXD pin. The SCI data register is the write-only buffer between the internal data bus and the transmit shift register.

To initiate an SCI transmission:

1. Enable the transmitter by writing a logic 1 to the transmitter enable bit (CTRL1[TE]).
2. Clear the transmit data register empty flag, STAT[TDRE], by first reading the SCI status register (STAT) and then writing to the SCI data register (DATA).
3. Repeat step 2 for each subsequent transmission.

Writing CTRL1[TE] bit from 0 to a 1 automatically loads the transmit shift register with a preamble of 10 ones (if CTRL1[M] = 0) or 11 ones (if CTRL1[M] = 1). After the preamble shifts out, control logic automatically transfers the data from the SCI data register into the transmit shift register. A logic zero start bit automatically goes into the least significant bit position of the transmit shift register. A logic one stop bit goes into the most significant bit (MSB) position of the frame.

Hardware supports odd or even parity. When parity is enabled, the MSB of the data character is replaced by the parity bit.

The transmit data register empty flag, STAT[TDRE], becomes set when the SCI data register transfers a character to the transmit shift register. STAT[TDRE] indicates that the SCI data register can accept new data from the internal data bus. If the transmitter empty interrupt enable bit, CTRL1[TEIE], is also set, STAT[TDRE] generates a transmitter interrupt request.

When the transmit shift register is not transmitting a frame and CTRL1[TE]=1, the TXD pin goes to the idle condition, logic one. If at any time software clears CTRL1[TE], the transmitter relinquishes control of the port I/O pin upon completion of the current transmission, causing the TXD pin to go to a high z state.

If software clears CTRL1[TE] while a transmission is in progress (STAT[TIDLE] = 0), the frame in the transmit shift register continues to shift out. Then transmission stops even if there is data pending in the SCI data register. To avoid accidentally cutting off the last frame in a message, always wait for STAT[TDRE] to go high after the last frame before clearing CTRL1[TE].

To separate messages with preambles with minimum idle line time, use this sequence between messages:

1. Write the last character of the first message to the DATA register.
2. Wait for STAT[TDRE] to go high while CTRL2[TFWM] = 00, indicating the transfer of the last frame to the transmit shift register.
3. Queue a preamble by clearing and then setting CTRL1[TE].
4. Write the first character of the second message to the DATA register.

29.4.3.3 Break Characters

Writing a logic one to the send break bit, CTRL1[SBK], loads the transmit shift register with a break character. A break character contains all logic zeroes and has no start, stop, or parity bit. Break character length depends on CTRL1[M]. As long as CTRL1[SBK] is at logic one, transmitter logic continuously loads break characters into the transmit shift register. After software clears CTRL1[SBK], the shift register finishes transmitting the last break character and then transmits at least one logic one. The automatic logic one at the end of the last break character guarantees the recognition of the start bit of the next frame.

In order to send an 11 bit break character, set the CTRL1[M] bit, set and then clear CTRL1[SBK], then poll STAT[TIDLE]. Once STAT[TIDLE] is asserted the CTRL1[M] bit can be cleared in order for future data words to be 8 bits long.

The SCI recognizes a break character when a start bit is followed by eight or nine logic zero data bits and a logic zero where the stop bit should be. Receiving a break character has these effects on SCI registers:

- Sets the framing error flag, STAT[FE]
- Sets the receive data register full flag, STAT[RDRF], if CTRL2[RFWM] = 00
- Clears the SCI data register
- May set the overrun flag (STAT[OR]), noise flag (STAT[NF]), parity error flag (STAT[PF]), or receiver active flag (STAT[RAF])

29.4.3.4 Preambles

A preamble contains all logic ones and has no start, stop, or parity bit. Preamble length depends on CTRL1[M]. The preamble is a synchronizing mechanism that begins the first transmission initiated after writing CTRL1[TE] from 0 to 1.

If CTRL1[TE] is cleared during a transmission, the TXD pin becomes idle after completion of the transmission in progress. Clearing and then setting CTRL1[TE] during a transmission queues a preamble to be sent after the frame currently being transmitted.

Note

Toggle CTRL1[TE] for a queued preamble when STAT[TDRE] becomes set and immediately before writing the next character to the DATA register.

When queueing a preamble, return CTRL1[TE] to logic one before the stop bit of the current frame shifts out to the TXD pin. Setting CTRL1[TE] after the stop bit appears on TXD causes data previously written to the SCI data register to be lost.

29.4.4 Receiver

The following figure is the block diagram of the SCI receiver.

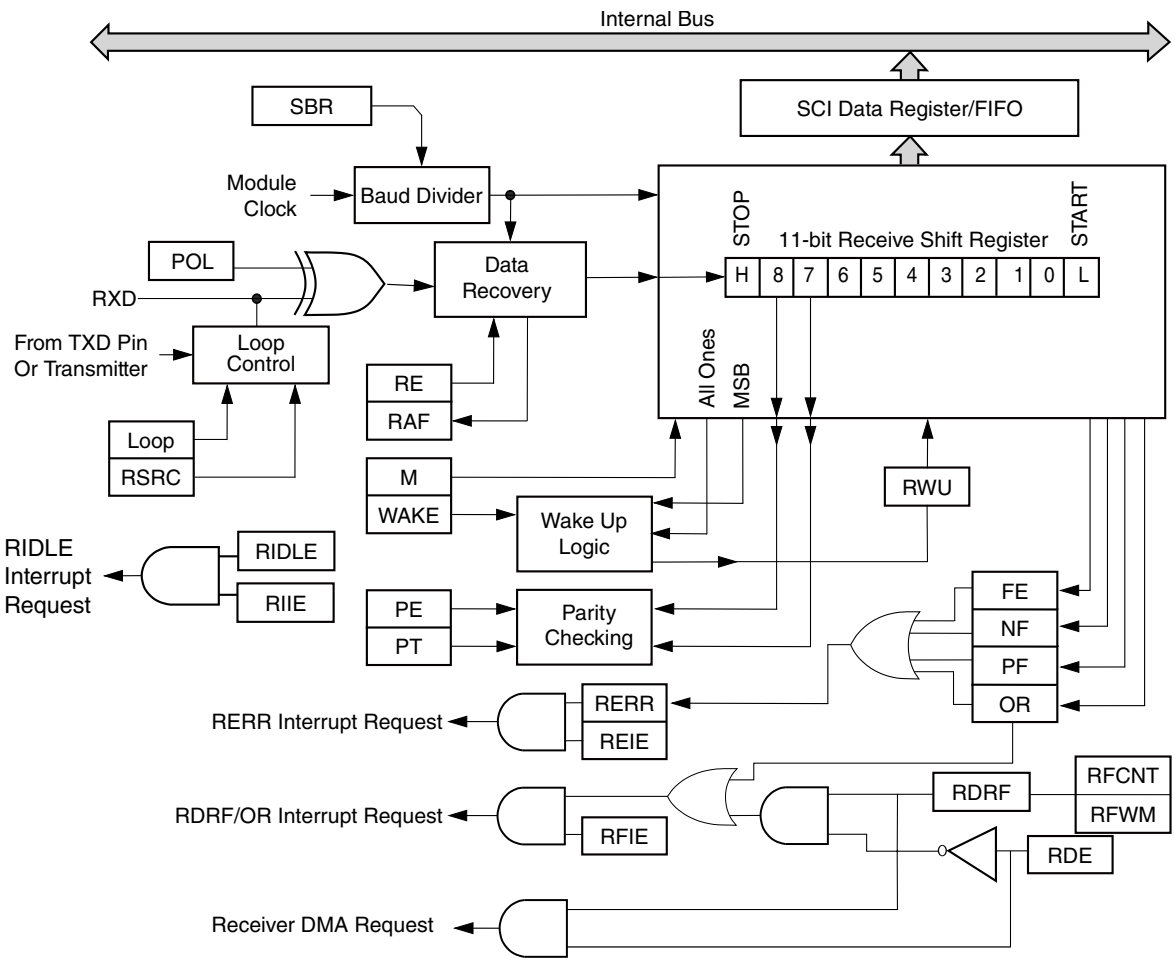


Figure 29-4. SCI Receiver Block Diagram with DMA

29.4.4.1 Character Length

The SCI receiver can accommodate either 8-bit or 9-bit data characters. The state of CTRL1[M] determines the length of data characters.

29.4.4.2 Character Reception

During an SCI reception, the receive shift register shifts a frame in from the RXD pin. The SCI data register/FIFO is the read-only buffer between the internal data bus and the receive shift register.

After a complete frame shifts into the receive shift register, the data portion of the frame along with the STAT[FE], STAT[NF], STAT[PF], and STAT[LSE] status flags transfer to the SCI data register. The receive data register full flag, STAT[RDRF], becomes set when the RX FIFO word count is above the watermark, indicating that a received character can be read. When the FIFO is enabled, there can be received data words to be read even if STAT[RDRF] is not set. If the receive interrupt enable bit, CTRL1[RFIE], is also set, STAT[RDRF] generates a Receiver Full interrupt request.

The STAT[FE], STAT[NF], STAT[PF], and STAT[LSE] flags are associated with the current character to be read from the receive data register/FIFO. When responding to a receiver error interrupt, read both the DATA register and the STAT register, and then write the STAT register to clear the error flags. When responding to a receiver full interrupt, it is necessary to read only the DATA register.

29.4.4.3 Data Sampling

The receiver samples the RXD pin 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 (shown in the following figure) is resynchronized:

- After every start bit
- After the receiver detects a data bit change from logic 1 to logic 0 (after most data bit samples at RT8, RT9, and RT10 return a valid logic 1 and most of the next RT8, RT9, and RT10 samples return a valid logic 0)

To locate the start bit, data recovery logic does an asynchronous search for a logic 0 preceded by three logic ones. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

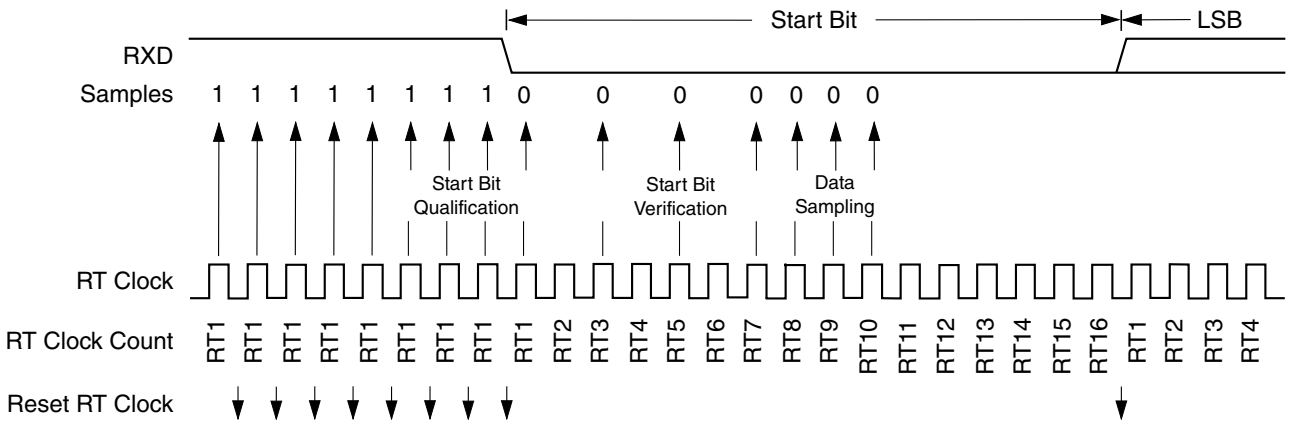


Figure 29-5. Receiver Data Sampling

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. The following table summarizes the results of the start bit verification samples.

Table 29-6. Start Bit Verification

RT3, RT5, and RT7 Samples	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 29-7. 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 ones following a successful start bit verification, the noise flag (STAT[NF]) is set and the receiver assumes that the bit is a start bit (logic zero).

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.

Table 29-8. 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. 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.

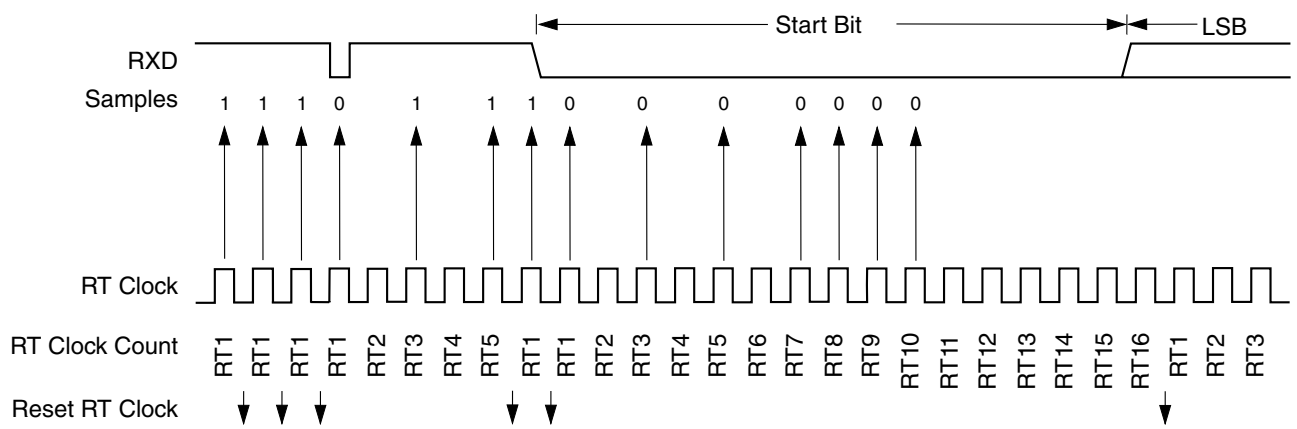


Figure 29-6. Start Bit Search Example 1

In the following figure, noise is perceived as the beginning of a start bit because the verification sample at RT3 is high. 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.

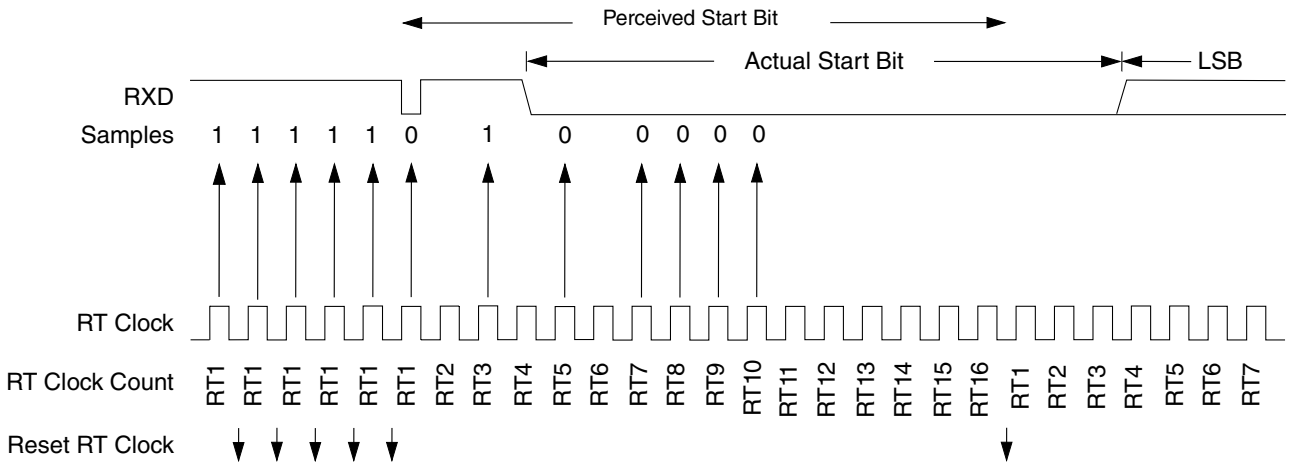


Figure 29-7. Start Bit Search Example 2

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. 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.

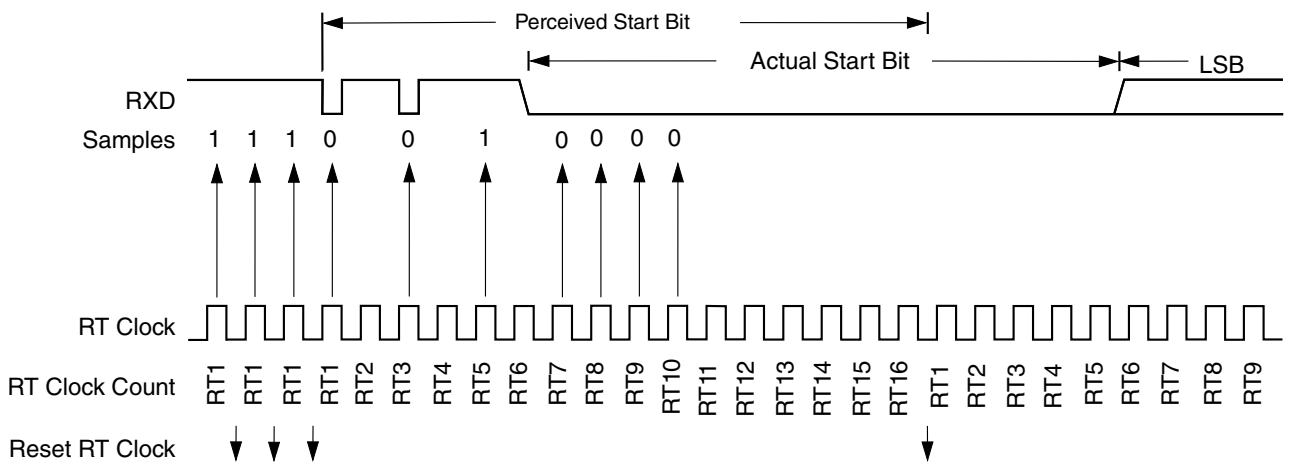


Figure 29-8. Start Bit Search Example 3

The following figure shows the effect of noise early in the start bit time. Although this noise does not affect proper synchronization with the start bit time, it does set the noise flag.

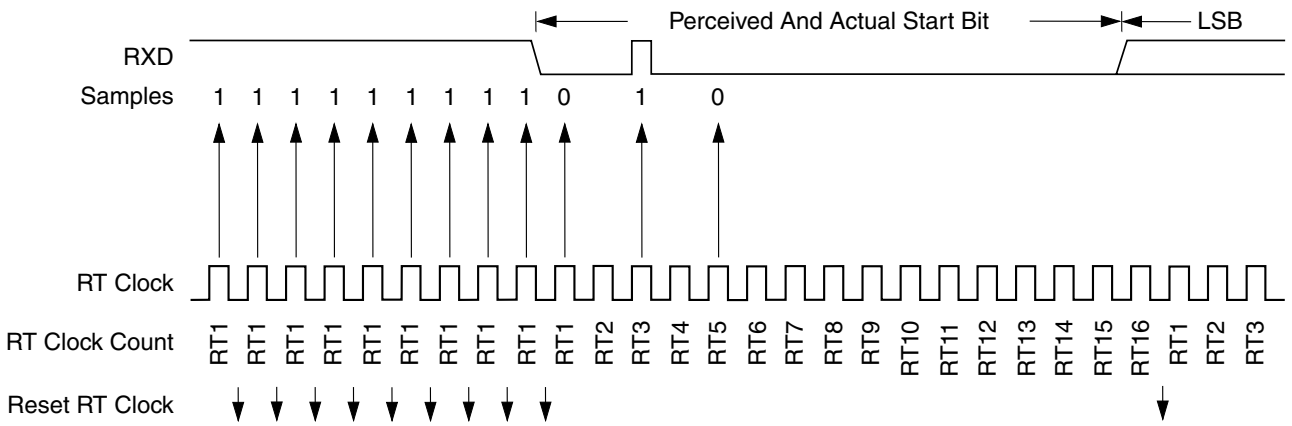


Figure 29-9. Start Bit Search Example 4

The following figure shows a burst of noise near the beginning of the start bit that resets the RT clock. 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.

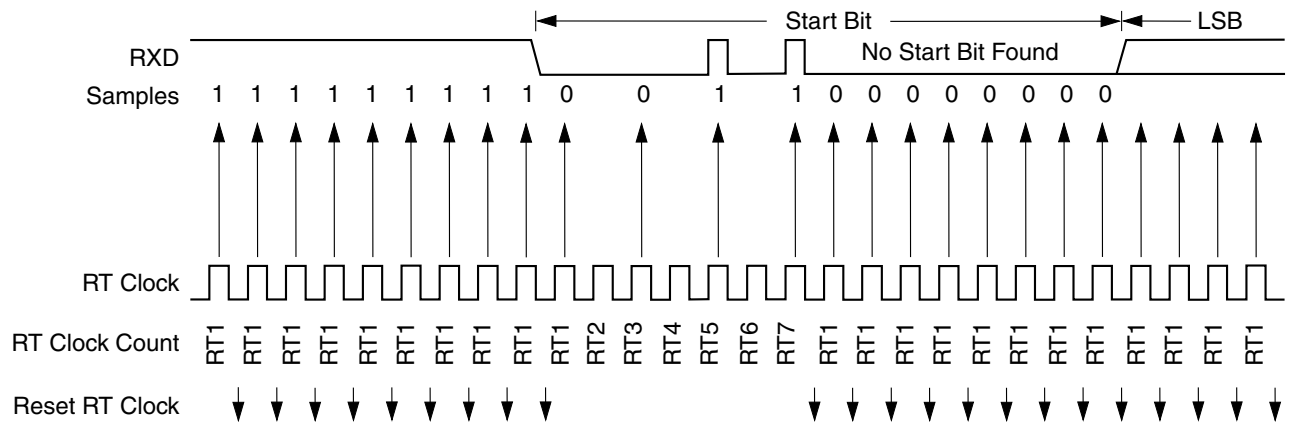


Figure 29-10. Start Bit Search Example 5

In the following figure, a noise burst makes the majority of data samples RT8, RT9, and RT10 high. 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.

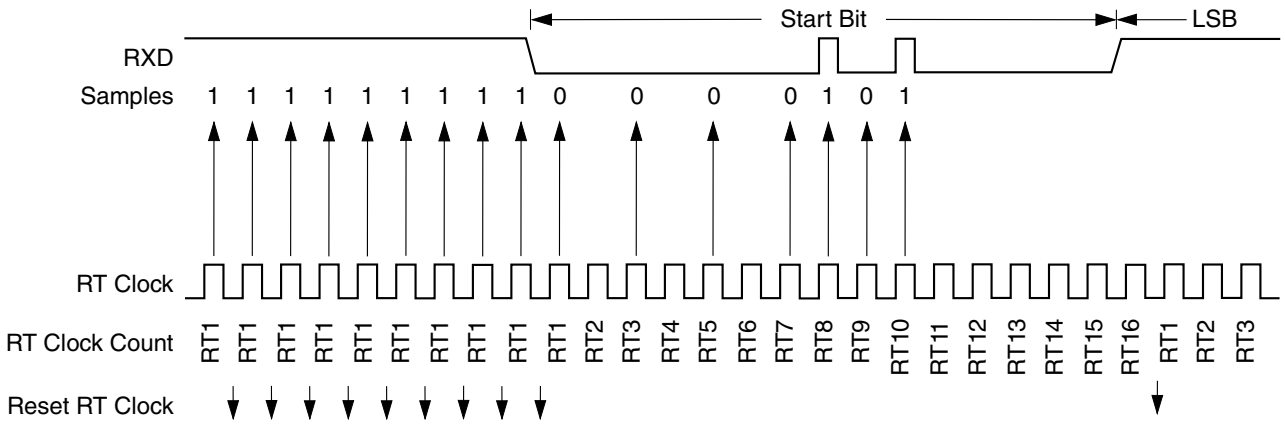


Figure 29-11. Start Bit Search Example 6

29.4.4.4 Framing Errors

If the data recovery logic does not detect a logic one where the stop bit should be in an incoming frame, it sets the framing error flag, STAT[FE]. A break character also sets STAT[FE] because a break character has no stop bit. STAT[FE] is set at the same time that STAT[RDRF] is set.

29.4.4.5 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 to fall outside the actual stop bit. Then a noise error occurs. If more than one of the samples is outside the stop bit, a framing error occurs. In most applications, the baud rate tolerance is much more than the degree of misalignment that is likely to occur.

As the receiver samples an incoming frame, it resynchronizes the RT clock on any valid falling edge within the frame. Resynchronization within frames corrects misalignments between transmitter bit times and receiver bit times.

29.4.4.6 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.

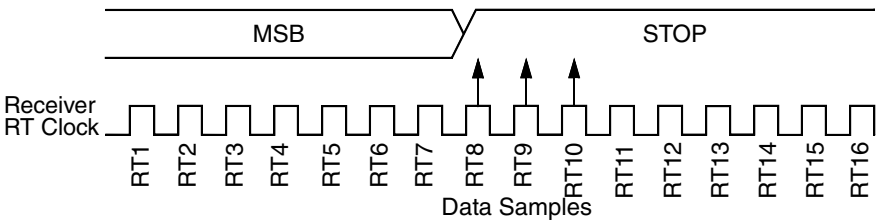


Figure 29-12. Slow Data

For an 8-bit data character, data sampling of the stop bit takes the receiver the following number of cycles:

```
9 bit × 16 RT cycles + 10 RT cycles = 154 RT cycles
```

With the misaligned character shown in the data figure, the receiver counts 154 RT cycles at the point when the count of the transmitting device is:

```
9 bit × 16 RT cycles + 3 RT cycles = 147 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) / 154) = 4.54%
```

For a 9-bit data character, data sampling of the stop bit takes the receiver the following number of cycles:

```
10 bit × 16 RT cycles + 10 RT cycles = 170 RT cycles
```

With the misaligned character shown in the slow data figure, the receiver counts 170 RT cycles at the point when the count of the transmitting device is:

```
10 bit × 16 RT cycles + 3 RT cycles = 163 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) / 170) = 4.12%
```

29.4.4.7 Fast Data Tolerance

The following figure shows how much a fast received frame can be misaligned without causing a noise error or a framing error. The fast stop bit ends at RT10 instead of RT16 but is still sampled at RT8, RT9, and RT10.

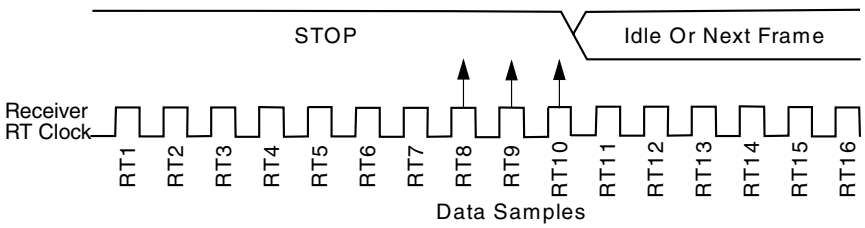


Figure 29-13. Fast Data

For an 8-bit data character, data sampling of the stop bit takes the receiver the following number of cycles:

```
9 bit × 16 RT cycles + 10 RT cycles = 154 RT cycles
```

With the misaligned character shown in the fast data figure, the receiver counts 154 RT cycles at the point when the count of the transmitting device is:

```
10 bit × 16 RT cycles = 160 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) / 154) = 3.90%
```

For a 9-bit data character, data sampling of the stop bit takes the receiver the following number of cycles:

```
10 bit × 16 RT cycles + 10 RT cycles = 170 RT cycles
```

With the misaligned character shown in the fast data figure, the receiver counts 170 RT cycles at the point when the count of the transmitting device is:

```
11 bit × 16 RT cycles = 176 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) / 170) = 3.53%
```

29.4.4.8 Receiver Wakeup

So that the SCI can ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wakeup bit, CTRL1[RWU], puts the receiver into a standby state during which receiver interrupts are disabled.

The transmitting device can address messages to selected receivers by including addressing information in the initial frame or frames of each message.

CTRL1[WAKE] determines how the SCI is brought out of the standby state to process an incoming message. CTRL1[WAKE] enables either idle line wakeup or address mark wakeup:

- Idle input line wakeup (CTRL1[WAKE] = 0): In this wakeup method, an idle condition on the RXD pin clears CTRL1[RWU] and wakes the SCI. The initial frame (or frames) of every message contains 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 CTRL1[RWU] bit and return to the standby state. CTRL1[RWU] remains set and the receiver remains on standby until another preamble appears on the RXD pin.

  The idle line wakeup method requires that messages be separated by at least one preamble and that no message contains preambles.

  The preamble that wakes a receiver does not set the receiver idle bit, STAT[RIDLE], or the receive data register full flag, STAT[RDRF].

- Address mark wakeup (CTRL1[WAKE] = 1): In this wakeup method, a logic one in the most significant bit (MSB) position of a frame clears CTRL1[RWU] and wakes up the SCI. The logic one in the MSB position 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 RWU bit and return to the standby state. CTRL1[RWU] remains set and the receiver remains on standby until another address frame appears on the RXD pin.

  The logic one MSB of an address frame clears the receiver's CTRL1[RWU] bit before the stop bit is received and sets STAT[RDRF].

  The address mark wakeup method allows messages to contain preambles but requires that the MSB be reserved for use in address frames.

Note

With CTRL1[WAKE] clear, setting CTRL1[RWU] after the RXD pin is idle can cause the receiver to wake up immediately.

29.4.4.9 Single-Wire Operation

Normally, the SCI uses two pins for transmitting and receiving. In single-wire operation, the RXD pin is disconnected from the SCI but is enabled, meaning CTRL1[RE] must be 1, and is available as a general-purpose I/O pin. The SCI uses the TXD pin for both receiving and transmitting.

Setting CTRL1[TE] configures TXD as the output for transmitted data. Clearing CTRL1[TE] configures TXD as the input for received data.

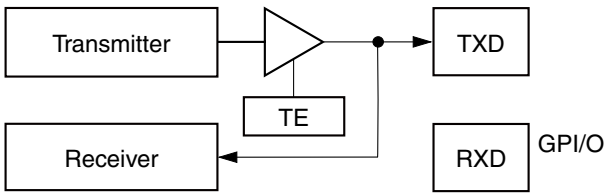


Figure 29-14. Single-Wire Operation (CTRL1[LOOP] = 1, CTRL1[RSRC] = 1)

Enable single-wire operation by setting CTRL1[LOOP] and the receiver source bit, CTRL1[RSRC]. Setting CTRL1[LOOP] disables the path from the RXD pin to the receiver. Setting CTRL1[RSRC] connects the receiver input to the output of the TXD pin driver.

29.4.4.10 Loop Operation

In loop operation the transmitter output goes to the receiver input. The RXD pin is disconnected from the SCI and is available as a general-purpose I/O pin.

Setting CTRL1[TE] connects the transmitter output to the TXD pin. Clearing CTRL1[TE] disconnects the transmitter output from the TXD pin.

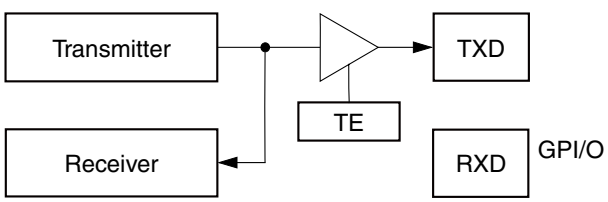


Figure 29-15. Loop Operation (CTRL1[LOOP] = 1, CTRL1[RSRC] = 0)

Enable loop operation by setting CTRL1[LOOP] and clearing CTRL1[RSRC]. Setting CTRL1[LOOP] disables the path from the RXD pin to the receiver. Clearing CTRL1[RSRC] connects the transmitter output to the receiver input. Both the transmitter and receiver must be enabled (CTRL1[TE] = 1 and CTRL1[RE] = 1).

29.4.5 DMA Operation

DMA operation is an optional feature that may not be implemented on all chips.

29.4.5.1 Transmit DMA Operation

Setting CTRL2[TDE] enables Transmit DMA mode. In this mode, the transmitter empty interrupt is suppressed, and instead a transmitter DMA request is generated whenever there is an empty space in the TX FIFO. The DMA controller then writes to the DATA register, clearing the request.

29.4.5.2 Receive DMA Operation

Setting CTRL2[RDE] enables Receiver DMA mode. In this mode, the Receiver Full interrupt is suppressed, and instead a Receiver DMA request is generated whenever there is data in the RX FIFO. The DMA controller then reads the DATA register, clearing the request.

29.4.5.3 Receiver Wakeup with DMA

If DMA operation is desired during either of the wakeup modes of operation, DMA requests should only be made for a message that is addressed to the receiver. In other words, wakeup operation proceeds normally until a desired receive message is identified. Then DMA requests are generated to buffer the remainder of the message. An example of the DMA receive operations is shown in Table 29-9.

Table 29-9. Receive DMA Operations

1. Configure the SCI for standard receive operation 2. SCI receives the first frame of the incoming message and stores it in the DATA register 3. A SCI Receiver Full interrupt occurs 4. The ISR looks at the message address information and determines that:	
MESSAGE NOT FOR US	**MESSAGE IS FOR US**
5. Configure SCI for RWU mode and wait for the end of the message. Repeat from step 1.	5. Enable DMA operation with a buffer large enough to accommodate the max message size 6. Enable CTRL2[RIIE] so the SCI interrupts at the completion of the message (assumes the DMA buffer does not fill up first). 7. When the Receiver Idle interrupt occurs: disable further Receiver Idle interrupts (by setting CTRL2[RIIE]=0), process the message and return to step 1.

29.4.6 LIN Slave Operation

LIN slave operation occurs when CTRL2[LIN MODE] is set. The receiver searches for a break character consisting of at least 11 consecutive samples of logic zero, the next field to be received is the sync field. The sync field is a word with 0x55 data that produces an alternating 0 and 1 pattern. The receiver detects the falling edge at the beginning of the start bit and starts counting system clocks until the falling edge at the beginning of data bit 7 is detected, at which point it stops counting. This count is divided by 8 (for the 8-bit periods that have passed) and further divided by 16 to provide new RATE[SBR] and RATE[FRAC_SBR] values. If the data value of the sync field is 0x55, then these new RATE[SBR] and RATE[FRAC_SBR] values are placed in the baud rate register. Then the slave is considered synced to the master and further data words are received properly. If the data value of the sync field isn't 0x55, then the LIN sync error (STAT[LSE]) bit is set and subsequent received data bytes should be ignored.

To detect the break character successfully, the initial baud rate for this slave device must be within 15 percent of the nominal baud rate for the LIN master device.

29.4.7 Low-Power Options

29.4.7.1 Run Mode

Clearing the transmitter enable or receiver enable bits (CTRL1[TE] or CTRL1[RE]) reduces power consumption in run mode. SCI registers are still accessible when CTRL1[TE] or CTRL1[RE] is cleared, but clocks to the core of the SCI are disabled.

29.4.7.2 Wait Mode

SCI operation in wait mode depends on the state of CTRL1[SWAI].

- If CTRL1[SWAI] is clear, the SCI operates normally when the DSC core is in wait mode.
- If CTRL1[SWAI] is set, SCI clock generation ceases and the SCI module enters a power-conservation state when the DSC core is in wait mode. SCI registers are not accessible. Setting CTRL1[SWAI] does not affect the state of the receiver enable bit, CTRL1[RE], or the transmitter enable bit, CTRL1[TE].

If CTRL1[SWAI] is set, any transmission or reception in progress stops at wait mode entry. The transmission or reception resumes when either an internal or external interrupt brings the DSC out of wait mode. Exiting wait mode via reset aborts any transmission or reception in progress and resets the SCI.

29.4.7.3 Stop Mode

SCI operation in stop mode depends on the state of the SCI stop disable bit in the SIM's applicable stop disable register.

- If the SCI stop disable bit is clear, the SCI is inactive in stop mode to reduce power consumption. The STOP instruction does not affect the SCI registers' states. SCI operation resumes after an interrupt brings the CPU out of stop mode. Exiting stop mode by reset aborts any transmission or reception in progress and resets the SCI.
- If the SCI stop disable bit is set, the SCI operates normally in stop mode.

29.5 Resets

Any system reset completely resets the SCI.

29.6 Clocks

All timing is derived from the IP Bus clock, which is the main clock for this module. See Baud-Rate Generation about how the data rate is determined.

29.7 Interrupts

Table 29-10. Interrupt Summary

Interrupt	Source	Description
TDRE	Transmitter	Transmit Data Register Empty interrupt
TIDLE	Transmitter	Transmit Idle interrupt
RIDLE	Receiver	Receive Idle interrupt
RERR	Receiver	Receive Error (FE, NF, PF, LSE, or OR) interrupt
RDRF/OR	Receiver	Receive Data Register Full / Overrun / Active Edge interrupt

29.7.1 Description of Interrupt Operation

Table 29-11. SCI Interrupt Sources

Interrupt Source	Flag	Local Enable	Description
Transmitter	STAT[TDRE]	CTRL1[TEIE]	Transmit Data Register Empty interrupt
Transmitter	STAT[TIDLE]	CTRL1[TIIE]	Transmit Idle interrupt
Receiver	STAT[RIDLE]	CTRL2[RIIE]	Receive Idle interrupt Only implemented if DMA is included in the chip architecture
Receiver	STAT[RDRF] STAT[OR]	CTRL1[RFIE]	Receive Data Register Full / Overrun / Active Edge interrupt
	STAT[RIEF]	CTRL2[RIEIE]	
Receiver	STAT[FE] STAT[NF] STAT[PF] STAT[LSE] STAT[OR]	CTRL1[REIE]	Receive Error (FE, NF, PF, LSE, or OR) interrupt

29.7.1.1 Transmitter Empty Interrupt

The transmitter empty interrupt is enabled by setting CTRL1[TEIE]. When this interrupt is enabled, an interrupt is generated while data is transferred from the SCI data register to the transmit shift register. The interrupt service routine should read the STAT register, verify that STAT[TDRE] is set, and write the next data to be transmitted to the DATA register, which clears STAT[TDRE].

NOTE

This interrupt is disabled if CTRL2[TDE] is set, enabling transmit DMA operations.

29.7.1.2 Transmitter Idle Interrupt

The transmitter idle interrupt is enabled by setting CTRL1[TIIE]. This interrupt indicates that STAT[TIDLE] is set and the transmitter is no longer sending data, preamble, or break characters. The interrupt service routine should read the STAT register, verify STAT[TIDLE] is set, and initiate a preamble, break, or write a data character to the DATA register. Any of these actions clears STAT[TIDLE] because the transmitter is then busy.

29.7.1.3 Receiver Full Interrupt

The receiver full interrupt is enabled by setting CTRL1[RFIE]. This interrupt indicates that receive data is available in the DATA register. The interrupt service routine should read the STAT register, verify that STAT[RDRF] is set, and then read the data from the DATA register, which clears STAT[RDRF].

NOTE

This interrupt is disabled if CTRL2[RDE] is set, enabling receive DMA operations.

29.7.1.4 Receiver Edge Interrupt

This interrupt is used signal that an active edge has been observed on the RXD input pin. An interrupt is generated only when enabled using the CTRL2[RIEIE] bit. The CTRL[POL] bit determines which logic state is considered active. This interrupt is typically used to wake the part from stop mode.

29.7.1.5 Receive Error Interrupt

The receive error interrupt is enabled by setting CTRL1[REIE]. This interrupt indicates that the receiver detected any of the errors reflected by the following conditions:

- Noise flag (STAT[NF]) set
- Parity error flag (STAT[PF]) set
- Framing error flag (STAT[FE]) set
- Overrun flag (STAT[OR]) set
- LIN sync error flag (STAT[LSE]) set

The interrupt service routine should read the STAT register to determine which of the error flags was set. The error flag is cleared by writing (anything) to the STAT register. Then software should take the appropriate action to handle the error condition.

29.7.1.6 Receiver Idle Interrupt

This interrupt is used in conjunction with receive DMA operation. See the CTRL2[RIIE] bit's description and Receiver Wakeup with DMA for a description of the intended operation of this interrupt. When this interrupt occurs, the appropriate response of the interrupt service routine is to disable the Receiver Idle until the next message receive sequence occurs.

29.7.2 Recovery from Wait and Stop Mode

Any enabled SCI interrupt request can bring the DSC core out of wait mode or stop mode (if the SCI module is enabled in stop mode).

29.8 DMA Requests

Table 29-12. SCI DMA Request Sources

DMA Request Source	Flag	Local Enable	Description
Transmitter	STAT[TDMA]	CTRL2[TDE]	Transmit Data Write Request
Receiver	STAT[RDMA]	CTRL2[RDE]	Receive Data Read Request

29.8.1 Transmit Data Write Request

This request is enabled by setting CTRL2[TDE]. Once enabled, the request is generated when there is an empty space in the TX data register/FIFO. The DMA controller should write data to the register/FIFO until full.

29.8.2 Receive Data Read Request

This request is enabled by setting CTRL2[RDE]. Once enabled, the request is generated when there is data in the RX data register/FIFO. The DMA controller should read the data register/FIFO until empty.

Chapter 30
Queued Serial Peripheral Interface (QSPI)

30.1 Introduction

30.1.1 Overview

The serial peripheral interface (SPI) module enables full-duplex, synchronous, serial communication between the chip and peripheral devices, including other chips. Software can poll the SPI status flags or SPI operation can be interrupt driven. The block contains six 16-bit memory mapped registers for control parameters, status, and data transfer.

Features of the SPI module include the following:

- Full-duplex operation
- Master and slave modes
- Double-buffered operation with separate transmit and receive registers
- Programmable Length Transactions (2 to 16 bits)
- Programmable transmit and receive shift order (MSB or LSB first)
- Fourteen master mode frequencies (maximum = bus frequency ÷ 2)
- Maximum slave mode frequency = bus frequency ÷ 4
- Serial clock with programmable polarity and phase
- Two separately enabled interrupts:
    - SPRF (SPI receiver full)
    - SPTE (SPI transmitter empty)
- Mode fault error flag with interrupt capability
- Overflow error flag with interrupt capability

- Wired OR mode functionality enabling connection to multiple SPIs
- Stop mode holdoff
- Separate RX and TX FIFO capable of handling 4 transactions

Maximum SPI data rates are limited by I/O pad performance as specified in the device's data sheet.

30.1.2 Block Diagram

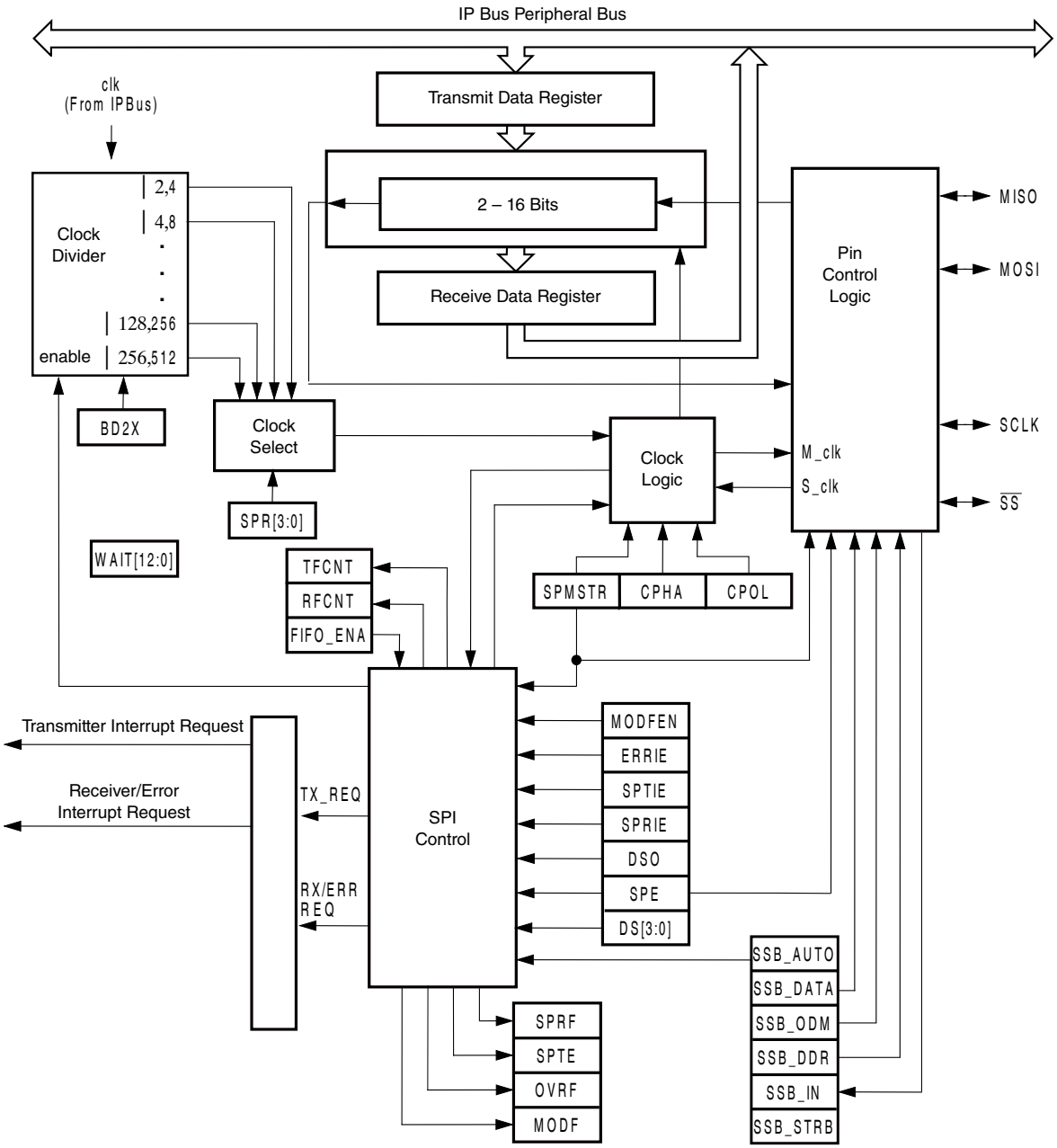


Figure 30-1. SPI Block Diagram

30.2 Signal Descriptions

30.2.1 External I/O Signals

The following are external I/O signals at the device interface. All of these pins are bidirectional.

Table 30-1. External I/O

Signal Name	Description	Direction	
		Master	Slave
MOSI	Master-out Slave-in Pad Pin	Output	Input
MISO	Master-in Slave-out Pad Pin	Input	Output
SCLK	Slave Clock Pad Pin	Output	Input
$\overline{SS}$	Slave Select Pad Pin (Active Low)	See Table 30-2.	

30.2.1.1 MISO (Master In/Slave Out)

MISO is one of the two SPI module pins that transmit serial data. In full duplex operation, the MISO pin of the master SPI module is connected to the MISO pin of the slave SPI module. The master SPI simultaneously receives data on its MISO pin and transmits data from its MOSI pin.

Slave output data on the MISO pin is enabled only when the SPI is configured as a slave. The SPI is configured as a slave when its SPMSTR bit (see the description of the SPI status and control register) is logic zero and its SS pin is at logic zero. To support a multiple-slave system, a logic one on the $\overline{SS}$ pin puts the MISO pin in a high-impedance state.

30.2.1.2 MOSI (Master Out/Slave In)

MOSI is one of the two SPI module pins that transmits serial data. In full-duplex operation, the MOSI pin of the master SPI module is connected to the MOSI pin of the slave SPI module. The master SPI simultaneously transmits data from its MOSI pin and receives data on its MISO pin.

30.2.1.3 SCLK (Serial Clock)

The serial clock synchronizes data transactions between master and slave devices. In a master device, the SCLK pin is the clock output. In a slave device, the SCLK pin is the clock input. In full duplex operation, the master and slave devices exchange data in the same number of clock cycles as the number of bits of transmitted data.

30.2.1.4 $\overline{SS}$ (Slave Select)

The $\overline{SS}$ pin has various functions depending on the current state of the SPI. For an SPI configured as a slave, the $\overline{SS}$ is used to select a slave. For CPHA = 0, the $\overline{SS}$ is used to define the start of a transaction. Because it is used to indicate the start of a transaction, the $\overline{SS}$ must be toggled high and low between each full length set of data transmitted for the CPHA = 0 format. However, it can remain low between transactions for the CPHA = 1 format.

When an SPI is configured as a slave, the $\overline{SS}$ pin is always configured as an input. The MODFEN bit can prevent the state of the $\overline{SS}$ from creating a MODF error.

NOTE

A logic one voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high-impedance state. The slave SPI ignores all incoming SCLK clocks, even if it is already in the middle of a transaction. A mode fault occurs if the $\overline{SS}$ pin changes state during a transaction.

When an SPI is configured as a master, the $\overline{SS}$ input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SCLK. For the state of the $\overline{SS}$ pin to set the MODF flag, the MODFEN bit in the SPI Status and Control register must be set.

Table 30-2. SPI IO Configuration

SPE	SPMSTR	MODFEN	SPI CONFIGURATION	STATE OF SS_B LOGIC
0	X[1]	X	Not Enabled	$\overline{SS}$ ignored by SPI
1	0	X	Slave	Input-only to SPI
1	1	0	Master without MODF	$\overline{SS}$ input ignored by SPI, $\overline{SS}$ output may be activated under software or hardware control to select slave devices.
1	1	1	Master with MODF	Input-only to SPI

1. X = don't care

30.3 Memory Map Registers

Six registers control and monitor QSPI module operation. These registers should be accessed only with word accesses. Accesses with lengths other than word lengths result in undefined results. Before QSPI registers can be changed, the bit corresponding to the QSPI must be set to 1 in the appropriate PCE register of the SIM.

QSPI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E0B0	SPI Status and Control Register (QSPI0_SPSCR)	16	R/W	6141h	30.3.1/910
E0B1	SPI Data Size and Control Register (QSPI0_SPDSR)	16	R/W	See section	30.3.2/913
E0B2	SPI Data Receive Register (QSPI0_SPDRR)	16	R	0000h	30.3.3/916
E0B3	SPI Data Transmit Register (QSPI0_SPDTR)	16	W	0000h	30.3.4/917
E0B4	SPI FIFO Control Register (QSPI0_SPFIFO)	16	R/W	000Ch	30.3.5/919
E0B5	SPI Word Delay Register (QSPI0_SPWAIT)	16	R/W	0000h	30.3.6/921
E0B6	SPI Control Register 2 (QSPI0_SPCTL2)	16	R/W	0000h	30.3.7/921

30.3.1 SPI Status and Control Register (QSPIx_SPSCR)

This register does the following:

- Selects master SPI baud rate
- Determines data shift order
- Enables SPI module interrupt requests
- Configures the SPI module as master or slave
- Selects serial clock polarity and phase

NOTE

Using BFCLR or BFSET instructions on this register can cause unintended side effects on the status bits.

Address: E0B0h base + 0h offset = E0B0h

Bit	15	14	13	12	11	10	9	8
Read Write	SPR[2:0]			DSO	ERRIE	MODFEN	SPRIE	SPMSTR
Reset	0	1	1	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
Read	CPOL	CPHA	SPE	SPTIE	SPRF	OVRF	MODF	SPTE
Write								
Reset	0	1	0	0	0	0	0	1

QSPIx_SPSCR field descriptions

Field	Description

15–13
SPR[2:0]
	SPI Baud Rate Select
	
	In master mode, these read/write bits select one of eight baud rates. SPR2, SPR1, and SPR0 have no effect in slave mode. Reset sets SPR[2:0] to b011.
	
	Use the following formula to calculate the SPI baud rate:
	
	`Baud rate = clk/BD`
	
	where:
	
	`clk = Peripheral Bus Clock`
	`BD = baud rate divisor`
	
	Restriction: The maximum data transmission rate for the SPI is typically limited by the bandwidth of the I/O drivers on the chip, which can be a function of manufacturing technology. The typical limit in Normal mode is 40 MHz and in Wired-OR mode is 10 MHz. These baud rate limitations apply to both master and slave mode. The value of BD must be set to ensure the module remains within these ranges.
	
	NOTE: The value of BD can also depend on the values of the SPR3 and BD2X fields in the SPI Data Size and Control Register.
	
	000 BD = 2 when SPR3 = 0, BD = 512 when SPR3 = 1 (double BD when BD2X = 1)
	001 BD = 4 when SPR3 = 0, BD = 1024 when SPR3 = 1 (double BD when BD2X = 1)
	010 BD = 8 when SPR3 = 0, BD = 2048 when SPR3 = 1 (double BD when BD2X = 1)
	011 BD = 16 when SPR3 = 0, BD = 4096 when SPR3 = 1 (double BD when BD2X = 1)
	100 BD = 32 when SPR3 = 0, BD = 8192 when SPR3 = 1 (double BD when BD2X = 1)
	101 BD = 64 when SPR3 = 0 (double BD when BD2X = 1), BD = 16384 when SPR3 = 1 (regardless of BD2X)
	110 BD = 128 when SPR3 = 0 (double BD when BD2X = 1), BD = 16384 when SPR3 = 1 (regardless of BD2X)
	111 BD = 256 when SPR3 = 0 (double BD when BD2X = 1), BD = 16384 when SPR3 = 1 (regardless of BD2X)

12 DSO	Data Shift Order This read/write bit determines which bit is transmitted or received first, either the MSB or LSB. Both master and slave SPI modules must transmit and receive packets of the same length. Regardless of how this bit is set, when reading from the data receive register or writing to the data transmit register, the LSB is always at bit location 0 and the MSB is at the correct bit position. If the data length is less than 16 bits, the upper bits of data are zero padded. 0 MSB transmitted first (MSB -> LSB) 1 LSB transmitted first (LSB -> MSB)
11 ERRIE	Error Interrupt Enable This read/write bit enables the MODF (if MODFEN is also set) and OVRF bits to generate device interrupt requests. Reset clears the ERRIE bit.

Table continues on the next page...

QSPIx_SPSCR field descriptions (continued)

Field	Description
	0 MODF and OVRF cannot generate device interrupt requests 1 MODF and OVRF can generate device interrupt requests
10 MODFEN	Mode Fault Enable This read/write bit, when set to 1, allows the MODF flag to be set. If the MODF flag is set, clearing the MODFEN bit does not clear the MODF flag. If the MODFEN bit is low, the level of the SS_B pin does not affect the operation of an enabled SPI configured as a master. If configured as a master and MODFEN=1, a transaction in progress will stop if SS_B goes low. For an enabled SPI configured as a slave, having MODFEN low only prevents the MODF flag from being set. It does not affect any other part of SPI operation.
9 SPRIE	SPI Receiver Interrupt Enable This read/write bit enables interrupt requests generated by the SPRF bit or the receive FIFO watermark register. 0 SPRF interrupt requests disabled 1 SPRF interrupt requests enabled
8 SPMSTR	SPI Master This read/write bit selects master mode operation or slave mode operation. 0 Slave mode 1 Master mode
7 CPOL	Clock Polarity This read/write bit determines the logic state of the SCLK pin between transactions. To transmit data between SPI modules, the SPI modules must have identical CPOL values. 0 Rising edge of SCLK starts transaction 1 Falling edge of SCLK starts transaction
6 CPHA	Clock Phase This read/write bit controls the timing relationship between the serial clock and SPI data. To transmit data between SPI modules, the SPI modules must have identical CPHA values. When CPHA = 0, the SS_B pin of the slave SPI module must be set to 1 between data words. To set SSB to 1 between data words when SSB_AUTO is 1, set SSB_STRB to 1. **Restriction:** *Do not* use CPHA = 0 while in DMA mode.
5 SPE	SPI Enable This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI. **Restriction:** When you change the SPE bit, the write statement must change *only* the SPE bit. Change any other bits in a separate write statement. In master mode the SPE bit can be cleared by a mode fault condition. 0 SPI module disabled 1 SPI module enabled
4 SPTIE	Transmit Interrupt Enable This read/write bit enables interrupt requests generated by the SPTE bit or the transmit FIFO watermark register.

Table continues on the next page...

QSPIx_SPSCR field descriptions (continued)

Field	Description
	0 SPTE interrupt requests disabled 1 SPTE interrupt requests enabled
3 SPRF	SPI Receiver Full This clearable, read-only flag is set each time data transfers from the shift register to the data receive register and no space is available in the RX queue to receive new data (RX FIFO is full). SPRF generates an interrupt request if the SPRIE bit in the SPI control register is set. This bit automatically clears after the data receive register is read. 0 Receive data register or FIFO is not full. (If using the FIFO, read RFCNT to determine the number of valid words available.) 1 Receive data register or FIFO is full.
2 OVRF	Overflow This clearable, read-only flag is set if software does not read the data in the receive data register before the next full data enters the shift register. In an overflow condition, the data already in the receive data register is unaffected, and the data shifted in last is lost. Clear the OVRF bit by reading the SPI status and control register with OVRF set and then reading the receive data register. 0 No overflow 1 Overflow
1 MODF	Mode Fault This clearable, read-only flag is set in a slave SPI if the SS_B pin goes high during a transaction with the MODFEN bit set. In a master SPI, the MODF flag is set if the SS_B pin goes low at any time with the MODFEN bit set. Clear the MODF bit by writing a one to the MODF bit when it is set. 0 SS_B pin at appropriate logic level 1 SS_B pin at inappropriate logic level
0 SPTE	SPI Transmitter Empty This clearable, read-only flag is set each time the transmit data register transfers data into the shift register and there is no more new data available in the TX queue (TX FIFO is empty). SPTE generates an interrupt request if the SPTIE bit in the SPI control register is set. SPTE is cleared by writing to the data transmit register. **CAUTION:** Do not write to the SPI data register unless the SPTE bit is high. Otherwise, data may be lost. 0 Transmit data register or FIFO is not empty. (If using the FIFO, read TFCNT to determine how many words can be written safely.) 1 Transmit data register or FIFO is empty.

30.3.2 SPI Data Size and Control Register (QSPIx_SPDSR)

This read/write register determines the data length for each transaction. The master and slave must transfer the same size data on each transaction. A new value takes effect only at the time the SPI is enabled (the SPE bit in the status and control register is set from 0 to 1). To have a new value take effect, disable and then re-enable the SPI with the new value in the register.

To use the SS_B control functions in master mode, set the appropriate bit in a GPIO peripheral enable register to enable peripheral control of the SS_B pin.

Address: E0B0h base + 1h offset = E0B1h

Bit	15	14	13	12	11	10	9	8
Read	WOM	TDMAEN	RDMAEN	BD2X	SSB_IN	SSB_DATA	SSB_ODM	SSB_AUTO
Write								
Reset	0	0	0	0	x*	1	0	0

Bit	7	6	5	4	3	2	1	0
Read	SSB_DDR	SSB_STRB	SSB_OVER	SPR3	DS[3:0]			
Write								
Reset	0	0	0	0	1	1	1	1

* Notes:
- x = Undefined at reset.

QSPIx_SPDSR field descriptions

Field	Description
15 WOM	Wired-OR Mode The Wired-OR mode (WOM) control bit is used to select the nature of the SPI pins. When enabled (the WOM bit is set), the SPI pins are configured as open-drain drivers. When disabled (the WOM bit is cleared), the SPI pins are configured as push-pull drivers. 0 The SPI pins are configured as push-pull drivers. 1 The SPI pins are configured as open-drain drivers with the pull-ups disabled.
14 TDMAEN	Transmit DMA Enable This read/write bit enables DMA control for transmit data.
13 RDMAEN	Receive DMA Enable This read/write bit enables DMA control for receive data.
12 BD2X	Baud Divisor Times Setting this bit causes the Baud Rate Divisor (BD) to be multiplied by two. The SPR bits define BD.
11 SSB_IN	SS_B Input This read only bit shows the current state of the SS_B pin in all modes. The bit's reset state is undefined.
10 SSB_DATA	SS_B Data This read/write bit is the value to drive on the SS_B pin. This bit is disabled when SSB_AUTO=1 or SSB_STRB=1. 0 SS_B pin is driven low if SSB_DDR=1 1 SS_B pin is driven high if SSB_DDR=1
9 SSB_ODM	SS_B Open Drain Mode This read/write bit enables open drain mode on the SS_B pin in master mode.

Table continues on the next page...

QSPIx_SPDSR field descriptions (continued)

Field	Description
	0 SS_B is configured for high and low drive. This mode is generally used in single master systems. 1 SS_B is configured as an open drain pin (only drives low output level). This mode is useful for multiple master systems.
8 SSB_AUTO	SS_B Automatic Mode This read/write bit enables hardware control of the SS_B pin in master mode. (The legacy design requires software to control the SS_B output pin.) The initial falling edge of SS_B is generated and SS_B is held low until the TX buffer or FIFO is empty. This bit may be used alone or in combination with SS_STRB to generate the required SS_B signal. 0 SS_B output signal is software generated by directly manipulating the various bits in this register or the GPIO registers (compatible with legacy SPI software). 1 SS_B output signal is hardware generated to create the initial falling edge and final rising edge. The idle state of the SS_B is high. **Restriction:** Do not use if MODFEN = 1.
7 SSB_DDR	SS_B Data Direction This read/write bit controls input/output mode on the SS_B pin in master mode. 0 SS_B is configured as an input pin. Use this setting in slave mode or in master mode with MODFEN=1. 1 SS_B is configured as an output pin. Use this setting in master mode with MODFEN=0.
6 SSB_STRB	SS_B Strobe Mode This read/write bit enables hardware pulse of the SS_B pin in master mode between words. This bit may be used alone or in combination with the SS_AUTO to generate the required SS_B signal. Pulses are generated between words irrespective of the setting of CPHA. 0 No SS_B pulse between words. 1 SS_B output signal is pulsed high between words. This adds 1.5 baud clocks to the total word period. The idle state of SS_B is low unless SSB_AUTO is high and then the idle state is high. **Restriction:** Do not use if MODFEN = 1.
5 SSB_OVER	SS_B Override This read/write bit overrides the internal SS_B signal input from the I/O pad and replaces it with a level equal to the setting of the SPMSTR bit. This allows the SPI to function in slave mode, when CPHA=1, without commiting a GPIO pin to be tied low. **Restriction:** This bit should not be used in multi-slave systems or when CPHA=0. **Restriction:** This bit should not be used in a multi-master system because in master mode a mode fault error cannot be generated. 0 SS_B internal module input is selected to be connected to a GPIO pin. 1 SS_B internal module input is selected to be equal to SPMSTR.
4 SPR3	SPI Baud Rate Select Use this bit with SPR[2:0] in the status and control register and BD2X to define the Baud Rate Divisor (BD).
DS[3:0]	Transaction data size 4'h0 Not allowed

Table continues on the next page...

QSPIx_SPDSR field descriptions (continued)

Field	Description
	4'h1 2 bits transaction data size 4'h2 3 bits transaction data size 4'h3 4 bits transaction data size 4'h4 5 bits transaction data size 4'h5 6 bits transaction data size 4'h6 7 bits transaction data size 4'h7 8 bits transaction data size 4'h8 9 bits transaction data size 4'h9 10 bits transaction data size 4'hA 11 bits transaction data size 4'hB 12 bits transaction data size 4'hC 13 bits transaction data size 4'hD 14 bits transaction data size 4'hE 15 bits transaction data size 4'hF 16 bits transaction data size

30.3.3 SPI Data Receive Register (QSPIx_SPDRR)

The SPI data receive register is read-only. Reading data from the register shows the last data received after a complete transaction. The SPRF bit is set when new data is transferred to this register.

Address: E0B0h base + 2h offset = E0B2h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QSPIx_SPDRR field descriptions

Field	Description
15 R15	Receive Data Bit 15
14 R14	Receive Data Bit 14
13 R13	Receive Data Bit 13
12 R12	Receive Data Bit 12
11 R11	Receive Data Bit 11

Table continues on the next page...

QSPIx_SPDRR field descriptions (continued)

Field	Description
10 R10	Receive Data Bit 10
9 R9	Receive Data Bit 9
8 R8	Receive Data Bit 8
7 R7	Receive Data Bit 7
6 R6	Receive Data Bit 6
5 R5	Receive Data Bit 5
4 R4	Receive Data Bit 4
3 R3	Receive Data Bit 3
2 R2	Receive Data Bit 2
1 R1	Receive Data Bit 1
0 R0	Receive Data Bit 0

30.3.4 SPI Data Transmit Register (QSPIx_SPDTR)

The SPI data transmit register is write-only. Writing data to this register writes the data to the transmit data buffer. When the SPTE bit is set, new data should be written to this register. If new data is not written while in master mode, a new transaction will not begin until this register is written.

When selected in slave mode, the old data will be re-transmitted. When *not* selected and in slave mode, transmit data will remain unchanged. All data should be written with the LSB at bit 0. This register can only be written when the SPI is enabled (SPE = 1).

Address: E0B0h base + 3h offset = E0B3h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read																
Write	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QSPIx_SPDTR field descriptions

Field	Description
15 T15	Transmit Data Bit 15
14 T14	Transmit Data Bit 14
13 T13	Transmit Data Bit 13
12 T12	Transmit Data Bit 12
11 T11	Transmit Data Bit 11
10 T10	Transmit Data Bit 10
9 T9	Transmit Data Bit 9
8 T8	Transmit Data Bit 8
7 T7	Transmit Data Bit 7
6 T6	Transmit Data Bit 6
5 T5	Transmit Data Bit 5
4 T4	Transmit Data Bit 4
3 T3	Transmit Data Bit 3
2 T2	Transmit Data Bit 2
1 T1	Transmit Data Bit 1
0 T0	Transmit Data Bit 0

30.3.5 SPI FIFO Control Register (QSPIx_SPFIFO)

This register is used for FIFO control and status.

Address: E0B0h base + 4h offset = E0B4h

Bit	15	14	13	12	11	10	9	8
Read	0	TFCNT			0	RFCNT		
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0	TFWM		0	RFWM		0	FIFO_ENA
Write								
Reset	0	0	0	0	1	1	0	0

QSPIx_SPFIFO field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–12 TFCNT	TX FIFO Level These read-only bits show how many words are used in the TX FIFO. Writes to the data transmit register cause TFCNT to increment, and, as words are pulled for transmission, TFCNT is decremented. Attempts to write new data to the data transmit register are ignored when TFCNT indicates the FIFO is full. If master mode is enabled, transmission continues until the FIFO is empty, even if SPE is set to 0. 000 Tx FIFO empty (if enabled Transmit Empty Interrupt asserted) 001 One word used in Tx FIFO 010 Two words used in Tx FIFO 011 Three words used in Tx FIFO 100 Tx FIFO full
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–8 RFCNT	RX FIFO Level These read-only bits show how many words are used in the RX FIFO. As words are received, the value of RFCNT is incremented; as words are read from the data receive register, the value of RFCNT is decremented. There is one word time to read the data receive register between when the SPRF status bit is set (interrupt asserted) and when an overflow condition is flagged. 000 Rx FIFO empty 001 One word used in Rx FIFO 010 Two words used in Rx FIFO 011 Three words used in Rx FIFO 100 Rx FIFO full (if enabled Receiver Full Interrupt asserted)

Table continues on the next page...

QSPIx_SPFIFO field descriptions (continued)

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–5 TFWM	Tx FIFO Watermark These read/write bits determine how many words must remain in the Tx FIFO before an interrupt is generated. Increasing the value of TFWM increases the allowable latency in servicing the Tx interrupt without underrunning the Tx buffer space. Larger values of TFWM may also increase the number of Tx interrupt service requests because the maximum number of Tx words may not be available when the service routine is activated. If TFWM is set to the minimum value then only one SPI word time in interrupt service latency is allowed before an underrun condition results and continuous transmission is stopped in master mode or the last data word is re-transmitted in slave mode. This field is ignored when FIFO_ENA = 0. To clear an interrupt generated by TFWM, new words must be written to the data transmit register or the value of TFWM must be reduced. 00 Transmit interrupt active when Tx FIFO is empty 01 Transmit interrupt active when Tx FIFO has one or fewer words available 10 Transmit interrupt active when Tx FIFO has two or fewer words available 11 Transmit interrupt active when Tx FIFO has three or fewer words available
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–2 RFWM	Rx FIFO Watermark These read/write bits determine how many words must be used in the Rx FIFO before an interrupt is generated. Decreasing the value of RFWM increases the allowable latency in servicing the Rx interrupt without overrunning the Rx buffer space. Smaller values of RFWM may also increase the number of Rx interrupt service requests because the maximum number of Rx words may not have been used when the service routine is activated. If RFWM is set to the maximum value then only one SPI word time in interrupt service latency is allowed before an overrun condition results and receive data is lost. This field is ignored when FIFO_ENA = 0. To clear an interrupt generated by RFWM, words must be read from the data receive register or the value of RFWM must be increased. 00 Receive interrupt active when Rx FIFO has at least one word used 01 Receive interrupt active when Rx FIFO has at least two words used 10 Receive interrupt active when Rx FIFO has at least three words used 11 Receive interrupt active when Rx FIFO is full
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 FIFO_ENA	FIFO Enable This read/write bit enables Tx and Rx FIFOs' mode. 0 FIFOs are disabled and reset. 1 FIFOs are enabled. FIFOs retain their status even if SPE is set to 0.

30.3.6 SPI Word Delay Register (QSPIx_SPWAIT)

This register is used to control the delay between words.

Address: E0B0h base + 5h offset = E0B5h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0			WAIT												
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QSPIx_SPWAIT field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
WAIT	Wait Delay This 13-bit register controls the time between data transactions in master mode. It sets the delay between words to be a number of Peripheral Bus Clocks equal to (WAIT + 1). This delay is used only when a word is waiting to be transmitted at the completion of the transmission of the current word. If no word is waiting to be transmitted, the SPI goes idle at the completion of the current transmission and subsequently starts transmission immediately when a new word is written to the Data Transmit register.

30.3.7 SPI Control Register 2 (QSPIx_SPCTL2)

This register controls the stop mode holdoff feature.

Address: E0B0h base + 6h offset = E0B6h

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

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

QSPIx_SPCTL2 field descriptions

Field	Description
15–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

QSPIx_SPCTL2 field descriptions (continued)

Field	Description
0 SHEN	Stop Mode Holdoff Enable When enabled, this bit allows the SPI module to hold off entry to chip level stop mode if a word is being transmitted or received. Stop mode will be entered after the SPI finishes transmitting/receiving. This bit does not allow the SPI to wake the chip from stop mode in any way. The SHEN bit can only delay the entry into stop mode. This bit should not be set in slave mode because the state of SS_B (which would be controlled by an external master device) may cause the logic to hold off stop mode entry forever. 0 Disable stop mode holdoff . 1 Enable stop mode holdoff while the SPI is transmitting/receiving.

30.4 Functional Description

30.4.1 Operating Modes

30.4.1.1 Master Mode

The SPI operates in master mode when the SPI master bit, SPMSTR, is set.

Note

Configure the SPI module as master or slave before enabling the SPI. Enable the master SPI before enabling the slave SPI. Disable the slave SPI before disabling the master SPI.

Only a master SPI module can initiate transactions. With the SPI enabled, software begins the transaction from the master SPI module by writing to the transmit data register. If the shift register is empty, the data immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE. The data begins shifting out on the MOSI pin under the control of the SPI serial clock, SCLK.

The SPR3, SPR2, SPR1, and SPR0 bits in the SPI registers control the baud rate generator and determine the speed of the shift register. Through the SCLK pin, the baud rate generator of the master also controls the shift register of the slave peripheral

As the data shifts out on the MOSI pin of the master, external data shifts in from the slave on the master's MISO pin. The transaction ends when the receiver full bit, SPRF, becomes set. At the same time that SPRF becomes set, the data from the slave transfers to the SPI Data Receive register. In normal operation, SPRF signals the end of a transaction. Software clears SPRF by reading the SPI Data Receive register. Writing to the SPI Data Transmit register clears the SPTE bit.

The following figure is an example configuration for a full-duplex master-slave configuration. Having the $\overline{SS}$ bit of the master device held high is only necessary if MODFEN = 1. Tying the slave $\overline{SS}$ bit to ground should only be done if CPHA = 1.

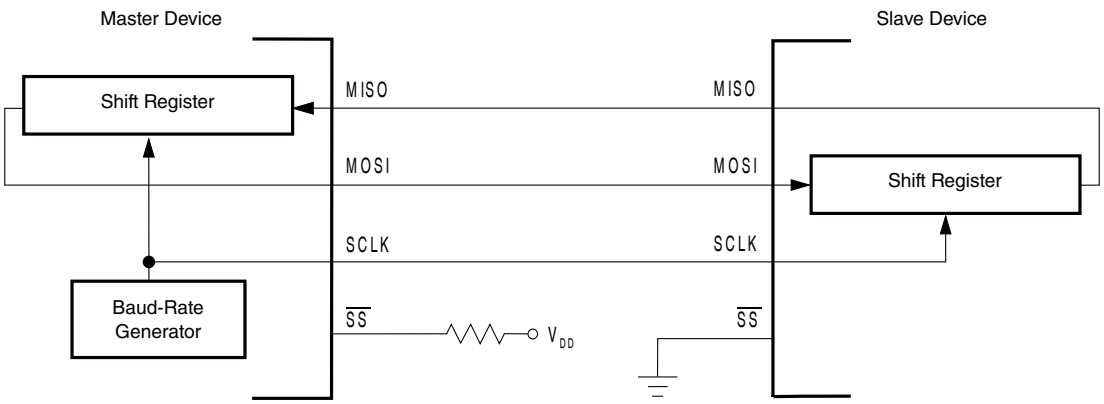


Figure 30-2. Full-Duplex Master-Slave Connections

30.4.1.2 Slave Mode

The SPI operates in slave mode when the SPMSTR bit is 0. In slave mode the SCLK pin is the input for the serial clock from the master device. Before a data transaction occurs, the SS pin of the slave SPI must be at logic zero. $\overline{SS}$ must remain low until the transaction completes or a mode fault error occurs.

Note

The SPI must be enabled (SPE = 1) for slave transactions to be received.

Note

Data in the transmitter shift register is unaffected by SCLK transitions when the SPI operates as a slave but is deselected ($\overline{SS}$ = 1).

In a slave SPI module, data enters the shift register under the control of the serial clock, SCLK, from the master SPI module. After a full data word enters the shift register of a slave SPI, it transfers to the SPI Data Receive register, and the SPRF bit is set. To prevent an overflow condition, slave software then must read the receive data register before another full data word enters the shift register.

The maximum frequency of the SCLK for an SPI configured as a slave is less than 1/2 the bus clock frequency. The frequency of the SCLK for an SPI configured as a slave does not have to correspond to any SPI baud rate as defined by the SPR bits. The SPR bits control only the speed of the SCLK generated by an SPI configured as a master.

When the master SPI starts a transaction, the data in the slave shift register begins shifting out on the MISO pin. The slave can load its shift register with new data for the next transaction by writing to its transmit data register. The slave must write to its transmit data register at least one bus cycle before the master starts the next transaction. Otherwise, the data that was last transmitted is reloaded into the slave shift register and shifts out on the MISO pin again. Data written to the slave shift register during a transaction remains in a buffer until the end of the transaction.

When the clock phase bit (CPHA) is set, the first edge of SCLK starts a transaction. When CPHA is clear, the falling edge of $\overline{SS}$ starts a transaction.

Note

SCLK must be in the proper idle state before the slave is enabled to preserve the proper SCLK, MISO, MOSI timing relationships.

30.4.1.3 DMA Mode

When the TDMAEN or RDMAEN bit is set to 1, the SPI operates in DMA mode. Normal SPTE and/or SPRF interrupts are suppressed. Instead, DMA requests are generated, signaling the DMA controller to read and write the SPDRR or SPDTR as needed. The Transmitter Write DMA request is set whenever there is an open location in the transmit FIFO. Similarly, the Receiver Read DMA request is set whenever the receive FIFO is not empty. The DMA requests are suppressed in the case of an OVRF or MODF interrupt, as shown in Figure 30-3. If the SPI is being used to send and receive data, both TDMAEN and RDMAEN should be enabled or disabled at the same time. If data is only being received or transmitted, TDMAEN or RDMAEN may remain inactive as appropriate. However, in this case, ERRIE must not be set to prevent the DMA request being masked by a mode-fault error or overflow error.

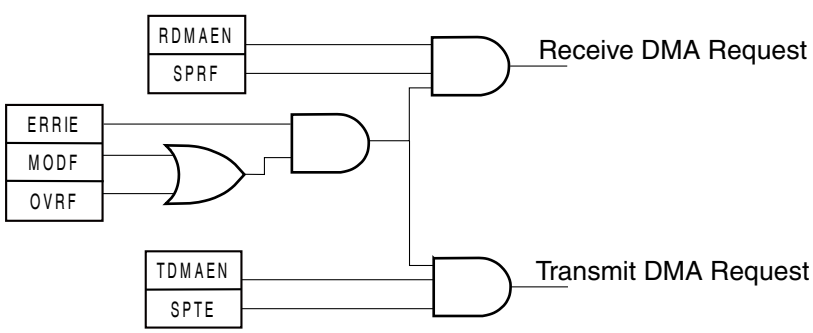


Figure 30-3. SPI DMA Request Generation

30.4.1.4 Wired-OR Mode

Wired-OR functionality is provided to permit the connection of multiple SPIs. The following figure illustrates the sharing of a single master device between multiple slave SPIs. When the WOM bit is set, the outputs switch from conventional complementary CMOS output to open drain outputs.

This internal pullup resistor brings the line high, and whichever SPI drives the line pulls it low as needed.

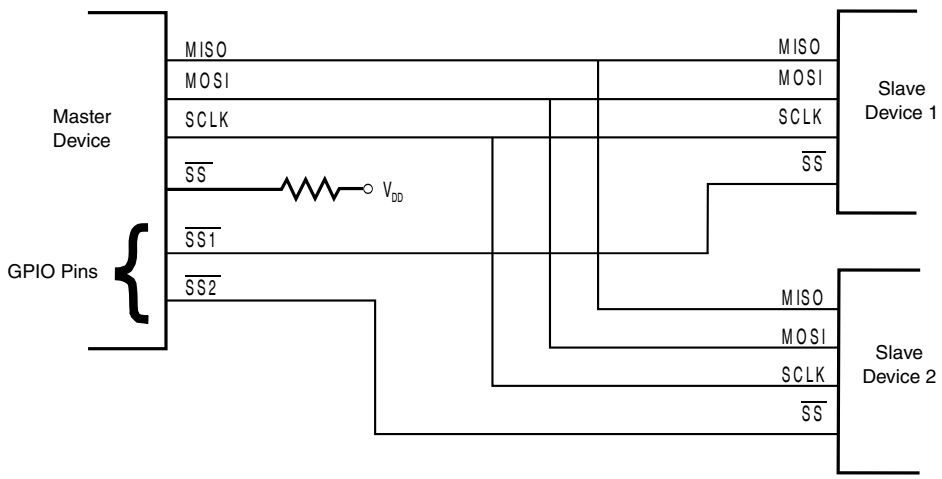


Figure 30-4. Master With Two Slaves

30.4.2 Transaction Formats

During an SPI transaction, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock synchronizes shifting and sampling on the two serial data lines. A slave select line enables selection of an individual slave SPI device. Slave devices that are not selected do not interfere with SPI bus activities. On a master SPI device, the slave select line can optionally be used to indicate multiple-master bus contention.

30.4.2.1 Data Transaction Length

The SPI can support data lengths of 2 to 16 bits. The length can be configured in the SPI Data Size and Control register. When the data length is less than 16 bits, the receive data register pads the upper bits with zeros.

Note

Data can be lost if the data length is not the same for both master and slave devices.

30.4.2.2 Data Shift Ordering

The SPI can be configured to transmit or receive the MSB of the desired data first or last, using the DSO bit in the SPI Status and Control register. Regardless of which bit is transmitted or received first, the data is always written to the SPI Data Transmit register and read from the SPI Data Receive register with the LSB in bit 0 and the MSB in correct position depending on the data transaction size.

30.4.2.3 Clock Phase and Polarity Controls

Software can select any of four combinations of serial clock (SCLK) phase and polarity using two bits in the SPI Status and Control register. The clock polarity is specified by the CPOL control bit, which selects an active high or low clock and has no significant effect on the transaction format.

The clock phase (CPHA) control bit selects one of two fundamentally different transaction formats. The clock phase and polarity should be identical for the master SPI device and the communicating slave device. In some cases, the phase and polarity are changed between transactions, to enable a master device to communicate with peripheral slaves having different requirements.

Note

Before writing to the CPOL bit or the CPHA bit, disable the SPI (by clearing the SPI enable bit (SPE)).

30.4.2.4 Transaction Format When CPHA = 0

The following figure shows an SPI transaction in which CPHA is logic zero. The figure should not be used as a replacement for data sheet parametric information. It assumes 16 bit data lengths and the MSB shifted out first.

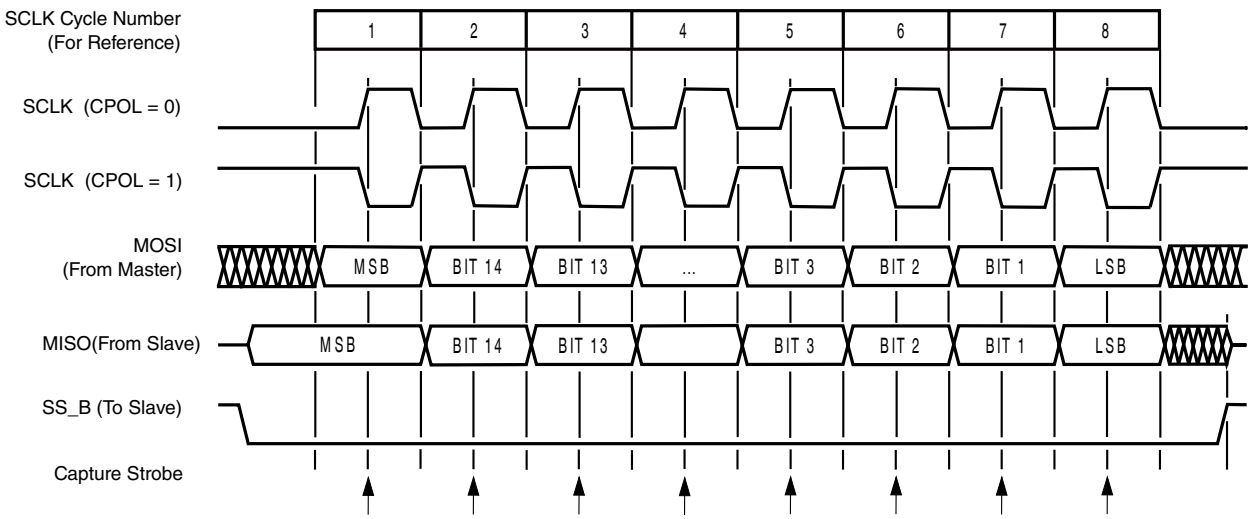


Figure 30-5. Transaction Format (CPHA = 0)

Two waveforms are shown for SCLK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCLK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. When CPHA = 0, the first SCLK edge is the MSB capture strobe. Therefore, the slave must begin driving its data before the first SCLK edge, and a falling edge on the $\overline{SS}$ pin is used to start the slave data transaction. The slave $\overline{SS}$ pin must be toggled back to high and then low again between each data word transmitted, as shown in the following figure.

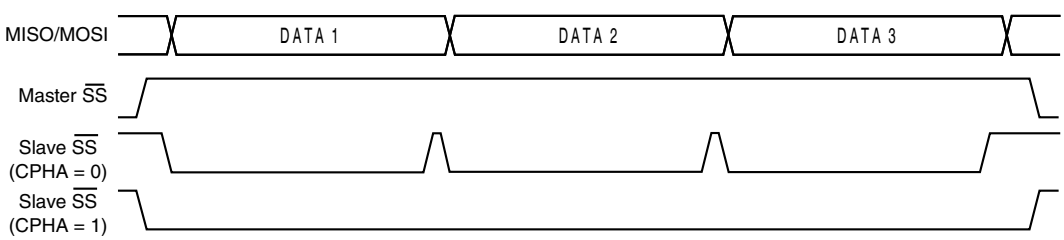


Figure 30-6. CPHA /$\overline{SS}$ Timing

When CPHA = 0 for a slave, the falling edge of $\overline{SS}$ indicates the beginning of the transaction. This causes the SPI to leave its idle state and begin driving the MISO pin with the first bit of its data. After the transaction begins, no new data is allowed into the shift register from the transmit data register. Therefore, the SPI data register of the slave must be loaded with transmit data before the falling edge of $\overline{SS}$. Any data written after the falling edge is stored in the transmit data register and transferred to the shift register after the current transaction. Also, for correct operation of the slave, SPE must be active before the negative edge of $\overline{SS}$ to correctly send/receive the first word. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic zero, so that only the selected slave drives to the master.

When CPHA = 0 for a master, normal operation would begin by the master initializing the $\overline{SS}$ pin of the slave high. A transfer would then begin by the master setting the $\overline{SS}$ pin of the slave low and then writing the SPI Data Transmit register. After a data transfer completes, the master device puts the $\overline{SS}$ pin back into the high state. While MODFEN = 1, the $\overline{SS}$ pin of the master must be high or a mode fault error occurs. If MODFEN = 0, the state of the $\overline{SS}$ pin is ignored.

30.4.2.5 Transaction Format When CPHA = 1

The following figure shows an SPI transaction in which CPHA is logic one. The figure should not be used as a replacement for data sheet parametric information. It assumes 16 bit data lengths and the MSB shifted out first.

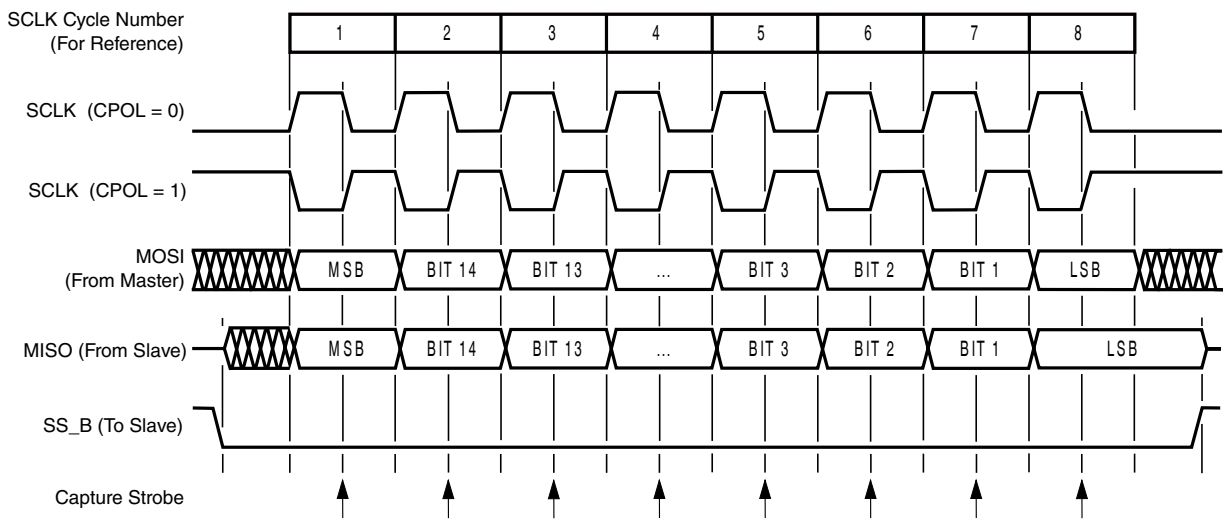


Figure 30-7. Transaction Format (CPHA = 1)

Two waveforms are shown for SCLK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCLK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master.

When CPHA = 1 for a slave, the first edge of the SCLK indicates the beginning of the transaction. This causes the SPI to leave its idle state and begin driving the MISO pin with the first bit of its data. After the transaction begins, no new data is allowed into the shift register from the transmit data register. Therefore, the SPI data register of the slave must be loaded with transmit data before the first edge of SCLK. Any data written after the first edge is stored in the transmit data register and transferred to the shift register after the current transaction. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic zero, so that only the selected slave drives to the master.

When CPHA = 1 for a master, the MOSI pin begins being driven with new data on the first SCLK edge. If MODFEN = 0 the $\overline{SS}$ pin of the master is ignored. Otherwise, the $\overline{SS}$ pin of the master must be high or a mode fault error occurs. The $\overline{SS}$ pin can remain low between transactions. This format may be preferable in systems with only one master and one slave driving the MISO data line.

30.4.2.6 Transaction Initiation Latency

When the SPI is configured as a master (SPMSTR is 1), writing to the SPI Data Transmit register starts a transaction. CPHA has no effect on the delay to the start of the transaction, but it does affect the initial state of the SCLK signal. When CPHA = 0, the SCLK signal remains inactive for the first half of the first SCLK cycle. When CPHA = 1, the first SCLK cycle begins with an edge on the SCLK line from its inactive to its active level. The SPI clock rate (selected by SPR2, SPR1, and SPR0) affects the delay from the write to the SPI Data Transmit register and the start of the SPI transaction. The internal baud clock in the master is a derivative of the internal device clock. To conserve power, it is enabled only after the SPMSTR bit is set and there is a new word written to the SPI Data Transmit register. If the SPI Data Transmit register has no new word when the current transaction completes, the internal baud clock is stopped. The initiation delay is a single SPI bit time, as the following figure shows. That is, the delay is 4 bus cycles for DIV4, 8 bus cycles for DIV8, 16 bus cycles for DIV16, 32 bus cycles for DIV32, and so on.

Note

The following figure assumes 16-bit data lengths and the MSB shifted out first.

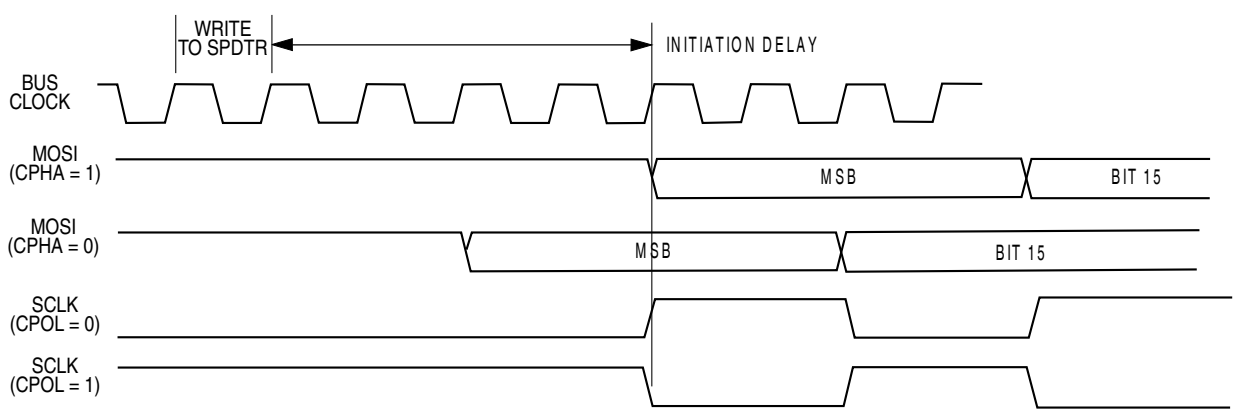


Figure 30-8. Transaction Start Delay (Master)

30.4.2.7 $\overline{SS}$ Hardware-Generated Timing in Master Mode

If the SSB_STRB bit is set in master mode, the SPI generates a word strobe pulse on $\overline{SS}$ for a slave device (see the following figures).

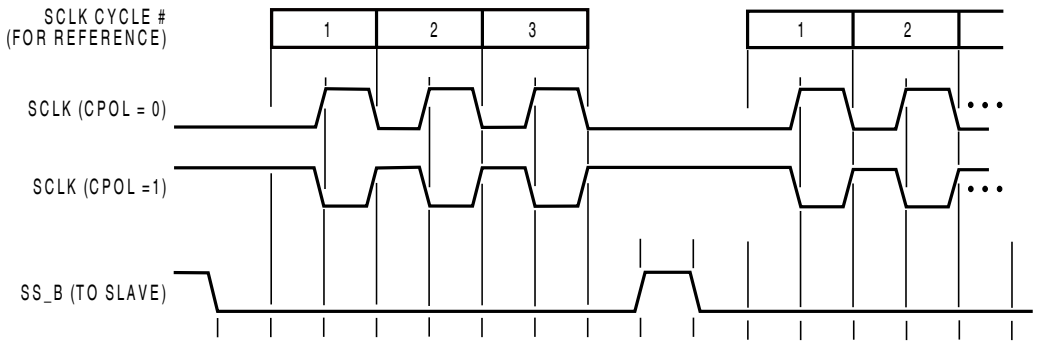


Figure 30-9. $\overline{SS}$ Strobe Timing (CPHA = 0)

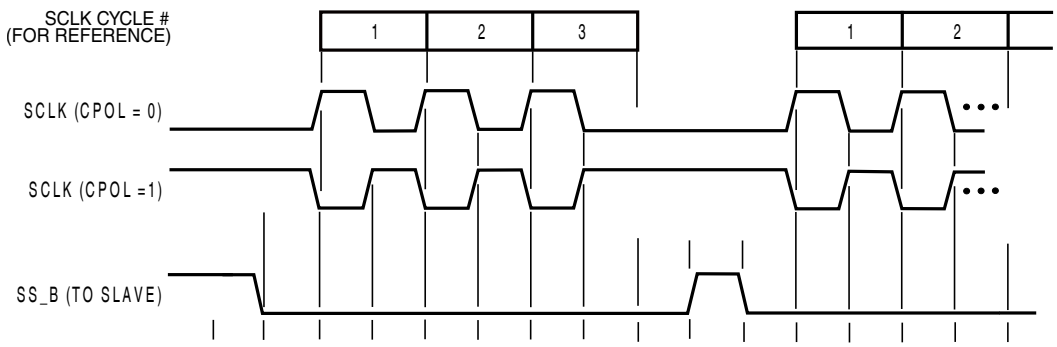


Figure 30-10. $\overline{SS}$ Strobe Timing (CPHA = 1)

If the SSB_AUTO bit is set in master mode, the SPI generates the initial falling edge and the final rising edge of $\overline{SS}$ for a slave device. The $\overline{SS}$ output has a falling edge one bit time before the first edge of SCLK (see the following figure).

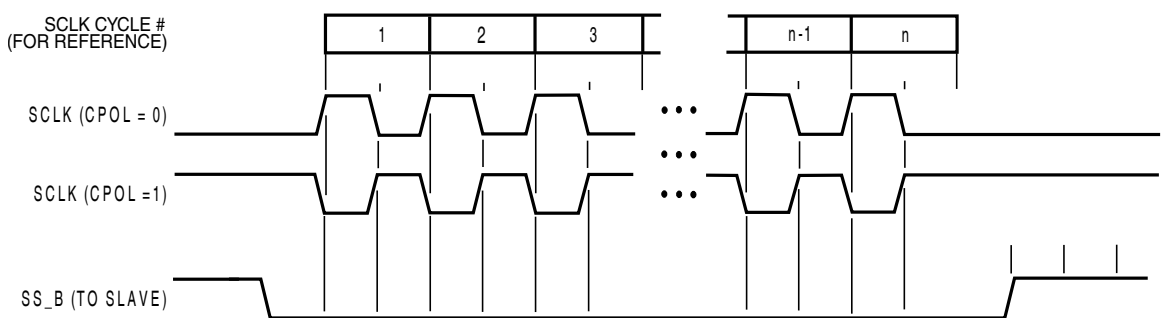


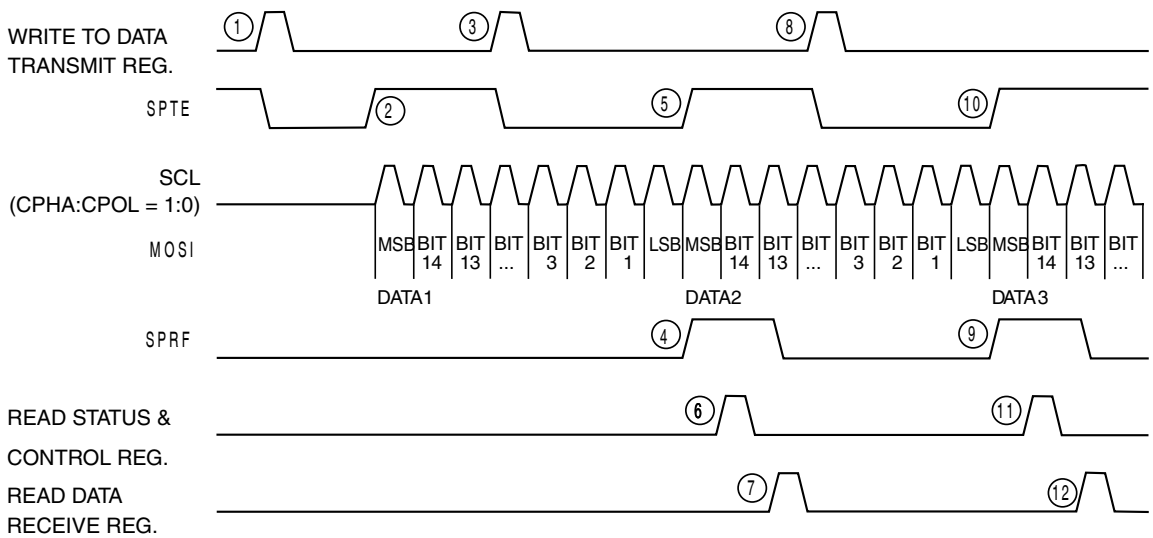
Figure 30-11. $\overline{SS}$ Auto Timing (CPHA = 1)

30.4.3 Transmission Data

The double-buffered data transmit register enables data to be queued and transmitted. For an SPI configured as a master, the queued data is transmitted immediately after the previous transaction has completed. The SPI transmitter empty flag (SPTE) indicates when the transmit data buffer is ready to accept new data. Write to the data transmit register only when the SPTE bit is high. The following figure shows the timing associated with doing back-to-back transactions with the SPI (SCLK has CPHA = 1; CPOL = 0).

Note

The following figure assumes 16-bit data lengths and the MSB shifted out first.



1. Device writes data 1 to Data Transmit register, clearing the SPTE bit.
2. Data 1 transfers from the Data Transmit register to the shift register, setting the SPTE bit.
3. Device writes data 2 to Data Transmit register, queueing data 2 and clearing the SPTE bit.
4. First incoming word transfers from shift register to the Data Receive register, settng the SPRF bit.
5. Data 2 transfers from the Data Transmit register to the shift register, setting the SPTE bit.
6. Device reads Status and Control register with the SPRF bit set.
7. Device reads Data Receive register, clearing the SPRF bit.
8. Device writes data 3 to Data Transmit register, queueing data 3 and clearing the SPTE bit.
9. The second incoming data transfers from the shift register to the Receive Data Register, setting the SPRF bit.
10. Data 3 transfers from the Data Transmit register to the shift register, setting the SPTE bit.
11. Device reads Status and Control register with the SPRF bit set.
12. Device reads Data Receive register, clearing the SPRF bit.

Figure 30-12. SPRF/SPTE Interrupt Timing

The transmit data buffer enables back-to-back transactions without the slave precisely timing its writes between transactions, as occurs in a system with a single data buffer. Also, in slave mode, if no new data is written to the SPI Data Transmit register, the last value contained in the SPI Data Transmit register is retransmitted if the external master starts a new transaction.

For an idle master that has no data loaded into its transmit buffer and no word currently being transmitted, the SPTE is set again no more than two bus cycles after the SPI Data Transmit register is written. This enables the user to queue up at most a 32-bit value to send. For an SPI operating in slave mode, the load of the shift register is controlled by the external master, and back-to-back writes to the transmit data register are not possible. The SPTE bit indicates when the next write can occur.

30.4.4 Error Conditions

The following flags signal SPI error conditions:

- Overflow (OVRF) — Failing to read the SPI Data Receive register before the next data word finishes entering the shift register sets the OVRF bit. The new data word does not transfer to the Receive Data register, and the unread data word can still be read. OVRF is in the SPI Status and Control register.
- Mode fault error (MODF) — The MODF bit indicates that the voltage on the slave select pin ($\overline{SS}$) is inconsistent with the mode of the SPI. MODF is in the SPI Status and Control register.

30.4.4.1 Overflow Error

The overflow flag (OVRF) is set if the SPI Receive Data register still has unread data from a previous transaction when the capture strobe of bit 1 of the next transaction occurs. The bit 1 capture strobe occurs in the middle of SCLK when the data length equals transaction data length minus 1. If an overflow occurs, all data received after the overflow and before the OVRF bit is cleared does not transfer to the SPI Receive Data register and does not set the SPI receiver full bit (SPRF). The unread data that is transferred to the SPI Receive Data register before the overflow occurred can still be read. Therefore, an overflow error always indicates the loss of data. Clear the overflow flag by reading the SPI Status and Control register and then reading the SPI Receive Data register.

OVRF generates a receiver/error interrupt request if the error interrupt enable bit (ERRIE) is also set. It is not possible to enable MODF or OVRF individually to generate a receiver/error interrupt request. However, leaving MODFEN low prevents MODF from being set.

If the SPRF interrupt is enabled and the ERRIE is not enabled, poll the OVRF bit to detect an overflow condition. The following figure shows how it is possible to miss an overflow. The first part of the figure shows how it is possible to read the SPI Status and Control register and the SPI Data Receive register to clear the SPRF without problems. However, as illustrated by the second transaction example, the OVRF bit can be set in between the time that the SPI Status and Control register and the SPI Data Receive register are read.

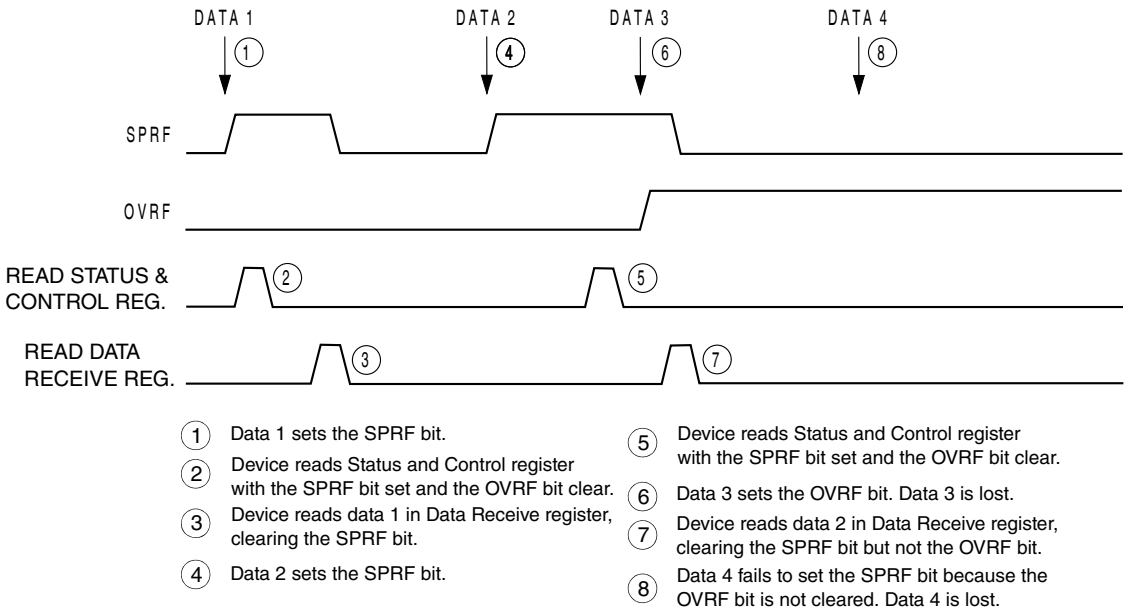


Figure 30-13. Missed Read of Overflow Condition

In this case, an overflow can easily be missed. Since no more SPRF interrupts can be generated until this OVRF is serviced, it is not obvious that data is being lost as more transactions are completed. To prevent this, either enable the OVRF interrupt or do another read of the SPI Status and Control register following the read of the SPI Data Receive register. This ensures that the OVRF was not set before the SPRF was cleared and that future transactions can set the SPRF bit. The following figure illustrates this process. Generally, to avoid this second read of the SPI Status and Control register, enable the OVRF to the device by setting the ERRIE bit.

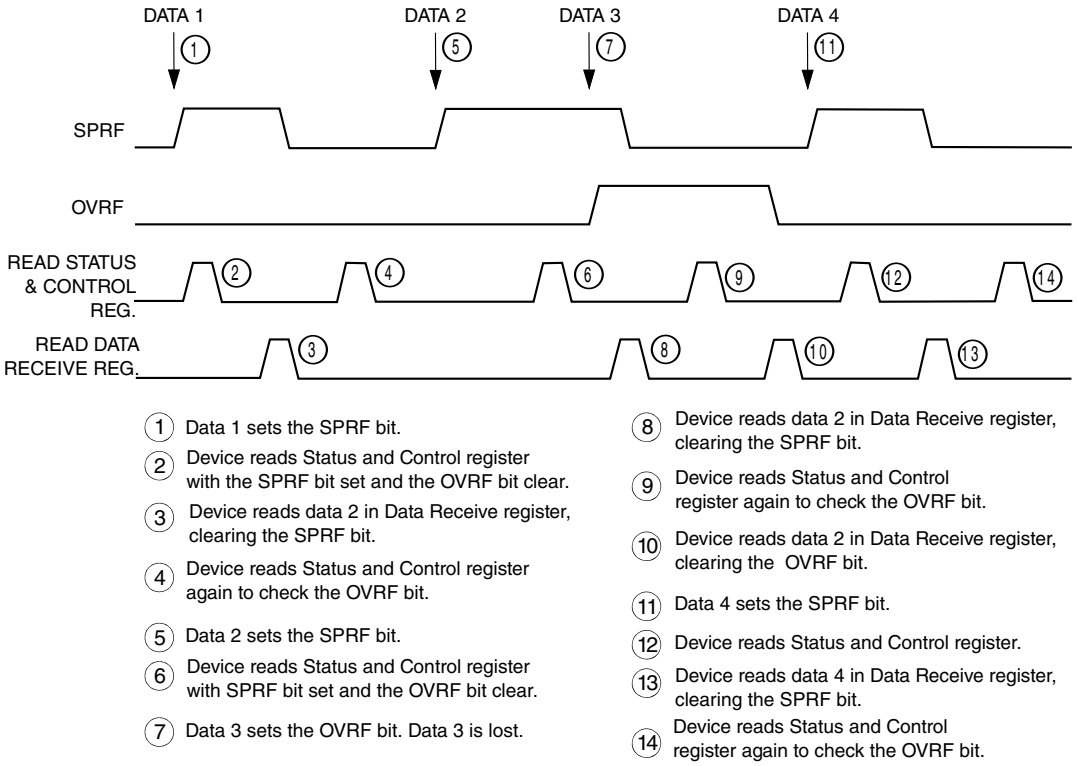


Figure 30-14. Clearing SPRF When OVRF Interrupt Is Not Enabled

30.4.4.2 Mode Fault Error

Setting the SPMSTR bit selects master mode and configures the SCLK and MOSI pins as outputs and the MISO pin as an input. Clearing SPMSTR selects slave mode and configures the SCLK and MOSI pins as inputs and the MISO pin as an output. The mode fault bit, MODF, is set any time the state of the slave select pin, $\overline{SS}$, is inconsistent with the mode selected by SPMSTR. To prevent SPI pin contention and damage to the device, a mode fault error occurs if:

- The $\overline{SS}$ pin of a slave SPI goes high during a transaction.
- The $\overline{SS}$ pin of a master SPI goes low at any time.

For the MODF flag to be set, the mode fault error enable bit (MODFEN) must be set. Clearing the MODFEN bit does not clear the MODF flag but does prevent MODF from being set again after MODF is cleared.

MODF generates a receiver/error interrupt request if the error interrupt enable bit (ERRIE) is also set. It is not possible to enable MODF or OVRF individually to generate a receiver/error interrupt request. However, leaving MODFEN low prevents MODF from being set.

30.4.4.2.1 Master Mode Fault

In a master SPI with the mode fault enable bit (MODFEN) set, the mode fault flag (MODF) is set if $\overline{SS}$ goes to logic zero. A mode fault in a master SPI causes the following events to occur:

- If ERRIE = 1, the SPI generates an SPI receiver/error interrupt request.
- The SPE bit is cleared (SPI disabled).
- The SPTE bit is set.
- The SPI state counter is cleared.

Note

Setting the MODF flag does not clear the SPMSTR bit. The SPMSTR bit has no function when SPE = 0. Reading SPMSTR when MODF = 1 shows the difference between a MODF occurring when the SPI is a master and when it is a slave.

In a master SPI, the MODF flag is not cleared until the $\overline{SS}$ pin is at a logic one or the SPI is configured as a slave.

30.4.4.2.2 Slave Mode Fault

When configured as a slave (SPMSTR = 0), the MODF flag is set if the $\overline{SS}$ pin goes high during a transaction. When CPHA = 0, a transaction begins when $\overline{SS}$ goes low and ends after the incoming SCLK goes back to its idle level following the shift of the last data bit. When CPHA = 1, the transaction begins when the SCLK leaves its idle level and $\overline{SS}$ is already low. The transaction continues until the SCLK returns to its idle level following the shift of the last data bit.

In a slave SPI (SPMSTR = 0), the MODF bit generates an SPI receiver/error interrupt request if the ERRIE bit is set. The MODF bit does not clear the SPE bit or reset the SPI in any way. Software can abort the SPI transaction by clearing the SPE bit of the slave.

Note

A logic one voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high impedance state. Also, the slave SPI ignores all incoming SCLK clocks, even if it was already in the middle of a transaction. A mode fault occurs if the $\overline{SS}$ pin changes state during a transaction.

When CPHA = 0, a MODF occurs if a slave is selected ($\overline{SS}$ is at logic 0) and later unselected ($\overline{SS}$ is at logic 1) after the first bit of data has been received (SCLK is toggled at least once). This happens because $\overline{SS}$ at logic 0 indicates the start of the transaction (MISO driven out with the value of MSB) for CPHA = 0. When CPHA = 1, a slave can be selected and then later unselected with no transaction occurring. Therefore, MODF does not occur because a transaction was never begun.

To clear the MODF flag, write a one to the MODF bit in the SPI Status and Control register. If the MODF flag is not cleared by writing a one to the MODF bit, the condition causing the mode fault still exists. In this case, the interrupt caused by the MODF flag can be cleared by disabling the EERIE bit or MODFEN bit (if set) or by disabling the SPI.

30.4.5 Resetting the SPI

Any system reset completely resets the SPI. Partial resets occur whenever the SPI enable bit (SPE) is low. Whenever SPE is low, the following occurs:

1. The SPTE flag is set.
2. Any slave mode transaction currently in progress is aborted.
3. Any master mode transaction currently in progress continues to completion.
4. The SPI state counter is cleared, making it ready for a new complete transaction.
5. All the SPI port logic is disabled.

Items 4 and 5 occur after 2 in slave mode, or after 3 in master mode.

The following items are reset only by a system reset:

- The SPI Data Transmit and SPI Data Receive registers
- All control bits in the SPI Status and Control register and the SPI Data Size and Control register
- The status flags SPRF, OVRF, and MODF

By not resetting the control bits when SPE is low, the user can clear SPE between transactions without having to set all control bits again when SPE is set back high for the next transaction.

By not resetting the SPRF, OVRF, and MODF flags, the user can still service these interrupts after the SPI is disabled. The user can disable the SPI by writing 0 to the SPE bit. The SPI is also disabled when a mode fault occurs in an SPI configured as a master.

30.5 Interrupts

Four SPI status flags can be enabled to generate device interrupt requests.

Table 30-3. SPI Interrupts

Flag	Interrupt Enabled By	Description
SPTE (Transmitter Empty)	SPI Enable / SPI Transmitter Interrupt Enable (SPTIE = 1, SPE = 1)	The SPI transmitter interrupt enable bit (SPTIE) enables the SPI transmitter empty (SPTE) flag or TFWM to generate transmitter interrupt requests, provided that the SPI is enabled (SPE = 1). The SPTE bit becomes set every time data transfers from the SPI Data Transmit register to the shift register and there is no more new data available in the TX queue. The clearing mechanism for the SPTE flag is a write to the SPI Data Transmit register.
SPRF (Receiver Full)	SPI Receiver Interrupt Enable (SPRIE = 1, SPE = 1)	The SPI receiver interrupt enable bit (SPRIE) enables the SPI receiver full (SPRF) bit or RFWM to generate receiver interrupt requests. The SPRF is set every time data transfers from the shift register to the SPI Data Receive register and there is no more room available in the RX queue to receive new data. The clearing mechanism for the SPRF flag is to read the SPI Data Receive register.
OVRF (Overflow)	SPI Receiver/Error Interrupt Enable (ERRIE = 1)	The error interrupt enable bit (ERRIE) enables both the MODF and OVRF bits to generate a receiver/error interrupt request.
MODF (Mode Fault)	SPI Receiver/Error Interrupt Enable (ERRIE = 1)	The mode fault enable bit (MODEFEN) enables the mode fault (MODF) bit to be set. The MODF bit allows the receiver/error interrupt request regardless of the state of the SPE bit as long as the interrupt enable (ERRIE) is set. The mode fault enable bit (MODFEN) can prevent the MODF flag from being set, so that only the OVRF bit is enabled by the ERRIE bit to generate receiver/error device interrupt requests.

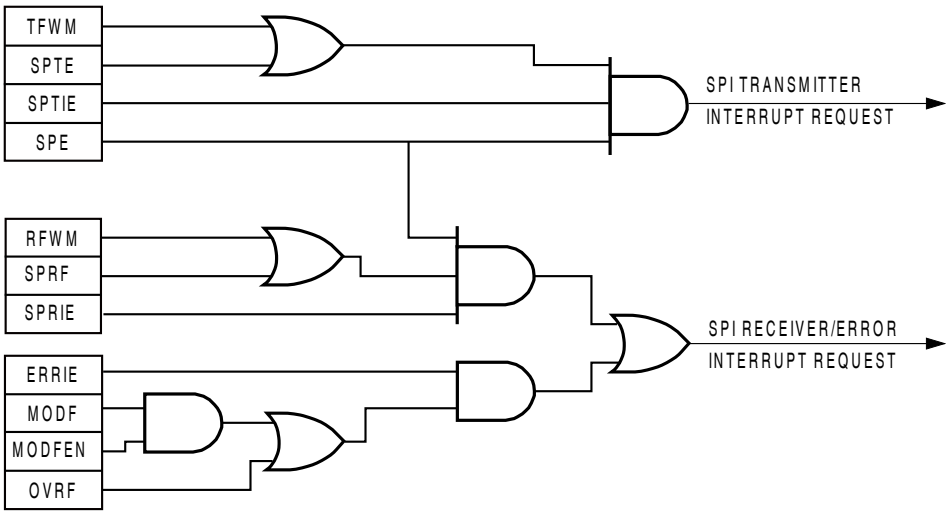


Figure 30-15. SPI Interrupt Request Generation

Chapter 31
Low Power Inter-Integrated Circuit (LPI2C)

31.1 Chip-specific information for this module

31.1.1 HREQ signal in this device

In this device, the LPI2C input trigger and Host Request (HREQ) signal are from XBAR. See XBAR_OUT64 and XBAR_OUT65 in XBARA Outputs for more details. Therefore, MCFGR0[HRSEL] (logic value either 0 or 1) selects the same signal from XBAR.

31.2 Introduction

The LPI2C is a low power Inter-Integrated Circuit (I2C) module that supports an efficient interface to an I^2C bus as a master and/or as a slave.

- The LPI2C implements logic support for standard-mode, fast-mode, fast-mode plus and ultra-fast modes of operation.
- The LPI2C is designed to use little CPU overhead, with DMA offloading of FIFO register accesses.
- The LPI2C can continue operating in stop modes if an appropriate clock is available.

The LPI2C module also complies with the System Management Bus (SMBus) Specification, version 2. The SMBus is a single-ended simple two-wire bus, which is typically used for low bandwidth communications.

NOTE

The I^2C (Inter-Integrated Circuit) serial bus is multi-master, multi-slave, packet-switched, and single-ended, and is often used to attach microcontroller ICs to lower-speed peripheral ICs.

31.2.1 Features

The LPI2C supports:

- Standard, Fast, Fast+ and Ultra Fast modes are supported
- High speed mode (HS) in slave mode
- High speed mode (HS) in master mode, if SCL pin implements current source pull-up (device-specific)
- Multi-master support, including synchronization and arbitration. Multi-master means any number of master nodes can be present. Additionally, master and slave roles may be changed between messages (after a STOP is sent).
- Clock stretching: Sometimes multiple I2C nodes may be driving the lines at the same time. If any I2C node is driving a line low, then that line will be low. I2C nodes that are starting to transmit a logical one (by letting the line float high) can detect that the line is low, and thereby know that another I2C node is active at the same time.
  - When node detection is used on the SCL line, it is called *clock stretching*, and clock stretching is used as a I2C flow control mechanism for multiple laves.
  - When node detection is used on the SDA line, it is called *arbitration*, and arbitration ensures that there is only one I2C node transmitter at a time.
- General call, 7-bit and 10-bit addressing
- Software reset, START byte and Device ID (also require software support)

The LPI2C master supports:

- Command/transmit FIFO of 4words.
- Receive FIFO of 4words.
- Command FIFO will wait for idle I2C bus before initiating transfer
- Command FIFO can initiate (repeated) START and STOP conditions and one or more master-receiver transfers
- STOP condition can be generated from command FIFO, or generated automatically when the transmit FIFO is empty
- Host request input to control the start time of an I2C bus transfer
- Flexible receive data match can generate interrupt on data match and/or discard unwanted data
- Flag and optional interrupt to signal Repeated START condition, STOP condition, loss of arbitration, unexpected NACK, and command word errors
- Supports configurable bus idle timeout and pin-stuck-low timeout

The LPI2C slave supports:

- Separate I2C slave registers to minimize software overhead because of master/slave switching

- Support for 7-bit or 10-bit addressing, address range, SMBus alert and general call address
- Transmit data register that supports interrupt or DMA requests
- Receive data register that supports interrupt or DMA requests
- Software-controllable ACK or NACK, with optional clock stretching on ACK/NACK bit
- Configurable clock stretching, to avoid transmit FIFO underrun and receive FIFO overrun errors
- Flag and optional interrupt at end of packet, STOP condition, or bit error detection

31.2.2 Block Diagram

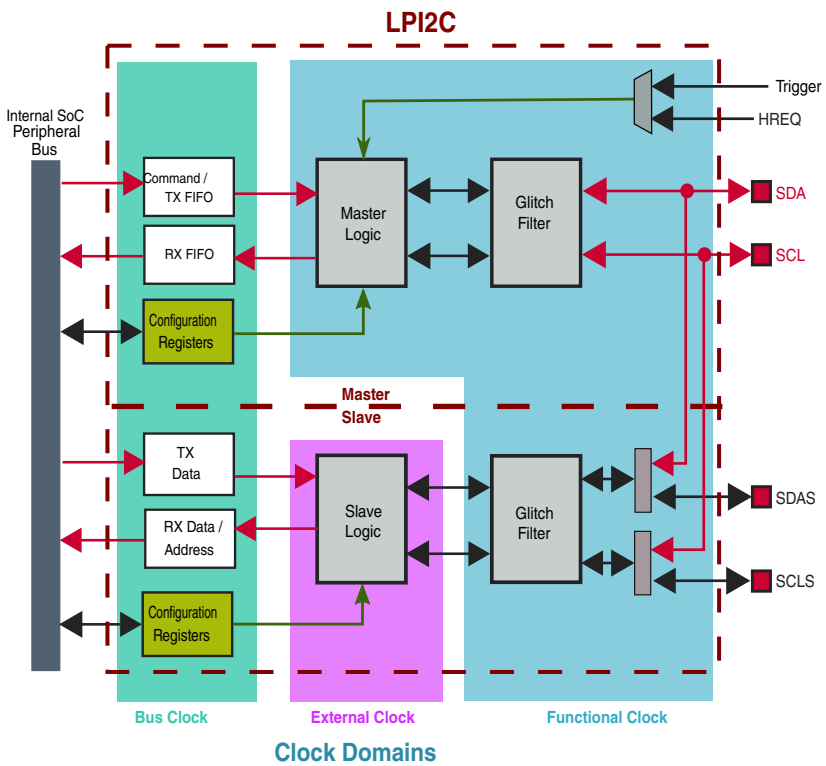


Figure 31-1. LPI2C block diagram

31.2.3 Modes of operation

Table 31-1. Chip modes supported by the LPI2C module

Chip mode	LPI2C Operation
Run	Normal operations
Stop	Can continue operating in stop mode if the Doze Enable bit (MCR[DOZEN]) is clear and the LPI2C is using an external or internal clock source that remains operating during stop mode.
Low Power Stop (also called Low Leakage Stop or LLS)	Before entering low power stop mode, the LPI2C will wait for the current transfer to finish any pending operation, while temporarily ignoring The Doze Enable (MCR[DOZEN]) bit.
Debug	Can continue operating in debug mode if the Debug Enable bit (MCR[DBGEN]) is set.

31.2.4 Signal Descriptions

Table 31-2. Signals

Signal	Name	2-Wire Scheme	4-Wire Scheme	I/O
SCL	LPI2C clock line	SCL	In 4-wire mode, this is the SCL input pin.	I/O
SDA	LPI2C data line	SDA	In 4-wire mode, this is the SDA input pin.	I/O
SCLS	Secondary I2C clock line	Not used	In 4-wire mode, this is the SCLS output pin. If LPI2C master/slave are configured to use separate pins, then this the LPI2C slave SCL pin.	I/O
SDAS	Secondary I2C data line	Not used	In 4-wire mode, this is the SDAS output pin. If LPI2C master/slave are configured to use separate pins, then this the LPI2C slave SDA pin.	I/O

31.3 Functional description

31.3.1 Clocking and Resets

For device-specific clocking information, refer to the Clocking chapter.

Table 31-3. Clocks

LPI2C Functional clock	The LPI2C functional clock is asynchronous to the bus clock and can remain enabled in low power modes to support I2C bus transfers by the LPI2C master. The functional clock is also used by the LPI2C slave to support digital filter and data hold time configurations. The LPI2C master divides the

Table continues on the next page...

Table 31-3. Clocks (continued)

	functional clock by a prescaler and the resulting frequency must be at least 8 times faster than the I2C bus bandwidth.
External clock	The LPI2C slave logic is clocked directly from the external pins SCL and SDA (or SCLS and SDAS if master and slave are implemented on separate pins). This allows the LPI2C slave to remain operational, even when the LPI2C functional clock is disabled. **NOTE:** The LPI2C slave digital filter must be disabled if the LPI2C functional clock is disabled, and this can affect compliance with some of the timing parameters of the I2C specification, such as the data hold time.
Bus clock	The bus clock is only used for bus accesses to the control and configuration registers. The bus clock frequency must be sufficient to support the data bandwidth requirements of the LPI2C master and slave registers.

Table 31-4. Resets

Chip reset	The logic and registers for the LPI2C master and slave are reset to their default state on a chip reset.
Software reset	• The LPI2C master implements a software reset bit in its Control Register. The MCR[RST] will reset all master logic and registers to their default state, except for the MCR itself. • The LPI2C slave implements a software reset bit in its Control Register. The SCR[RST] will reset all slave logic and registers to their default state, except for the SCR itself.
FIFO reset	• The LPI2C master implements write-only control bits that reset the transmit FIFO (MCR[RTF]) and receive FIFO (MCR[RRF]). After a FIFO is reset, that FIFO is empty. • The LPI2C slave implements write-only control bits that reset the transmit data register (SCR[RTF] and receive data register (SCR[RRF]). After a data register is reset, that data register is empty.

31.3.2 Master Mode

The LPI2C master logic operates independently from the slave logic to perform all master mode transfers on the I2C bus.

31.3.2.1 Transmit and Command FIFO commands

The transmit FIFO stores command data to initiate the various I2C operations. The following operations can be initiated through commands in the transmit FIFO:

- START or Repeated START condition with address byte and expecting ACK or NACK.
- Transmit data (this is the default for zero extended byte writes to the transmit FIFO).
- Receive 1-256 bytes of data (can also be configured to discard receive data and not store in receive FIFO).
- STOP condition (can also be configured to send STOP condition when transmit FIFO is empty).

Multiple transmit and receive commands can be inserted between the START condition and STOP conditon; transmit and receive commands must not be interleaved (to comply with the I2C specification). The receive data command and the receive data and discard commands can be interleaved, to ensure that only the desired received data is stored in the receive FIFO (or compared with the data match logic).

The LPI2C master will automatically transmit a NACK on the last byte of a receive data command unless the next command in the FIFO is also a receive data command. A NACK is also automatically transmitted if the transmit FIFO is empty when a receive data command completes.

The LPI2C master supports 10-bit addressing through a (repeated) START condition, followed by a transmit data byte containing the second address byte, followed by any number of data bytes with the master-transmit data.

A START or Repeated START condition that is expecting a NACK (for example,HS-mode master code) must be followed by a STOP or (repeated) START condition.

31.3.2.2 Master operations

Whenever the LPI2C is enabled, it monitors the I2C bus to detect when the I2C bus is idle (MSR[BBF]). The I2C bus is no longer considered idle if either SCL or SDA are low, and the I2C bus becomes idle if a STOP condition is detected or if a bus idle timeout is detected (as configured by MCFGR2[BUSIDLE]). After the I2C bus is idle, the transmit FIFO is not empty, and the host request is either asserted or disabled, then the LPI2C master will initiate a transfer on the I2C bus. This involves the following steps:

- Wait the bus idle time equal to (MCCR0[CLKLO] + 1) multiplied by the prescaler (MCFGR1[PRESCALE]).
- Transmit a START condition and address byte using the timing configuration in the Master Clock Configuration Register 0/1 (MCCR0/1); if a high speed mode transfer is configured, then the timing configuration from Master Clock Configuration Register 2/3 (MCCR2/3) is used instead.
- Perform master-transmit or master-receive transfers, as configured by the transmit FIFO.
- Transmit NACK on the last byte of a master-receive transfer, unless the next command in the transmit FIFO is also a receive data command and the transmit FIFO is not empty.
- Transmit a Repeated START or STOP condition as configured by the transmit FIFO and/or MCFGR1[AUTOSTOP]. A repeated START can change which timing configuration register is used.

When the LPI2C master is disabled (either due to MCR[MEN] being clear or automatically due to mode entry), the LPI2C will continue to empty the transmit FIFO until a STOP condition is transmitted. However, the LPI2C will no longer stall the I2C bus waiting for the transmit or receive FIFO, and after the transmit FIFO is empty, the LPI2C will generate a STOP condition automatically.

The LPI2C master can stall the I2C bus under certain conditions; this will result in SCL pulled low continuously on the first bit of a byte, until the condition is removed:

- LPI2C master is enabled and busy, the transmit FIFO is empty, and MCFGR1[AUTOSTOP] is clear.
- LPI2C master is enabled and receiving data, receive data is not being discarded (due to command or receive data match), and the receive FIFO is full.

31.3.2.3 Receive FIFO and Data Matching

The receive FIFO is used to store receive data during master-receiver transfers. Receive data can also be configured to discard receive data instead of storing in the receive FIFO; this is configured by the command word in the transmit FIFO.

Receive data supports a receive data match function that can match received data against one of two bytes or against a masked data byte. The data match function can also be configured to compare only the first one or two received data words since the last (repeated) START condition. Receive data that is already discarded due to the command word cannot cause the data match to set, and will delay the match on the first received data word until after the discarded data is received. The receiver match function can also be configured to discard all receive data until a data match is detected, using the MCFGR0[RDMO] control bit. When clearing the MCFGR0[RDMO] control bit following a data match, clear MCFGR0[RDMO] before clearing MSR[DMF], to allow all subsequent data to be received.

31.3.2.4 Timing Parameters

The following timing parameters can be configured by the LPI2C master. Parameters are configured separately for high speed mode (MCCR2/3) and other modes (MCCR0/1). This allows the high speed mode master code to be sent using the regular timing parameters, and then switch to the high speed mode timing (following a repeated START) until the next STOP condition.

The LPI2C master timing parameters in LPI2C functional clock cycles are configured as follows. They must be configured to meet the I2C timing specification for the required mode.

Table 31-5. Timing Parameters

I2C Specification Timing Parameter	I2C Specification Timing Symbol	LPI2C Timing Parameter (LPI2C functional clock cycles)
SCL clock period	tSCL	(CLKHI + CLKLO + 2 + SCL_LATENCY) x (2 ^ PRESCALE)
hold time (repeated) START condition	tHD:STA	(SETHOLD +1) x (2 ^ PRESCALE)
LOW period of the SCL clock	tLOW	(CLKLO + 1) x (2 ^ PRESCALE)
HIGH period of the SCL clock	tHIGH	(CLKHI + 1 + SCL_LATENCY) x (2 ^ PRESCALE)
setup time for a repeated START condition or STOP condition	tSU:STA, tSU:STO	(SETHOLD + 1 + SCL_LATENCY) x (2 ^ PRESCALE)
data hold time	tHD:DAT	(DATAVD + 1) x (2 ^ PRESCALE)
data setup time	tSU:DAT	(SDA_LATENCY + 1) x (2 ^ PRESCALE)
bus free time between a STOP and START condition	tBUF	(CLKLO + 1 + SDA_LATENCY) x (2 ^ PRESCALE)
data valid time, data valid acknowledge time	tVD:DAT, tVD:ACK	(DATAVD + 1) x (2 ^ PRESCALE)

The latency parameters are defined in the following table, these parameters assume the risetime is less than one LPI2C functional clock cycle. The risetime depends on a number of factors, including the I/O propagation delay, the I2C bus loading and the external pull-up resistor sizing. A larger risetime will increase the number of cycles that the signal takes to propagate through the synchronizer (and glitch filter), which increases the latency.

Table 31-6. Synchronization Latency

Timing Parameter	Timing Definition
SCL_LATENCY	ROUNDDOWN ((2 + FILTSCL + SCL_RISETIME) / (2 ^ PRESCALE))
SDA_LATENCY	ROUNDDOWN ((2 + FILTSDA + SDA_RISETIME) / (2 ^ PRESCALE))

The following timing restrictions must be enforced to avoid unexpected START or STOP conditions on the I2C bus, or to avoid unexpected START or STOP conditions detected by the LPI2C master. The timing restrictions can be summarized as **SDA cannot change when SCL is high outside of a transmitted (repeated) START or STOP condition**.

Table 31-7. LPI2C Timing Parameter Restrictions

Timing Parameter	Minimum	Maximum	Comment
CLKLO	0x03	-	Also: CLKLO x (2 ^ PRESCALE) > SCL_LATENCY

Table continues on the next page...

Table 31-7. LPI2C Timing Parameter Restrictions (continued)

Timing Parameter	Minimum	Maximum	Comment
CLKHI	0x01	-	Configure CLKHI to meet the duty cycle requirements in the I2C specification
SETHOLD	0x02	-	Also: SETHOLD x (2 ^ PRESCALE) > SDA_LATENCY
DATAVD	0x01	CLKLO - SDA_LATENCY - 1	Configure DATAVD to meet the data hold requirement in the I2C specification
FILTSCL	0x00	[CLKLO x (2 ^ PRESCALE)] - 3	FILTSCL and FILTSDA are the only parameters not multiplied by (2 ^ PRESCALE)
FILTSDA	FILTSCL	[CLKLO x (2 ^ PRESCALE)] - 3	Configuring FILTSDA greater than FILTSCL can delay the SDA input to compensate for board level skew
BUSIDLE	(CLKLO+SETHOLD+2) x 2	-	Must also be greater than (CLKHI+1)

The timing parameters must be configured to meet the requirements of the I2C specification; this will depend on the mode being supported and the LPI2C functional clock frequency. When switching between two modes using the different clock configuration registers (for example, Fast and HS-mode), the PRESCALE factor must remain constant between the modes. Some example timing configurations are provided below.

Table 31-8. LPI2C Example Timing Configurations

I2C Mode	Clock Frequency	Baud Rate	PRESCALE	FILTSCL / FILTSDA	SETHOLD	CLKLO	CLKHI	DATAVD
Fast	8 MHz	400 kbps	0x0	0x0/0x0	0x04	0x0B	0x05	0x02
Fast+	8 MHz	1 Mbps	0x0	0x0/0x0	0x02	0x03	0x01	0x01
Fast	48 MHz	400 kbps	0x0	0x1/0x1	0x1D	0x3E	0x35	0x0F
Fast	48 MHz	400 kbps	0x2	0x1/0x1	0x07	0x11	0x0B	0x03
Fast+	48 MHz	1 Mbps	0x2	0x1/0x1	0x03	0x06	0x04	0x04
HS-mode	48 MHz	3.2 Mbps	0x0	0x0/0x0	0x07	0x08	0x03	0x01
Fast	60 MHz	400 kbps	0x1	0x2/0x2	0x11	0x28	0x1F	0x08
Fast+	60 MHz	1 Mbps	0x1	0x2/0x2	0x07	0x0F	0x0B	0x01
HS-mode	60 MHz	3.33 Mbps	0x1	0x0/0x0	0x04	0x04	0x02	0x01

31.3.2.5 Error Conditions

The LPI2C master will monitor for errors while it is *active*[1]. The following conditions will generate an error flag and block a new START condition from being sent, until the flag is cleared by software:

- A START or STOP condition is detected and is not generated by the LPI2C master (sets MSR[ALF]).
- Transmitting data on SDA and different values are being received (sets MSR[ALF]).
- NACK is detected when transmitting data, and MCFGR1[IGNACK] is clear (sets MSR[NDF]).
- NACK is detected and is expecting ACK for the address byte, and MCFGR1[IGNACK] is clear (sets MSR[NDF]).
- ACK is detected and is expecting NACK for the address byte, and MCFGR1[IGNACK] is clear (sets MSR[NDF]).
- Transmit FIFO is requesting to transmit or receive data without a START condition (sets MSR[FEF]).
- SCL (or SDA if MCFGR1[TIMECFG] is set) is low for (MCFGR2[TIMELOW] * 256) prescaler cycles without a pin transition (sets MSR[PLTF]).

Software must respond to the MSR[PTLF] flag to terminate the existing command either cleanly (by clearing MCR[MEN]), or abruptly (by setting MCR[RST]).

The MCFGR2[BUSIDLE] field can be used to force the I2C bus to be considered idle when SCL and SDA remain high for (BUSIDLE+1) prescaler cycles. The I2C bus is normally considered idle when the LPI2C master is first enabled, but when BUSIDLE is configured greater than zero then SCL and/or SDA must be high for (BUSIDLE+1) prescaler cycles before the I2C bus is first considered idle.

31.3.2.6 Pin Configuration

- **Open-drain support:** The LPI2C master defaults to open-drain configuration of the SDA and SCL pins. Support for true open drain depends on the specific device, and requires the pins where LPI2C pins are muxed to support true open drain.
- **High Speed mode support:** Support for high speed mode also depends on the specific device, and requires the SCL pin to support the current source pull-up required in the I2C specification.
- **Ultra-Fast mode support:** The LPI2C master also supports the output-only push-pull function required for I2C ultra-fast mode using the SDA and SCL pins. Support for ultra-fast mode also requires the IGNACK bit to be set.

1. If the MTDR is empty, then LPI2C master is idle so it is not monitoring for errors.

- **Push-pull 2-wire support:** A push-pull 2-wire configuration is also available to the LPI2C master that may support a partial high speed mode, if the LPI2C is the only master and all I2C pins on the bus are at the same voltage. This will configure the SCL pin as push-pull for every clock except the 9th clock pulse, to allow high speed mode compatible slaves to perform clock stretching. In this mode, the SDA pin is tristated for master-receive data bits and master-transmit ACK/NACK bits, and is configured as push-pull at other times. To avoid the risk of contention when SDA is push-pull, the pin can be configured for open-drain operation, as part of the device-specific configuration.
- **Push-pull 4-wire support:** The push-pull 4-wire configuration separates the SCL input data and output data into separate pins, and separates the SDA input data and output data into separate pins. The SCL/SDA pins are used for input data; the SCLS/SDAS pins are used for output data, with configurable polarity. This simplifies external connections when connecting the LPI2C to the I2C bus through external level shifters or discrete components. When using this 4-wire configuration, the LPI2C master logic and LPI2C slave logic are not able to connect to separate I2C buses.

31.3.3 Slave Mode

To perform all slave mode transfers on the I2C bus, the LPI2C slave logic operates independently from the LPI2C master logic.

31.3.3.1 Address Matching

The LPI2C slave can be configured:

- to match one of two addresses, using either 7-bit or 10-bit addressing modes for each address
- to match a range of addresses in either 7-bit or 10-bit addressing modes
- to match the General Call Address, and generate appropriate flags
- to match the SMBus Alert Address, and generate appropriate flags
- to detect the high speed mode master code, and to disable the digital filters and output valid delay time until the next STOP condition is detected

After a valid address is matched, the LPI2C slave will automatically perform slave-transmit or slave-receive transfers until:

- a NACK is detected (unless IGNACK is set)
- a bit error is detected (the LPI2C slave is driving SDA, but a different value is sampled)
- a (repeated) START or STOP condition is detected

31.3.3.2 Transmit and Receive Data

- The Transmit and Receive Data registers are double-buffered and only update during a slave-transmit and slave-receive transfer, respectively.
- The slave address *that was received* can be configured to be read from either the Receive Data register (for example, when using DMA to transfer data), or from the Address Status register.
- The Transmit Data register can be configured to only request data after a slave-transmit transfer is detected, or to request new data whenever the Transmit Data register is empty.
- The Transmit Data register should only be written when the Transmit Data flag is set.
- The Receive Data register should only be read when the Received Data flag is set (or the Address Valid flag is set and RXCFG=1).
- The Address Status register should only be read when the Address Valid flag is set.

31.3.3.3 Clock Stretching

The LPI2C slave supports many configurable options for when clock stretching is performed. The following conditions can be configured to perform clock stretching:

- During the 9th clock pulse of the address byte and the Address Valid flag is set.
- During the 9th clock pulse of a slave-transmit transfer and the Transmit Data flag is set.
- During the 9th clock pulse of a slave-receive transfer and the Receive Data flag is set.
- During the 8th clock pulse of an address byte or a slave-receive transfer and the Transmit ACK flag is set. In high speed mode, this is disabled.
- Clock stretching can also be extended for CLKHOLD cycles, to allow additional setup time to sample the SDA pin externally. In high speed mode, this is disabled.

Unless extended by the CLKHOLD configuration, clock stretching will extend for one peripheral bus clock cycle after SDA updates when clock stretching is enabled.

31.3.3.4 Timing Parameters

The LPI2C slave can configure the following timing parameters. These parameters are disabled when SCR[FILTEN] is clear, when SCR[FILTDZ] is set in Doze mode, and when LPI2C slave detects high speed mode. When disabled, the LPI2C slave is clocked directly from the I2C bus, and may not satisfy all timing requirements of the I2C specification (such as SDA minimum hold time in Standard/Fast mode).

- SDA data valid time from SCL negation to SDA update
- SCL hold time when clock stretching is enabled to increase setup time when sampling SDA externally
- SCL glitch filter time
- SDA glitch filter time

The LPI2C slave imposes the following restrictions on the timing parameters.

- FILTSDA must be configured to greater than or equal to FILTSCL (unless compensating for board level skew between SDA and SCL).
- DATAVD must be configured less than the minimum SCL low period.

31.3.3.5 Error Conditions

The LPI2C slave can detect the following error conditions:

- Bit error flag will set when the LPI2C slave is driving SDA, but samples a different value than what is expected.
- FIFO error flag will set due to a transmit data underrun or a receive data overrun. To eliminate the possibility of underrun and overrun occurring, enable clock stretching.
- FIFO error flag will also set due to an address overrun when RXCFG is set, otherwise an address overrun is not flagged. To eliminate the possibility of overrun occurring, enable clock stretching.

The LPI2C slave does not implement a timeout due to SCL and/or SDA being stuck low. If this detection is required, then the LPI2C master logic should be used and so software can reset the LPI2C slave when this condition is detected.

31.3.4 Interrupts and DMA Requests

Depending on the specific device, interrupts and DMA requests can be combined in some ways:

- The LPI2C master and slave interrupts may be combined
- The LPI2C master and slave transmit DMA requests may be combined
- The LPI2C master and slave receive DMA requests may be combined

31.3.4.1 Master mode

The next table lists the master mode sources that can generate LPI2C master interrupts and LPI2C master transmit/receive DMA requests.

Table 31-9. Master Interrupts and DMA Requests

Master Status Register (MSR)		Description	Can generate		
Flag	Name		Interrupt?	DMA Request?	Low Power Wakeup?
TDF	Transmit Data Flag	Data can be written to Transmit FIFO, as configured by the Transmit FIFO Watermark MFCR[TXWATER]	Y	TX	Y
RDF	Receive Data Flag	Data can be read from the Receive FIFO, as configured by the Receive FIFO Watermark MFCR[RXWATER]	Y	RX	Y
EPF	End Packet Flag	Master has transmitted a Repeated START or STOP condition	Y	N	Y
SDF	STOP Detect Flag	Master has transmitted a STOP condition	Y	N	Y
NDF	NACK Detect Flag	• During an address byte, the master expected an ACK but detected a NACK • During an address byte, the master expected a NACK but detected an ACK • During a master-transmitter data byte, the master detected a NACK	Y	N	Y
ALF	Arbitration Lost Flag	• The master lost arbitration due to a START/STOP condition detected at the wrong time • Or the master was transmitting data but received different data than the data that was transmitted	Y	N	Y
FEF	FIFO Error Flag	The master is expecting a START condition in the Command FIFO, but the next entry in the Command FIFO is not a START condition	Y	N	Y
PLTF	Pin Low Timeout Flag	Pin low timeout is enabled and SCL (or SDA if configured) is low for longer than the configured timeout	Y	N	Y
DMF	Data Match Flag	The received data matches the configured data match, but the received data is not discarded due to a command FIFO entry	Y	N	Y
MBF	Master Busy Flag	LPI2C master is busy transmitting/receiving data	N	N	N
BBF	Bus Busy Flag	LPI2C master is enabled and activity is detected on I2C bus, but a STOP condition has not been detected and a bus idle timeout (if enabled) has not occurred.	N	N	N

31.3.4.2 Slave mode

The next table lists the slave mode sources that can generate LPI2C slave interrupts and the LPI2C slave transmit/receive DMA requests.

Table 31-10. Slave Interrupts and DMA Requests

Slave Status Register (SSR)		Description	Can generate		
Flag	Name		Interrupt?	DMA Request?	Low Power Wakeup?
TDF	Transmit Data Flag	Data can be written to the Slave Transmit Data Register (STDR)	Y	TX	Y
RDF	Receive Data Flag	Data can be read from the Slave Receive Data Register (SRDR)	Y	RX	Y
AVF	Address Valid Flag	Address can be read from the Slave Address Status Register (SASR)	Y	RX	Y
TAF	Transmit ACK Flag	ACK/NACK can be written to the Slave Transmit ACK Register (STAR)	Y	N	Y
RSF	Repeated Start Flag	Slave has detected an address match followed by a Repeated START condition	Y	RX	Y
SDF	STOP Detect Flag	Slave has detected an address match followed by a STOP condition	Y	RX	Y
BEF	Bit Error Flag	Slave was transmitting data, but received different data than what was transmitted	Y	N	Y
FEF	FIFO Error Flag	• Transmit data underrun • Receive data overrun • Address status overrun (when Receive Data Configuration SCFGR0[RXCFG] = 1,) FEF flag can only set when clock stretching is disabled.	Y	N	Y
AM0F	Address Match 0 Flag	Slave detected an address match with SAMR0[ADDR0] field	Y	N	N
AM1F	Address Match 1 Flag	Slave detected an address match with SAMR1[ADDR1] field or using an address range	Y	N	N
GCF	General Call Flag	Slave detected an address match with the General Call address	Y	N	N
SARF	SMBus Alert Response Flag	Slave detected an address match with the SMBus Alert address	Y	N	N
SBF	Slave Busy Flag	LPI2C slave is busy receiving an address byte or is transmitting/receiving data	N	N	N
BBF	Bus Busy Flag	LPI2C slave is enabled and a START condition is detected on I2C bus, but a STOP condition has not been detected	N	N	N

31.3.4.3 End of Packet DMA Transfer

The end of packet functionality is designed for serial interfaces where the size of the transfer may not be known by software in advance and the data is being pushed by an external device. Examples include UART receive, I2C slave mode and SPI slave mode. The end of packet processing is intended to ensure data does not become stranded in either the receive FIFO or the DMA receive buffer. Support for end of packet processing must be implemented in both the serial interfaces and the DMA controller.

The condition that signals the end of packet is different for each serial interface but is processed by the serial peripheral and DMA in the same way. For example, UART end of packet is signaled by idle line condition, I2C end of packet by STOP and/or Repeated START condition, and SPI end of packet by PCS negation.

When the serial peripheral is configured to signal end of packet condition to the DMA and an end of packet condition is detected by the serial interface, it asserts the DMA request for the receive FIFO irrespective of the watermark configuration. For larger watermark configurations, this ensures the last few words of the transfer are first flushed from the receive FIFO.

The DMA then reads the contents of the receive FIFO, depending on the first word in the FIFO:

- If the receive FIFO is empty, the serial interface signals an end of packet condition to the DMA controller.
- If the receive FIFO is not empty, but the first word in the FIFO is the start of a new packet, then data is not pulled from the receive FIFO and the serial interface signals an end of packet condition to the DMA controller.
- If the receive FIFO is not empty, and the first word in the FIFO is not the start of a new packet, the DMA will transfer the receive data as normal.

Since the DMA may be transferring multiple words on each request, the end of packet condition persists until the DMA minor loop has completed and no additional data is pulled from the receive FIFO. The status flag that triggered the end of packet condition is cleared when the minor loop completes following end of packet being signaled to the DMA controller.

When the DMA detects the end of packet condition, it writes all received words up to the end of packet into system memory and saves the destination address for the word after the last valid data. The DMA then terminates the channel as if the major loop completed, including final offsets and optional interrupts, channel linking and scatter/gather. The final destination address can optionally be saved to system memory or is available in the destination address register.

Since the DMA terminates the major loop, no servicing of the receive FIFO occurs until the DMA is reconfigured through either software or hardware (eg: channel linking or scatter-gather). This delay should be minimized to avoid receiver FIFO overrun. The automatic DMA end of packet processing is not recommended when there are only a few words transferred between end of packet conditions since the DMA will spend more time processing the end of packet than transferring the data. For example, the UART idle line length should be increased as needed to avoid an excessive number of idle conditions.

31.3.5 Peripheral Triggers

The connection of the LPI2C peripheral triggers request is via Inter-Peripheral Crossbar Switch (XBAR).

Table 31-11. LPI2C Triggers

Trigger	Description
Master Output Trigger	The LPI2C master generates an output trigger that can be connected to other peripherals on the device. The master output trigger asserts on both a Repeated START or STOP condition, and the master output trigger remains asserted for one cycle of the LPI2C functional clock divided by the prescaler.
Slave Output Trigger	The LPI2C slave generates an output trigger that can be connected to other peripherals on the device. The slave output trigger asserts on both a Repeated START or STOP condition that occurs after a slave address match, and the slave output trigger remains asserted until the next slave SCL pin negation.
Input Trigger	To control the start of a LPI2C bus transfer, the LPI2C input trigger can be selected instead of the HREQ input. The input trigger is synchronized and to be detected, the input trigger must assert for at least 2 cycles of the LPI2C functional clock divided by the PRESCALE configuration. When the LPI2C is busy, the HREQ input (and therefore the input trigger) is ignored.
Host Request - HREQ input	If host request is asserted and the I2C bus is idle, then it will initiate an LPI2C master transfer.

31.4 Memory Map and Registers

NOTE

Writing a Read-Only (RO) register or reading a Write-Only (WO) register can cause bus errors. This module will not check if programmed values in the registers are correct; the application software must ensure that valid programmed values are being written.

31.4.1 LPI2C register descriptions

31.4.1.1 LPI2C memory map

LPI2C0 base address: E480h

LPI2C1 base address: E4C0h

Offset	Register	Width (In bits)	Access	Reset value
0h	Master Control Register (MCR)	16	RW	0000h
1h	Master Status Register (MSR)	16	W1C	0001h
2h	Master Interrupt Enable Register (MIER)	16	RW	0000h
3h	Master DMA Enable Register (MDER)	16	RW	0000h
4h	Master Configuration Register 0 (MCFGR0)	16	RW	0000h
5h	Master Configuration Register 1 (MCFGR1)	16	RW	0000h
6h	Master Configuration Register 2 (MCFGR2)	16	RW	0000h
7h	Master Configuration Register 3 (MCFGR3)	16	RW	0000h
8h	Master Configuration Register 4 (MCFGR4)	16	RW	0000h
9h	Master Data Match Register (MDMR)	16	RW	0000h
Ah	Master Clock Configuration Register 0 (MCCR0)	16	RW	0000h
Bh	Master Clock Configuration Register 1 (MCCR1)	16	RW	0000h
Ch	Master Clock Configuration Register 2 (MCCR2)	16	RW	0000h
Dh	Master Clock Configuration Register 3 (MCCR3)	16	RW	0000h
Eh	Master FIFO Control Register (MFCR)	16	RW	0000h
Fh	Master FIFO Status Register (MFSR)	16	RO	0000h
10h	Master Transmit Data Register (MTDR)	16	WO	0000h
11h	Master Receive Data Register (MRDR)	16	RO	4000h
12h	Slave Control Register (SCR)	16	RW	0000h
13h	Slave Status Register (SSR)	16	W1C	0000h
14h	Slave Interrupt Enable Register (SIER)	16	RW	0000h
15h	Slave DMA Enable Register (SDER)	16	RW	0000h
16h	Slave Configuration Register 0 (SCFGR0)	16	RW	0000h
17h	Slave Configuration Register 1 (SCFGR1)	16	RW	0000h
18h	Slave Configuration Register 2 (SCFGR2)	16	RW	0000h
19h	Slave Configuration Register 3 (SCFGR3)	16	RW	0000h
1Ah	Slave Address Match Register 0 (SAMR0)	16	RW	0000h
1Bh	Slave Address Match Register 1 (SAMR1)	16	RW	0000h
1Ch	Slave Address Status Register (SASR)	16	RO	4000h
1Dh	Slave Transmit ACK Register (STAR)	16	RW	0000h

Table continues on the next page...

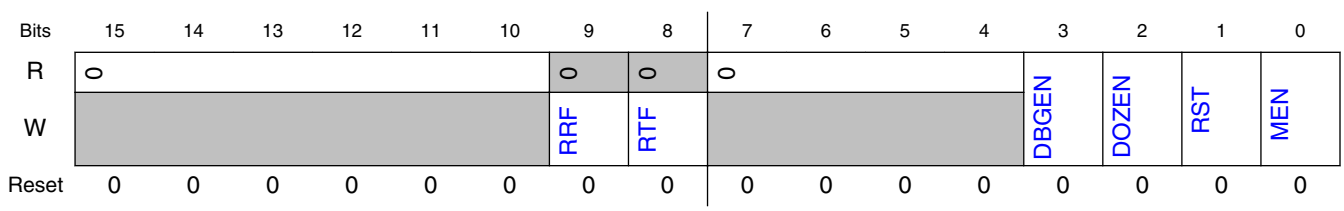
Offset	Register	Width (In bits)	Access	Reset value
1Eh	Slave Transmit Data Register (STDR)	16	WO	0000h
1Fh	Slave Receive Data Register (SRDR)	16	RO	4000h

31.4.1.2 Master Control Register (MCR)

31.4.1.2.1 Offset

Register	Offset
MCR	0h

31.4.1.2.2 Diagram



31.4.1.2.3 Fields

Field	Function
15-10 —	Reserved
9 RRF	Reset Receive FIFO 0b - No effect 1b - Receive FIFO is reset
8 RTF	Reset Transmit FIFO 0b - No effect 1b - Transmit FIFO is reset
7-4 —	Reserved
3 DBGEN	Debug Enable 0b - Master is disabled in debug mode 1b - Master is enabled in debug mode
2 DOZEN	Doze mode enable Enables or disables the master in Doze mode (also called STOP mode in SoC level).

Table continues on the next page...

Field	Function
	0b - Master is enabled in Doze mode 1b - Master is disabled in Doze mode
1 RST	Software Reset Reset all internal master logic and registers, except the Master Control Register. RST remains set until cleared by software. The reset takes effect immediately and remains asserted until negated by software. There is no minimum delay required before clearing the software reset. 0b - Master logic is not reset 1b - Master logic is reset
0 MEN	Master Enable 0b - Master logic is disabled 1b - Master logic is enabled

31.4.1.3 Master Status Register (MSR)

31.4.1.3.1 Offset

Register	Offset
MSR	1h

31.4.1.3.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	DMF	PLTF	FEF	ALF	NDF	SDF	EPF	BBF	MBF	0				RDF	TDF
W		W1C	W1C	W1C	W1C	W1C	W1C	W1C								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

31.4.1.3.3 Fields

Field	Function
15 —	Reserved
14 DMF	Data Match Flag Indicates that the received data has matched the MATCH0 and/or MATCH1 fields (as configured by MCFGR1[MATCFG], Match Configuration). Received data *that is discarded due to CMD field* does not cause Data Match Flag to set. 0b - Have not received matching data

Table continues on the next page...

Field	Function
	1b - Have received matching data
13 PLTF	Pin Low Timeout Flag Will set when the SCL and/or SDA input is low for more than PINLOW cycles (Pin Low Timeout, MCFGR4[PINLOW]), even when the LPI2C master is idle. • Software is responsible for resolving the pin low condition. • Pin Low Timeout Flag cannot be cleared as long as the pin low timeout continues. • Before the LPI2C can initiate a START condition, the Pin Low Timeout Flag must be cleared. 0b - Pin low timeout has not occurred or is disabled 1b - Pin low timeout has occurred
12 FEF	FIFO Error Flag Detects an attempt to send or receive data without first generating a (repeated) START condition. This can occur if the transmit FIFO underflows when the AUTOSTOP bit is set. When FIFO Error Flag is set, the LPI2C master will send a STOP condition (if busy), and will not initiate a new START condition until FIFO Error Flag has been cleared. 0b - No error 1b - Master sending or receiving data without a START condition
11 ALF	Arbitration Lost Flag Set: • if the LPI2C master transmits a logic one and detects a logic zero on the I2C bus • or if the LPI2C master detects a START or STOP condition while the LPI2C master is transmitting data When the Arbitration Lost Flag sets, the LPI2C master will release the I2C bus (go idle), and the LPI2C master will not initiate a new START condition until the Arbitration Lost Flag has been cleared. 0b - Master has not lost arbitration 1b - Master has lost arbitration
10 NDF	NACK Detect Flag Set if the LPI2C master detects a NACK when transmitting an address or data. When set, the master will transmit a STOP condition and will not initiate a new START condition until NACK Detect Flag has been cleared. If a NACK is expected for a given address (as configured by the command word), then the NACK Detect Flag will set if a NACK is not generated. 0b - Unexpected NACK was not detected 1b - Unexpected NACK was detected
9 SDF	STOP Detect Flag Set when the LPI2C master generates a STOP condition. 0b - Master has not generated a STOP condition 1b - Master has generated a STOP condition
8 EPF	End Packet Flag Set when the LPI2C master generates either a repeated START condition or a STOP condition. Does not set when the master first generates a START condition. 0b - Master has not generated a STOP or Repeated START condition 1b - Master has generated a STOP or Repeated START condition
7 BBF	Bus Busy Flag It monitors the SDA and SCL pins. When any of them is low, this flag bit is 1. Otherwise, it is 0. 0b - I2C Bus is idle 1b - I2C Bus is busy
6 MBF	Master Busy Flag It monitors the internal master state machine. When the state machine is in IDLE state, this flag bit is 0. Otherwise, it is 1.

Table continues on the next page...

Field	Function
	0b - I2C Master is idle 1b - I2C Master is busy
5-2 —	Reserved
1 RDF	Receive Data Flag The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than RXWATER. 0b - Receive Data is not ready 1b - Receive data is ready
0 TDF	Transmit Data Flag The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than TXWATER. 0b - Transmit data is not requested 1b - Transmit data is requested

31.4.1.4 Master Interrupt Enable Register (MIER)

31.4.1.4.1 Offset

Register	Offset
MIER	2h

31.4.1.4.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	DMIE	PLTIE	FEIE	ALIE	NDIE	SDIE	EPIE	0						RDIE	TDIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

31.4.1.4.3 Fields

Field	Function
15 —	Reserved
14 DMIE	Data Match Interrupt Enable 0b - Disabled 1b - Enabled

Table continues on the next page...

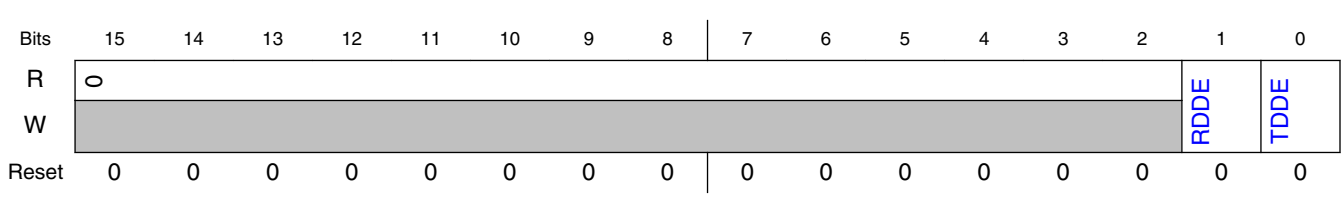
Field	Function
13 PLTIE	Pin Low Timeout Interrupt Enable 0b - Disabled 1b - Enabled
12 FEIE	FIFO Error Interrupt Enable 0b - Enabled 1b - Disabled
11 ALIE	Arbitration Lost Interrupt Enable 0b - Disabled 1b - Enabled
10 NDIE	NACK Detect Interrupt Enable 0b - Disabled 1b - Enabled
9 SDIE	STOP Detect Interrupt Enable 0b - Disabled 1b - Enabled
8 EPIE	End Packet Interrupt Enable 0b - Disabled 1b - Enabled
7-2 —	Reserved
1 RDIE	Receive Data Interrupt Enable 0b - Disabled 1b - Enabled
0 TDIE	Transmit Data Interrupt Enable 0b - Disabled 1b - Enabled

31.4.1.5 Master DMA Enable Register (MDER)

31.4.1.5.1 Offset

Register	Offset
MDER	3h

31.4.1.5.2 Diagram



31.4.1.5.3 Fields

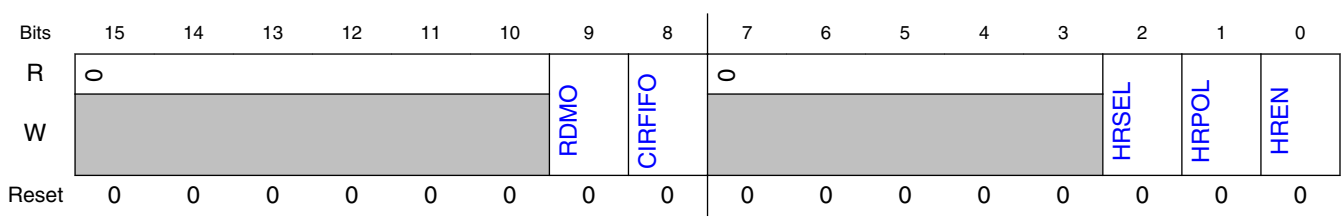
Field	Function
15-2 —	Reserved
1 RDDE	Receive Data DMA Enable 0b - DMA request is disabled 1b - DMA request is enabled
0 TDDE	Transmit Data DMA Enable 0b - DMA request is disabled 1b - DMA request is enabled

31.4.1.6 Master Configuration Register 0 (MCFGR0)

31.4.1.6.1 Offset

Register	Offset
MCFGR0	4h

31.4.1.6.2 Diagram



31.4.1.6.3 Fields

Field	Function
15-10 —	Reserved
9 RDMO	Receive Data Match Only When enabled, all received data that does not cause the Data Match Flag (MSR[DMF]) to set is discarded. After the Data Match Flag is set, the RDMO configuration is ignored. When disabling RDMO, clear RDMO before clearing the Data Match Flag, to ensure that no receive data is lost. 0b - Received data is stored in the receive FIFO

Table continues on the next page...

Field	Function
	1b - Received data is discarded unless the the Data Match Flag (MSR[DMF]) is set
8 CIRFIFO	Circular FIFO Enable When enabled, the transmit FIFO read pointer is saved to a temporary register. The transmit FIFO will be emptied as normal, but after the LPI2C master is idle and the transmit FIFO is empty, then the read pointer value will be restored from the temporary register. This will cause the contents of the transmit FIFO to be cycled through repeatedly. If AUTOSTOP is set, then a STOP condition will be sent whenever the transmit FIFO is empty and the read pointer is restored. 0b - Circular FIFO is disabled 1b - Circular FIFO is enabled
7-3 —	Reserved
2 HRSEL	Host Request Select Selects the source of the host request input. When host request input is enabled, the Host Request Select field should remain static (the Host Request Select should not change). 0b - Host request input is HREQ 1b - Host request input is input trigger
1 HRPOL	Host Request Polarity Configures the polarity of the host request input pin. When host request input is enabled, the Host Request Polarity field should remain static (Host Request Polarity should not change). 0b - Active low 1b - Active high
0 HREN	Host Request Enable When enabled, the LPI2C master will only initiate a START condition if the host request input is asserted and the bus is idle. A repeated START is not affected by the host request. 0b - Host request input is disabled 1b - Host request input is enabled

31.4.1.7 Master Configuration Register 1 (MCFGR1)

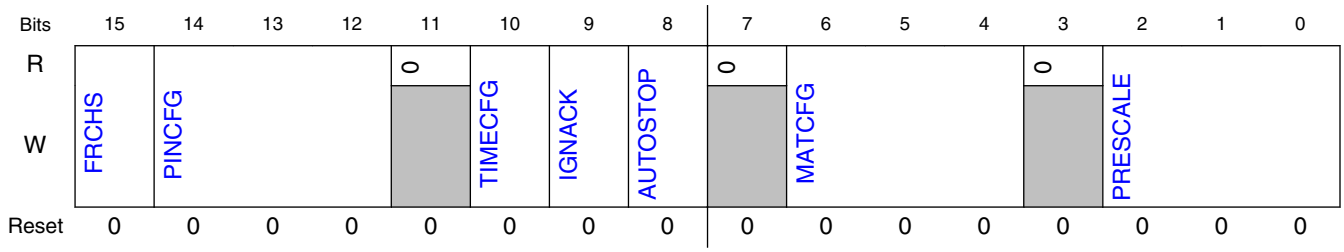
31.4.1.7.1 Offset

Register	Offset
MCFGR1	5h

31.4.1.7.2 Function

The MCFGR1 should only be written when the I2C Master is disabled.

31.4.1.7.3 Diagram



31.4.1.7.4 Fields

Field	Function
15 FRCHS	Force HS-mode When set, forces the LPI2C pin state into HS-mode. Setting this does not impact the digital filter configuration or digital timing parameters. 0b - No effect 1b - LPI2C pin state forced into HS-mode.
14-12 PINCFG	Pin Configuration Configures the pin mode for LPI2C. **Table 31-12. 2-pin / 4-pin pin configurations for masters and slaves** 000b - 2-pin open drain mode 001b - 2-pin output only mode (ultra-fast mode) 010b - 2-pin push-pull mode 011b - 4-pin push-pull mode 100b - 2-pin open drain mode with separate LPI2C slave 101b - 2-pin output only mode (ultra-fast mode) with separate LPI2C slave 110b - 2-pin push-pull mode with separate LPI2C slave 111b - 4-pin push-pull mode (inverted outputs)
11 —	Reserved

PINCFG	SCL / SDA pins	SCLS / SDAS pins
000	Bi-directional open drain for master and slave	Not used
001	Output-only (ultra-fast mode) open drain for master and slave	Not used
010	Bi-directional push-pull for master and slave	Not used
011	Input only for master and slave	Output-only push-pull for master and slave
100	Bi-directional open drain for master	Bi-directional open drain for slave
101	Output-only (ultra-fast mode) open drain for master	Output-only open drain for slave
110	Bi-directional push-pull for master	Bi-directional push-pull for slave
111	Input only for master and slave	Inverted output-only push-pull for master and slave

Table continues on the next page...

Field	Function
10 TIMECFG	Timeout Configuration 0b - Pin Low Timeout Flag will set if SCL is low for longer than the configured timeout 1b - Pin Low Timeout Flag will set if either SCL or SDA is low for longer than the configured timeout
9 IGNACK	IGNACK When set, the received NACK field is ignored and assumed to be ACK. IGNACK bit is required to be set in Ultra-Fast Mode. 0b - LPI2C Master will receive ACK and NACK normally 1b - LPI2C Master will treat a received NACK as if it (NACK) was an ACK
8 AUTOSTOP	Automatic STOP Generation When enabled, a STOP condition is generated whenever the LPI2C master is busy and the transmit FIFO is empty. The STOP condition can also be generated using a transmit FIFO command. 0b - No effect 1b - STOP condition is automatically generated whenever the transmit FIFO is empty and the LPI2C master is busy
7 —	Reserved
6-4 MATCFG	Match Configuration Configures the condition that will cause the DMF to set. 000b - Match is disabled 001b - Reserved 010b - Match is enabled (1st data word equals MATCH0 OR MATCH1) 011b - Match is enabled (any data word equals MATCH0 OR MATCH1) 100b - Match is enabled (1st data word equals MATCH0 AND 2nd data word equals MATCH1) 101b - Match is enabled (any data word equals MATCH0 AND next data word equals MATCH1) 110b - Match is enabled (1st data word AND MATCH1 equals MATCH0 AND MATCH1) 111b - Match is enabled (any data word AND MATCH1 equals MATCH0 AND MATCH1)
3 —	Reserved
2-0 PRESCALE	Prescaler Configures the clock prescaler used for all LPI2C master logic, except for the digital glitch filters. 000b - Divide by 1 001b - Divide by 2 010b - Divide by 4 011b - Divide by 8 100b - Divide by 16 101b - Divide by 32 110b - Divide by 64 111b - Divide by 128

31.4.1.8 Master Configuration Register 2 (MCFGR2)

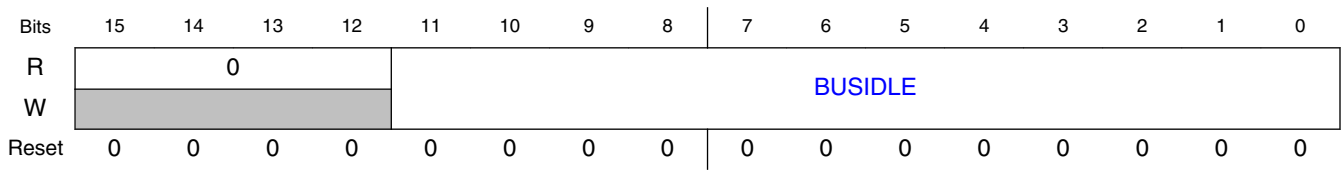
31.4.1.8.1 Offset

Register	Offset
MCFGR2	6h

31.4.1.8.2 Function

The Master Configuration Register 2 should only be written when the I2C Master is disabled.

31.4.1.8.3 Diagram



31.4.1.8.4 Fields

Field	Function
15-12 —	Reserved
11-0 BUSIDLE	Bus Idle Timeout Configures the bus idle timeout period in clock cycles. • If both SCL and SDA are high for longer than BUSIDLE cycles, then the I2C bus is assumed to be idle and the master can generate a START condition • When Bus Idle Timeout is set to zero, the Bus Idle Timeout is disabled

31.4.1.9 Master Configuration Register 3 (MCFGR3)

31.4.1.9.1 Offset

Register	Offset
MCFGR3	7h

31.4.1.9.2 Function

The Master Configuration Register 3 should only be written when the I2C Master is disabled.

31.4.1.9.3 Diagram

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

31.4.1.9.4 Fields

Field	Function
15-12 —	Reserved
11-8 FILTSDA	Glitch Filter SDA Configures the I2C master digital glitch filters for SDA input. • A configuration of 0 will disable the glitch filter • Glitches equal to or less than FILTSDA cycles long will be filtered out and ignored • The latency through the glitch filter is equal to FILTSDA cycles, and must be configured to be less than the minimum SCL low or high period • The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is automatically bypassed in High Speed mode
7-4 —	Reserved
3-0 FILTSCL	Glitch Filter SCL Configures the I2C master digital glitch filters for SCL input. • A configuration of 0 will disable the glitch filter • Glitches equal to or less than FILTSCL cycles long will be filtered out and ignored • The latency through the glitch filter is equal to FILTSCL cycles, and must be configured to be less than the minimum SCL low or high period • The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is automatically bypassed in High Speed mode

31.4.1.10 Master Configuration Register 4 (MCFGR4)

31.4.1.10.1 Offset

Register	Offset
MCFGR4	8h

31.4.1.10.2 Function

The MCFGR4 should only be written when the I2C Master is disabled.

31.4.1.10.3 Diagram

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

31.4.1.10.4 Fields

Field	Function
15-12 —	Reserved
11-0 PINLOW	Pin Low Timeout Configures the pin low timeout flag in clock cycles. • If SCL or, either SCL or SDA, is low for longer than (PINLOW * 256) cycles, then PLTF is set • When Pin Low Timeout is set to zero, the Pin Low Timeout feature is disabled

31.4.1.11 Master Data Match Register (MDMR)

31.4.1.11.1 Offset

Register	Offset
MDMR	9h

31.4.1.11.2 Function

The MDMR should only be written when the I2C Master is disabled or idle.

31.4.1.11.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MATCH1								MATCH0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

31.4.1.11.4 Fields

Field	Function
15-8 MATCH1	Match 1 Value Compared against the received data when receive data match is enabled.
7-0 MATCH0	Match 0 Value Compared against the received data when receive data match is enabled.

31.4.1.12 Master Clock Configuration Register 0 (MCCR0)

31.4.1.12.1 Offset

Register	Offset
MCCR0	Ah

31.4.1.12.2 Function

The MCCR0 cannot be changed when the I2C master is enabled and is used for standard, fast, fast-mode plus and ultra-fast transfers.

31.4.1.12.3 Diagram

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

31.4.1.12.4 Fields

Field	Function
15-14 —	Reserved
13-8 CLKHI	Clock High Period Minimum number of cycles (minus one) that the SCL clock is driven high by the master. The SCL high time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles.
7-6	Reserved

Table continues on the next page...

Field	Function
—	
5-0 CLKLO	Clock Low Period Minimum number of cycles (minus one) that the SCL clock is driven low by the master. The Clock Low Period value is also used for the minimum bus free time between a STOP and a START condition; this is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles.

31.4.1.13 Master Clock Configuration Register 1 (MCCR1)

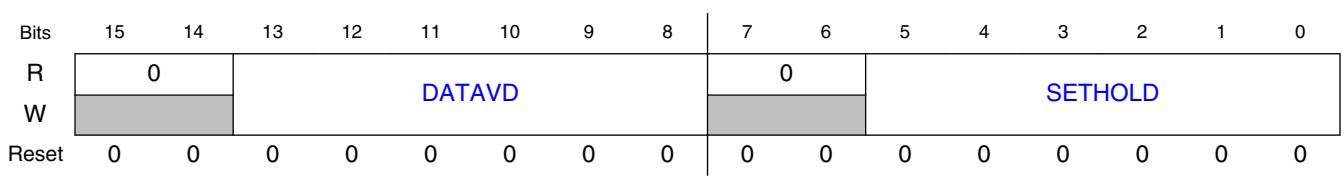
31.4.1.13.1 Offset

Register	Offset
MCCR1	Bh

31.4.1.13.2 Function

The MCCR1 cannot be changed when the I2C master is enabled and is used for standard, fast, fast-mode plus and ultra-fast transfers.

31.4.1.13.3 Diagram



31.4.1.13.4 Fields

Field	Function
15-14 —	Reserved
13-8 DATAVD	Data Valid Delay Minimum number of cycles (minus one) that is used as the data hold time for SDA. Must be configured less than the minimum SCL low period.
7-6 —	Reserved

Table continues on the next page...

Field	Function
5-0 SETHOLD	Setup Hold Delay Minimum number of cycles (minus one) that is used by the master • as the hold time for a START condition • as the setup and hold time for a repeated START condition • as the setup time for a STOP condition The setup time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles.

31.4.1.14 Master Clock Configuration Register 2 (MCCR2)

31.4.1.14.1 Offset

Register	Offset
MCCR2	Ch

31.4.1.14.2 Function

The MCCR2 cannot be changed when the I2C master is enabled and is used for high speed mode transfers. The separate clock configuration for high speed mode allows arbitration to take place in Fast mode (with timing configured by MCCR0 and MCCR1), before switching to high speed mode (with timing configured by MCCR2 and MCCR3).

31.4.1.14.3 Diagram

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

31.4.1.14.4 Fields

Field	Function
15-14 —	Reserved
13-8	Clock High Period

Table continues on the next page...

Field	Function
CLKHI	Minimum number of cycles (minus one) that the SCL clock is driven high by the master. The SCL high time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles.
7-6 —	Reserved
5-0 CLKLO	Clock Low Period Minimum number of cycles (minus one) that the SCL clock is driven low by the master. The Clock Low Period value is also used for the minimum bus free time between a STOP and a START condition; this is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles.

31.4.1.15 Master Clock Configuration Register 3 (MCCR3)

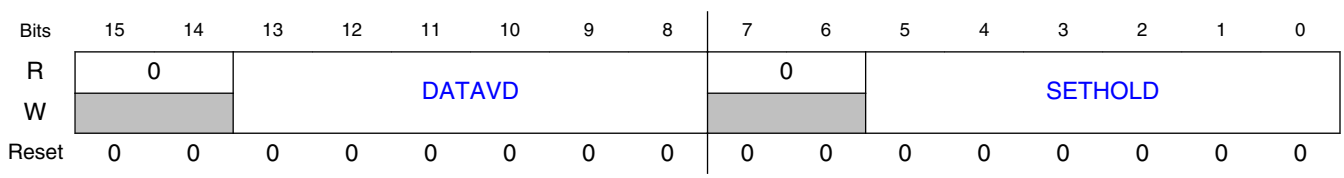
31.4.1.15.1 Offset

Register	Offset
MCCR3	Dh

31.4.1.15.2 Function

The MCCR3 cannot be changed when the I2C master is enabled and is used for high speed mode transfers. The separate clock configuration for high speed mode allows arbitration to take place in Fast mode (with timing configured by MCCR0 and MCCR1), before switching to high speed mode (with timing configured by MCCR2 and MCCR3).

31.4.1.15.3 Diagram



31.4.1.15.4 Fields

Field	Function
15-14 —	Reserved

Table continues on the next page...

Field	Function
13-8 DATAVD	Data Valid Delay Minimum number of cycles (minus one) that is used as the data hold time for SDA. The Data Valid Delay must be configured to be less than the minimum SCL low period.
7-6 —	Reserved
5-0 SETHOLD	Setup Hold Delay Minimum number of cycles (minus one) that is used by the master • as the hold time for a START condition • as the setup and hold time for a repeated START condition • as the setup time for a STOP condition The setup time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles.

31.4.1.16 Master FIFO Control Register (MFCR)

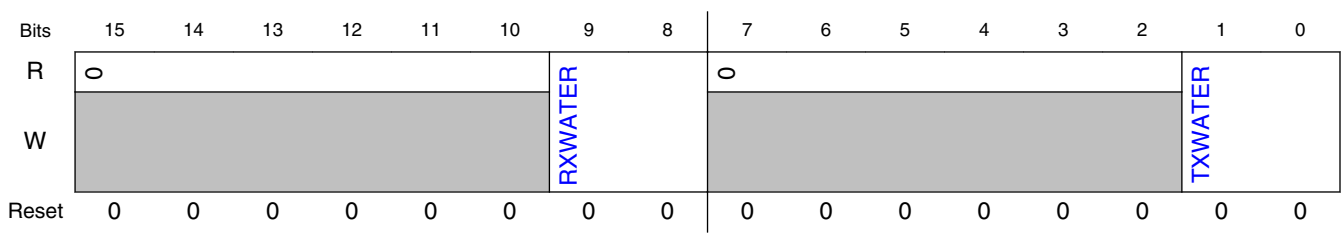
31.4.1.16.1 Offset

Register	Offset
MFCR	Eh

31.4.1.16.2 Function

The Master FIFO control register is only used in Stop mode when the MFCR register is static (i.e., the MFCR register is not changing).

31.4.1.16.3 Diagram



31.4.1.16.4 Fields

Field	Function
15-10 —	Reserved
9-8 RXWATER	Receive FIFO Watermark The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than RXWATER. Writing a value equal to or greater than the FIFO size will be truncated.
7-2 —	Reserved
1-0 TXWATER	Transmit FIFO Watermark The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than TXWATER. Writing a value equal to or greater than the FIFO size will be truncated.

31.4.1.17 Master FIFO Status Register (MFSR)

31.4.1.17.1 Offset

Register	Offset
MFSR	Fh

31.4.1.17.2 Diagram

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

31.4.1.17.3 Fields

Field	Function
15-11 —	Reserved
10-8 RXCOUNT	Receive FIFO Count Returns the number of words in the receive FIFO.
7-3	Reserved

Table continues on the next page...

Field	Function
—	
2-0 TXCOUNT	Transmit FIFO Count Returns the number of words in the transmit FIFO.

31.4.1.18 Master Transmit Data Register (MTDR)

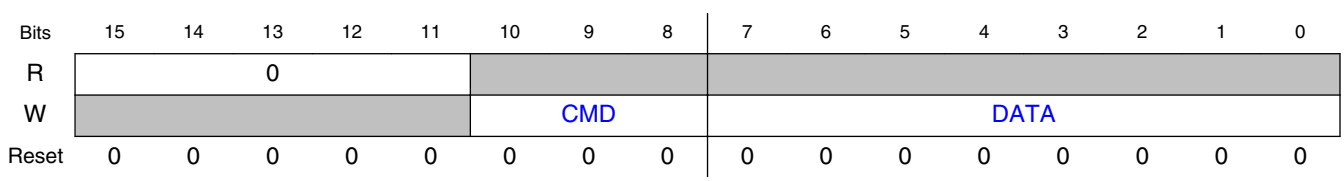
31.4.1.18.1 Offset

Register	Offset
MTDR	10h

31.4.1.18.2 Function

- An 8-bit write to the CMD field will store the data in the Command FIFO, but does not increment the FIFO write pointer.
- An 8-bit write to the DATA field will zero extend the CMD field, unless the CMD field has been written separately since the last FIFO write; it (the 8-bit write) also increments the FIFO write pointer.
- A 16-bit write to both the CMD and DATA fields will increment the FIFO.

31.4.1.18.3 Diagram



31.4.1.18.4 Fields

Field	Function
15-11 —	Reserved
10-8 CMD	Command Data 000b - Transmit DATA[7:0] 001b - Receive (DATA[7:0] + 1) bytes 010b - Generate STOP condition

Table continues on the next page...

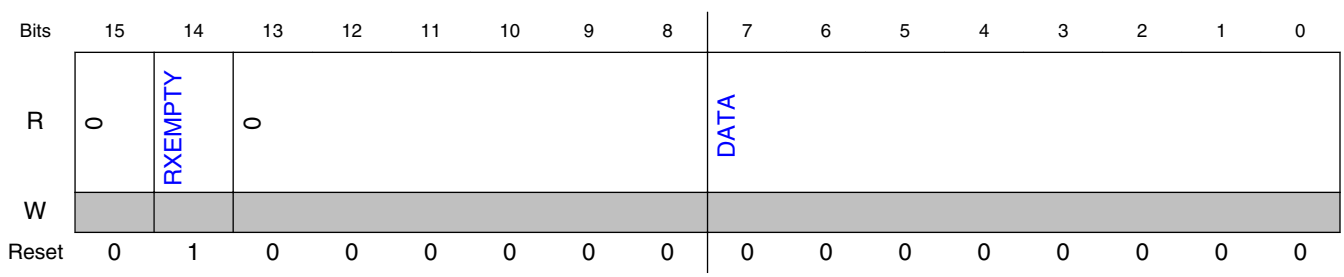
Field	Function
	011b - Receive and discard (DATA[7:0] + 1) bytes 100b - Generate (repeated) START and transmit address in DATA[7:0] 101b - Generate (repeated) START and transmit address in DATA[7:0]. This transfer expects a NACK to be returned. 110b - Generate (repeated) START and transmit address in DATA[7:0] using high speed mode 111b - Generate (repeated) START and transmit address in DATA[7:0] using high speed mode. This transfer expects a NACK to be returned.
7-0 DATA	Transmit Data Performing an 8-bit write to DATA will zero extend the CMD field.

31.4.1.19 Master Receive Data Register (MRDR)

31.4.1.19.1 Offset

Register	Offset
MRDR	11h

31.4.1.19.2 Diagram



31.4.1.19.3 Fields

Field	Function
15 —	Reserved
14 RXEMPTY	RX Empty 0b - Receive FIFO is not empty 1b - Receive FIFO is empty
13-8 —	Reserved
7-0	Receive Data

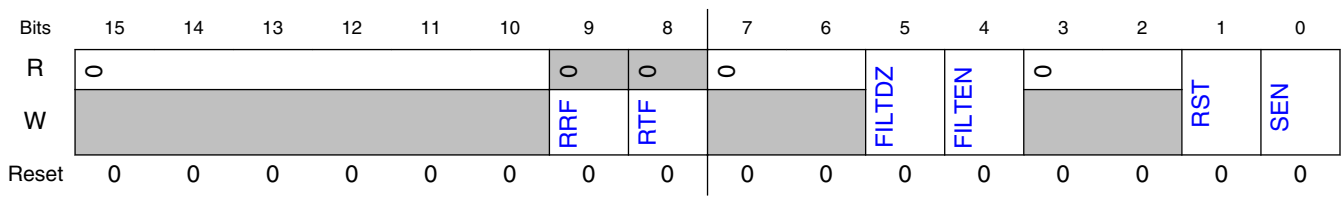
Field	Function
DATA	Reading the Receive Data register returns the data received by the I2C master that has not been discarded. Receive data can be discarded due to the CMD field, or the master can be configured to discard non-matching data.

31.4.1.20 Slave Control Register (SCR)

31.4.1.20.1 Offset

Register	Offset
SCR	12h

31.4.1.20.2 Diagram



31.4.1.20.3 Fields

Field	Function
15-10 —	Reserved
9 RRF	Reset Receive FIFO 0b - No effect 1b - Receive Data Register is now empty
8 RTF	Reset Transmit FIFO 0b - No effect 1b - Transmit Data Register is now empty
7-6 —	Reserved
5 FILTDZ	Filter Doze Enable Filter Doze Enable bit should only be updated when the I2C Slave is disabled. 0b - Filter remains enabled in Doze mode 1b - Filter is disabled in Doze mode
4 FILTEN	Filter Enable Filter Enable bit should only be updated when the I2C Slave is disabled.

Table continues on the next page...

Field	Function
	0b - Disable digital filter and output delay counter for slave mode 1b - Enable digital filter and output delay counter for slave mode
3-2 —	Reserved
1 RST	Software Reset The reset takes effect immediately and remains asserted until negated by software. There is no minimum delay required before clearing the software reset. 0b - Slave mode logic is not reset 1b - Slave mode logic is reset
0 SEN	Slave Enable 0b - I2C Slave mode is disabled 1b - I2C Slave mode is enabled

31.4.1.21 Slave Status Register (SSR)

31.4.1.21.1 Offset

Register	Offset
SSR	13h

31.4.1.21.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SARF	GCF	AM1F	AM0F	FEF	BEF	SDF	RSF	BBF	SBF	0		TAF	AVF	RDF	TDF
W					W1C	W1C	W1C	W1C								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

31.4.1.21.3 Fields

Field	Function
15 SARF	SMBus Alert Response Flag • SMBus Alert Response Flag is cleared by reading the Address Status Register • SMBus Alert Response Flag cannot generate an asynchronous wakeup 0b - SMBus Alert Response is disabled or not detected 1b - SMBus Alert Response is enabled and detected

Table continues on the next page...

Field	Function
14 GCF	General Call Flag Indicates whether a slave has detected the General Call Address. • General Call Flag is cleared by reading the Address Status Register • General Call Flag cannot generate an asynchronous wakeup 0b - Slave has not detected the General Call Address or the General Call Address is disabled 1b - Slave has detected the General Call Address
13 AM1F	Address Match 1 Flag Indicates that the received address has matched the ADDR1 field or ADDR0 to ADDR1 range as configured by ADDRCFG. • Address Match 1 Flag is cleared by reading the Address Status Register • Address Match 1 Flag cannot generate an asynchronous wakeup 0b - Have not received an ADDR1 or ADDR0/ADDR1 range matching address 1b - Have received an ADDR1 or ADDR0/ADDR1 range matching address
12 AM0F	Address Match 0 Flag Indicates that the received address has matched the ADDR0 field as configured by ADDRCFG. • Address Match 0 Flag is cleared by reading the Address Status Register • Address Match 0 Flag cannot generate an asynchronous wakeup 0b - Have not received an ADDR0 matching address 1b - Have received an ADDR0 matching address
11 FEF	FIFO Error Flag FIFO error flag can only be set when clock stretching is disabled. 0b - FIFO underflow or overflow was not detected 1b - FIFO underflow or overflow was detected
10 BEF	Bit Error Flag Bit Error Flag will set if the LPI2C slave transmits a logic one and detects a logic zero on the I2C bus. The slave will ignore the rest of the transfer until the next (repeated) START condition. 0b - Slave has not detected a bit error 1b - Slave has detected a bit error
9 SDF	STOP Detect Flag STOP Detect Flag will set when the LPI2C slave detects a STOP condition and if the LPI2C slave matched the last address byte. 0b - Slave has not detected a STOP condition 1b - Slave has detected a STOP condition
8 RSF	Repeated Start Flag Repeated Start Flag will set when the LPI2C slave detects a repeated START condition and if the LPI2C slave matched the last address byte. The Repeated Start Flag does not set when the slave first detects a START condition. 0b - Slave has not detected a Repeated START condition 1b - Slave has detected a Repeated START condition
7 BBF	Bus Busy Flag Indicates if an I2C bus is idle or busy. 0b - I2C Bus is idle 1b - I2C Bus is busy
6 SBF	Slave Busy Flag Indicates if an I2C slave is idle or busy. 0b - I2C Slave is idle 1b - I2C Slave is busy

Table continues on the next page...

Field	Function
5-4 —	Reserved
3 TAF	Transmit ACK Flag Transmit ACK Flag is cleared by writing the Transmit ACK register. 0b - Transmit ACK/NACK is not required 1b - Transmit ACK/NACK is required
2 AVF	Address Valid Flag Address Valid Flag is cleared by reading the address status register. When Receive Data Configuration (SCFGR0[RXCFG]) is set, the Address Valid Flag is also cleared by reading the Receive Data register. 0b - Address Status Register is not valid 1b - Address Status Register is valid
1 RDF	Receive Data Flag Receive Data Flag is cleared by reading the receive data register. When Receive Data Configuration (SCFGR0[RXCFG]) is set, the Receive Data Flag is not cleared when reading the Receive Data register and if AVF is set. 0b - Receive data is not ready 1b - Receive data is ready
0 TDF	Transmit Data Flag Transmit Data Flag is cleared by writing the Transmit Data register. When Transmit Flag Configuration (TXCFG) is clear, and if a NACK or Repeated START or STOP condition is detected, then Transmit Data Flag is also cleared. 0b - Transmit data not requested 1b - Transmit data is requested

31.4.1.22 Slave Interrupt Enable Register (SIER)

31.4.1.22.1 Offset

Register	Offset
SIER	14h

31.4.1.22.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SARIE	GCIE	AM1IE	AM0IE	FEIE	BEIE	SDIE	RSIE	0				TAIE	AVIE	RDIE	TDIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

31.4.1.22.3 Fields

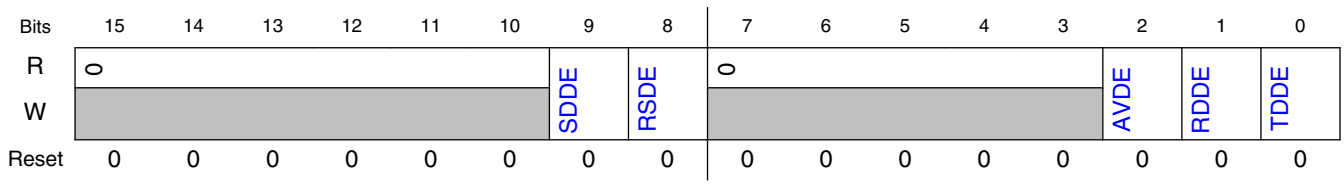
Field	Function
15 SARIE	SMBus Alert Response Interrupt Enable 0b - Disabled 1b - Enabled
14 GCIE	General Call Interrupt Enable 0b - Disabled 1b - Enabled
13 AM1IE	Address Match 1 Interrupt Enable 0b - Disabled 1b - Enabled
12 AM0IE	Address Match 0 Interrupt Enable 0b - Disabled 1b - Enabled
11 FEIE	FIFO Error Interrupt Enable 0b - Disabled 1b - Enabled
10 BEIE	Bit Error Interrupt Enable 0b - Disabled 1b - Enabled
9 SDIE	STOP Detect Interrupt Enable 0b - Disabled 1b - Enabled
8 RSIE	Repeated Start Interrupt Enable 0b - Disabled 1b - Enabled
7-4 —	Reserved
3 TAIE	Transmit ACK Interrupt Enable 0b - Disabled 1b - Enabled
2 AVIE	Address Valid Interrupt Enable 0b - Disabled 1b - Enabled
1 RDIE	Receive Data Interrupt Enable 0b - Disabled 1b - Enabled
0 TDIE	Transmit Data Interrupt Enable 0b - Disabled 1b - Enabled

31.4.1.23 Slave DMA Enable Register (SDER)

31.4.1.23.1 Offset

Register	Offset
SDER	15h

31.4.1.23.2 Diagram



31.4.1.23.3 Fields

Field	Function
15-10 —	Reserved
9 SDDE	Stop Detect DMA Enable Enables DMA end-of-packet processing on stop detect when set. Reading the Receive Data register when the Receive Data register is empty, will: • generate a DMA end-of-packet response • return 0x40FF • clear the STOP Detect Flag (MSR[SDF]) 0b - DMA request is disabled 1b - DMA request is enabled
8 RSDE	Repeated Start DMA Enable When Repeated Start DMA Enable is set, it enables DMA end-of-packet processing on repeated start. Reading the Receive Data register when the Receive Data register is empty, will: • generate a DMA end of packet response • return 0x40FF • clear the Receive Data Flag (MSR[RDF]) 0b - DMA request is disabled 1b - DMA request is enabled
7-3 —	Reserved
2 AVDE	Address Valid DMA Enable The Address Valid DMA request is shared with the Receive Data DMA request. If both Address Valid DMA request and Receive Data DMA request are enabled, then set Receive Data Configuration (SCFGR0[RXCFG]), to allow the DMA to read the address from the Receive Data Register. 0b - DMA request is disabled 1b - DMA request is enabled
1	Receive Data DMA Enable

Table continues on the next page...

Field	Function
RDDE	0b - DMA request is disabled 1b - DMA request is enabled
0 TDDE	Transmit Data DMA Enable 0b - DMA request is disabled 1b - DMA request is enabled

31.4.1.24 Slave Configuration Register 0 (SCFGR0)

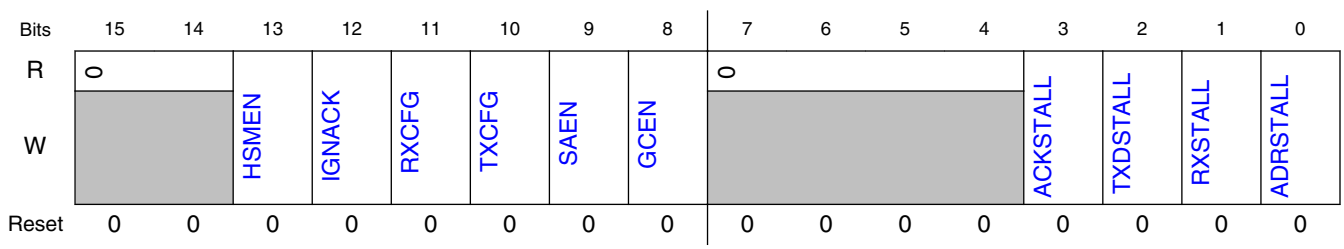
31.4.1.24.1 Offset

Register	Offset
SCFGR0	16h

31.4.1.24.2 Function

The Slave Configuration Register 0 should only be written when the I2C Slave is disabled.

31.4.1.24.3 Diagram



31.4.1.24.4 Fields

Field	Function
15-14 —	Reserved
13 HSMEN	High Speed Mode Enable Enables detection of the High-Speed Mode master code of slave address 0000_1XX, but does not cause an address match on this code. When set and any HS-mode master code is detected, the FILTEN and ACKSTALL bits are ignored until the next STOP condition is detected. 0b - Disables detection of HS-mode master code 1b - Enables detection of HS-mode master code

Table continues on the next page...

Field	Function
12 IGNACK	Ignore NACK When Ignore NACK is set, the LPI2C slave will continue transfers after a NACK is detected. Ignore NACK bit is required to be set in Ultra-Fast Mode. 0b - Slave will end transfer when NACK is detected 1b - Slave will not end transfer when NACK detected
11 RXCFG	Receive Data Configuration 0b - Reading the Receive Data register will return received data and clear the Receive Data flag (MSR[RDF]). 1b - Reading the Receive Data register when the Address Valid flag (SSR[AVF])is set, will return the Address Status register and clear the Address Valid flag. Reading the Receive Data register when the Address Valid flag is clear, will return received data and clear the Receive Data flag (MSR[RDF]).
10 TXCFG	Transmit Flag Configuration The transmit data flag will always assert before a NACK is detected at the end of a slave-transmit transfer. This can cause an extra word to be written to the transmit data FIFO. • When TXCFG=0, the Transmit Data register is automatically emptied when a slave-transmit transfer is detected. This causes the transmit data flag to assert whenever a slave-transmit transfer is detected, and causes the transmit data flag to negate at the end of the slave-transmit transfer. • When TXCFG=1, the Transmit Data flag will assert whenver the Transmit Data register is empty, and the Transmit Data flag will negate when the Transmit Data register is full. This allows the Transmit Data register to be filled before a slave-transmit transfer is detected, but can cause the Transmit Data register to be written before a NACK is detected on the last byte of a slave transmit transfer. 0b - Transmit Data Flag will only assert during a slave-transmit transfer when the Transmit Data register is empty 1b - Transmit Data Flag will assert whenever the Transmit Data register is empty
9 SAEN	SMBus Alert Enable 0b - Disables match on SMBus Alert 1b - Enables match on SMBus Alert
8 GCEN	General Call Enable 0b - General Call address is disabled 1b - General Call address is enabled
7-4 —	Reserved
3 ACKSTALL	ACK SCL Stall Enables SCL clock stretching during slave-transmit address byte(s) and slave-receiver address and data byte(s), to allow software to write the Transmit ACK Register before the ACK or NACK is transmitted. Clock stretching occurs when transmitting the 9th bit, and is therefore not compatible with high speed mode. When ACKSTALL is enabled, there is no need to set either RX SCL Stall (SCFGR0[RXSTALL]) or Address SCL Stall (SCFGR0[ADRSTALL]). When ACKSTALL is enabled and there is an address match on the first byte of a 10-bit address, the Address Valid Flag will be set allowing software to read the Received Address before writing the Transmit ACK Register. 0b - Clock stretching is disabled 1b - Clock stretching is enabled
2 TXDSTALL	TX Data SCL Stall Enables SCL clock stretching when the Transmit Data flag (SSR[TDF]) is set during a slave-transmit transfer. Clock stretching occurs following the 9th bit, and is therefore compatible with high speed mode.

Table continues on the next page...

Field	Function
	0b - Clock stretching is disabled 1b - Clock stretching is enabled
1 RXSTALL	RX SCL Stall Enables SCL clock stretching when the Receive Data flag (SSR[RDF]) is set during a slave-receive transfer. Clock stretching occurs following the 9th bit, and is therefore compatible with high speed mode. 0b - Clock stretching is disabled 1b - Clock stretching is enabled
0 ADRSTALL	Address SCL Stall Enables SCL clock stretching when the Address Valid Flag (SSR[AVF]) is asserted. Clock stretching only occurs following the 9th bit, and is therefore compatible with high speed mode. 0b - Clock stretching is disabled 1b - Clock stretching is enabled

31.4.1.25 Slave Configuration Register 1 (SCFGR1)

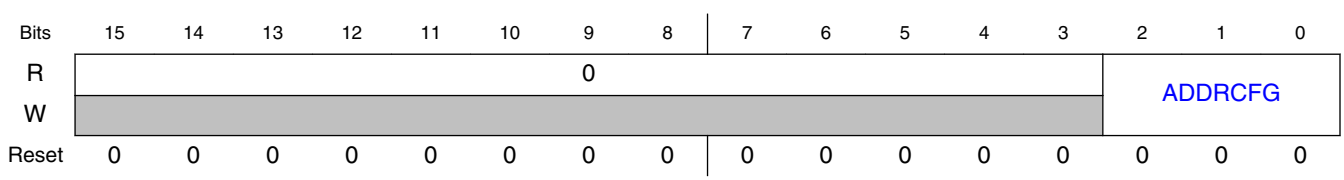
31.4.1.25.1 Offset

Register	Offset
SCFGR1	17h

31.4.1.25.2 Function

The Slave Configuration Register 1 should only be written when the I2C Slave is disabled.

31.4.1.25.3 Diagram



31.4.1.25.4 Fields

Field	Function
15-3 —	Reserved

Table continues on the next page...

Field	Function
2-0 ADDRCFG	Address Configuration Configures the condition that will cause an address to match. 000b - Address match 0 (7-bit) 001b - Address match 0 (10-bit) 010b - Address match 0 (7-bit) or Address match 1 (7-bit) 011b - Address match 0 (10-bit) or Address match 1 (10-bit) 100b - Address match 0 (7-bit) or Address match 1 (10-bit) 101b - Address match 0 (10-bit) or Address match 1 (7-bit) 110b - From Address match 0 (7-bit) to Address match 1 (7-bit) 111b - From Address match 0 (10-bit) to Address match 1 (10-bit)

31.4.1.26 Slave Configuration Register 2 (SCFGR2)

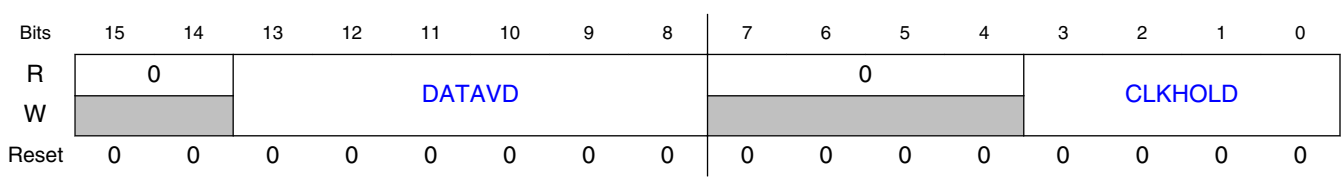
31.4.1.26.1 Offset

Register	Offset
SCFGR2	18h

31.4.1.26.2 Function

The Slave Configuration Register 2 should only be written when the I2C Slave is disabled.

31.4.1.26.3 Diagram



31.4.1.26.4 Fields

Field	Function
15-14 —	Reserved
13-8 DATAVD	Data Valid Delay Configures the SDA data valid delay time for the I2C slave, and is equal to FILTSCL+DATAVD+3 cycles.

Table continues on the next page...

Field	Function
	• The data valid delay must be configured to be less than the minimum SCL low period • The I2C slave data valid delay time is not affected by the PRESCALE configuration, and the I2C slave data valid delay time is disabled in high speed mode
7-4 —	Reserved
3-0 CLKHOLD	Clock Hold Time Configures the minimum clock hold time for the I2C slave, when clock stretching is enabled. • The minimum hold time is equal to CLKHOLD+3 cycles • The I2C slave clock hold time is not affected by the PRESCALE configuration, and the I2C slave clock hold time is disabled in high speed mode

31.4.1.27 Slave Configuration Register 3 (SCFGR3)

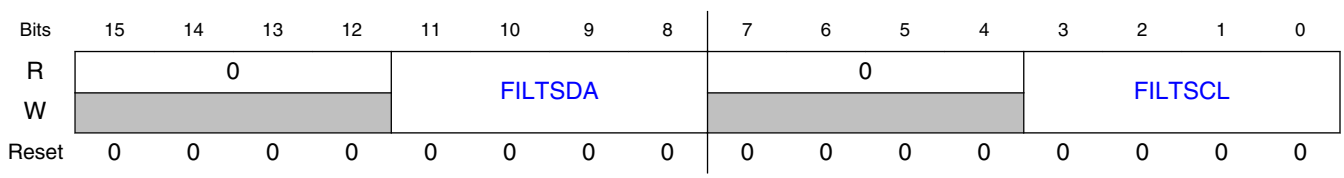
31.4.1.27.1 Offset

Register	Offset
SCFGR3	19h

31.4.1.27.2 Function

The Slave Configuration Register 3 should only be written when the I2C Slave is disabled.

31.4.1.27.3 Diagram



31.4.1.27.4 Fields

Field	Function
15-12 —	Reserved
11-8 FILTSDA	Glitch Filter SDA Configures the I2C slave digital glitch filters for SDA input.

Table continues on the next page...

Field	Function
	• A configuration of 0 will disable the glitch filter • Glitches equal to or less than FILTSDA cycles long will be filtered out and ignored • The latency through the glitch filter is equal to FILTSDA+3 cycles, and must be configured to be less than the minimum SCL low or high period • The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is disabled in high speed mode
7-4 —	Reserved
3-0 FILTSCL	Glitch Filter SCL Configures the I2C slave digital glitch filters for SCL input. • A configuration of 0 will disable the glitch filter • Glitches equal to or less than FILTSCL cycles long will be filtered out and ignored • The latency through the glitch filter is equal to FILTSCL+3 cycles, and must be configured to be less than the minimum SCL low or high period • The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is disabled in high speed mode

31.4.1.28 Slave Address Match Register 0 (SAMR0)

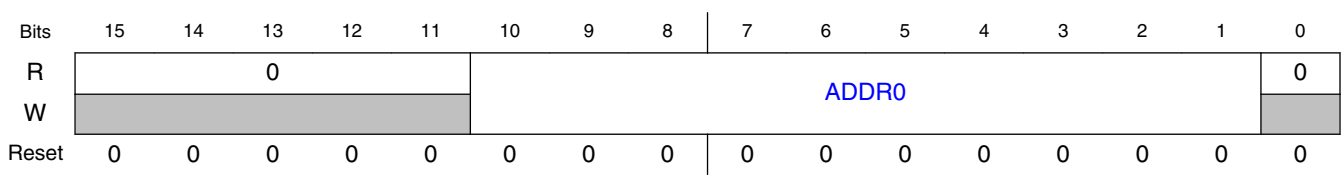
31.4.1.28.1 Offset

Register	Offset
SAMR0	1Ah

31.4.1.28.2 Function

The SAMR0 should only be written when the I2C Slave is disabled.

31.4.1.28.3 Diagram



31.4.1.28.4 Fields

Field	Function
15-11	Reserved

Table continues on the next page...

Field	Function
—	
10-1 ADDR0	Address 0 Value Compared against the received address to detect the Slave Address. • In 10-bit mode, the first address byte is compared to { 11110, ADDR0[10:9] } and the second address byte is compared to ADDR0[8:1] • In 7-bit mode, the address is compared to ADDR0[7:1] •
0 —	Reserved

31.4.1.29 Slave Address Match Register 1 (SAMR1)

31.4.1.29.1 Offset

Register	Offset
SAMR1	1Bh

31.4.1.29.2 Function

The SAMR1 should only be written when the I2C Slave is disabled.

31.4.1.29.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0					ADDR1										0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

31.4.1.29.4 Fields

Field	Function
15-11 —	Reserved
10-1 ADDR1	Address 1 Value Compared against the received address to detect the Slave Address.

Table continues on the next page...

Field	Function
	• In 10-bit mode, the first address byte is compared to { 11110, ADDR1[10:9] } and the second address byte is compared to ADDR1[8:1] • In 7-bit mode, the address is compared to ADDR1[7:1]
0 —	Reserved

31.4.1.30 Slave Address Status Register (SASR)

31.4.1.30.1 Offset

Register	Offset
SASR	1Ch

31.4.1.30.2 Diagram

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

31.4.1.30.3 Fields

Field	Function
15 —	Reserved
14 ANV	Address Not Valid 0b - Received Address (RADDR) is valid 1b - Received Address (RADDR) is not valid
13-11 —	Reserved
10-0 RADDR	Received Address The Received Address updates whenever the AMF is set; the AMF is cleared by reading the Slave Address Status register. • In 7-bit mode, the address byte is store in RADDR[7:0] • In 10-bit mode, the first address byte is { 11110, RADDR[10:9], RADDR[0] } and the second address byte is RADDR[8:1]. The R/W bit is therefore always stored in RADDR[0] • When ACKSTALL is set, if the first address byte matches in 10-bit mode, then the first address byte is stored in RADDR[7:0] so software can read the Received Address before writing the Transmit

Field	Function
	ACK. The Received Address then updates with the full 10-bit address if the second address byte matches.

31.4.1.31 Slave Transmit ACK Register (STAR)

31.4.1.31.1 Offset

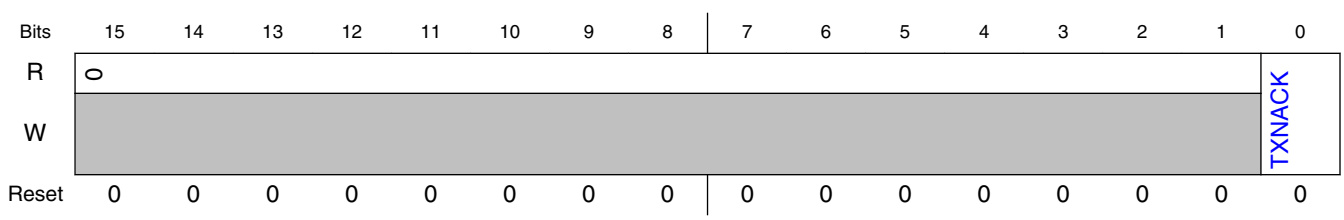
Register	Offset
STAR	1Dh

31.4.1.31.2 Function

The Slave Transmit ACK Register can only be written when the ACK SCL Stall bit is set in Slave Configuration Register 1 (SCFGR0[ACKSTALL].

- The ACKSTALL bit will enable clock stretching during the ACK/NACK bit slot, and during this time, the STAR register can be written by software.
- The logic ensures that the clock stretching continues for at least 1 bus clock cycle after the STAR register is updated.
- This clock stretching time can be extended more using the Clock Hold Time field (SCFGR2[CLKHOLD], Slave Configuration Register 2).

31.4.1.31.3 Diagram



31.4.1.31.4 Fields

Field	Function
15-1 —	Reserved
0	Transmit NACK

Field	Function
TXNACK	After receiving each word, software can transmit either an ACK (logic 0) or a NACK (logic 1); Transmit NACK selects which to use: ACK or NACK. • When ACKSTALL is set, a Transmit NACK must be written once for each matching address byte and each received word. ACKSTALL must be set, because that will stall the data transfer until software reads the received word (and decides whether to respond with an ACK or NACK). • To configure the default ACK/NACK, Transmit NACK can also be written when LPI2C Slave is disabled or idle. 0b - Write a Transmit ACK for each received word 1b - Write a Transmit NACK for each received word

31.4.1.32 Slave Transmit Data Register (STDR)

31.4.1.32.1 Offset

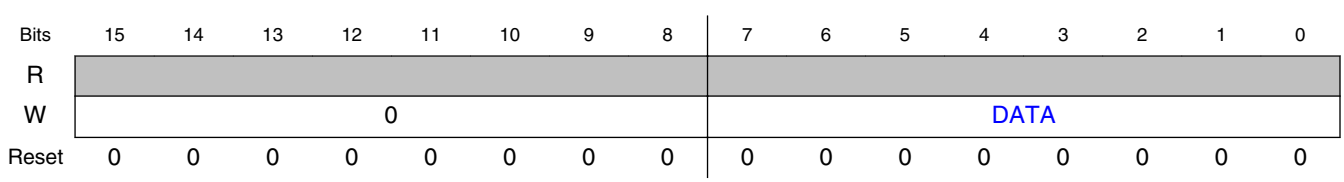
Register	Offset
STDR	1Eh

31.4.1.32.2 Function

Clock stretching (enabled or disabled) affects when the transmit data is transferred. The TXDSTALL bit will enable clock stretching during the 1st data bit of a slave-transmit transfer.

- **If clock stretching is enabled (TXDSTALL=1)**, then the transmit data transfer is stalled until the Slave Transmit register (STDR) is updated. Clock stretching is extended by at least 1 bus clock cycle after the Slave Transmit Data Register (STDR) is updated, and clock stretching can be delayed even more using the Clock Hold Time field (SCFGR2[CLKHOLD], Slave Configuration Register 2).
- **If clock stretching is disabled (TXDSTALL=0)**, then the transmit data should be written before the start of the slave-transmit transfer; otherwise (i.e., if the transmit data is not written before the start of the slave-transmit transfer), the FIFO Error Flag (SSR[FEF]) will be set.

31.4.1.32.3 Diagram



31.4.1.32.4 Fields

Field	Function
15-8 —	Reserved
7-0 DATA	Transmit Data Writing to the Slave Transmit Data Register (STDR) will store I2C slave transmit data *in* the Slave Transmit Data Register

31.4.1.33 Slave Receive Data Register (SRDR)

31.4.1.33.1 Offset

Register	Offset
SRDR	1Fh

31.4.1.33.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SOF	RXEMPTY	0						DATA							
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

31.4.1.33.3 Fields

Field	Function
15 SOF	Start Of Frame 0b - Indicates this is not the first data word since a (repeated) START or STOP condition 1b - Indicates this is the first data word since a (repeated) START or STOP condition
14 RXEMPTY	RX Empty 0b - The Receive Data Register is not empty 1b - The Receive Data Register is empty
13-8 —	Reserved

Table continues on the next page...

Field	Function
7-0 DATA	Receive Data Reading the Slave Receive Data Register returns the data received by the I2C slave

Chapter 32
General-Purpose Input/Output (GPIO)

32.1 Chip-specific information for this module

32.1.1 GPIO Port D[4:0] configuration

By default, pins 0-4 of GPIO port D are configured differently from other GPIO port pins. These five pins have specific purposes:

- The GPIOD4 pin acts as the RESETB input by default.
- The GPIOD3 pin acts as the TMS input by default.
- The GPIOD2 pin acts as the TCK input by default.
- The GPIOD1 pin acts as the TDO output by default.
- The GPIOD0 pin acts as the TDI input by default.

32.2 Overview

The general-purpose input/output (GPIO) module allows direct read or write access to pin values or the ability to assign a pin to be used as an external interrupt. GPIO pins are multiplexed with other peripherals on the package. The device's data sheet specifies the assigned GPIO ports and the multiplexed pin package.

A GPIO pin can be configured in different operation modes:

- As GPIO input with, or without, pull resistor
- As GPIO output with push-pull mode or open-drain mode
- As a peripheral pin when multiplexed with another module

GPIOs are placed on the device in groups of one to sixteen bits, called ports and designated as A, B, C, and so on. Refer to the device's data sheet for the specific definition of each of the GPIO ports on the chip.

32.2.1 Features

The GPIO module's design includes these features:

- Individual control for each pin to be in either peripheral mode or GPIO mode
- Individual direction control for each pin in GPIO mode
- Individual pull resistor enable control for each pin in either peripheral mode or GPIO mode
- Individual pull resistor type selection for each pin in either peripheral mode or GPIO mode
- Individual selection of output push-pull mode or open-drain mode for each pin
- Individual output drive strength control (high-power mode or low-power mode) for each pin
- Ability to monitor each pin's logic values when pin is in either GPIO mode or peripheral mode by using raw data (RDATA) register
- Each pin has the ability to generate an interrupt with programmable rising or falling edge and software interrupt
- Output edge slew rate control for each pin to reduce switch noise

32.2.2 Modes of Operation

The GPIO module can operate in two major modes:

- Peripheral mode: The peripheral module controls the pin. However, if the pin is not configured as an analog input, then output drive strength, edge slew rate control, push-pull or open-drain output, and pull resistor enable and type select remain controlled by GPIO registers.
- GPIO mode: The GPIO module controls the pin. Any data output and input can be written to or read from GPIO data registers. Pull resistor enables and type select are controlled by a GPIO register. GPIO pins can generate the edge interrupt and insert the software interrupt.

32.3 Memory Map and Registers

Each GPIO register contains up to 16 bits, each of which performs an identical function for one of the GPIO pins controlled by that GPIO port. However, initial operating conditions at reset can vary; some GPIO modes are on by default, and others are not.

NOTE

The reset value of these registers may differ depending on the reset function of specific pins. Refer to the device's data sheet.

For simplicity, each GPIO port's registers appear with the same width of 16 bits, corresponding to 16 pins. The actual number of pins per port (and therefore the number of usable control bits per port register) is as follows:

- Port A: 8
- Port B: 8
- Port C: 16
- Port D: 5
- Port E: 8
- Port F: 9

Any register bit for which no corresponding port pin exists, such as bits 15-8 of every GPIOA register, is reserved and always reads as 0.

GPIO memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E200	GPIO Pull Resistor Enable Register (GPIOA_PUR)	16	R/W	See section	32.3.1/1000
E201	GPIO Data Register (GPIOA_DR)	16	R/W	Undefined	32.3.2/1000
E202	GPIO Data Direction Register (GPIOA_DDR)	16	R/W	0000h	32.3.3/1001
E203	GPIO Peripheral Enable Register (GPIOA_PER)	16	R/W	See section	32.3.4/1002
E204	GPIO Interrupt Assert Register (GPIOA_IAR)	16	R/W	0000h	32.3.5/1002
E205	GPIO Interrupt Enable Register (GPIOA_IENR)	16	R/W	0000h	32.3.6/1003
E206	GPIO Interrupt Polarity Register (GPIOA_IPOLR)	16	R/W	0000h	32.3.7/1003
E207	GPIO Interrupt Pending Register (GPIOA_IPR)	16	R	0000h	32.3.8/1004
E208	GPIO Interrupt Edge Sensitive Register (GPIOA_IESR)	16	R/W	0000h	32.3.9/1005
E209	GPIO Push-Pull Mode Register (GPIOA_PPMODE)	16	R/W	See section	32.3.10/ 1005
E20A	GPIO Raw Data Register (GPIOA_RAWDATA)	16	R	Undefined	32.3.11/ 1006
E20B	GPIO Drive Strength Control Register (GPIOA_DRIVE)	16	R/W	See section	32.3.12/ 1007
E20C	GPIO Pull Resistor Type Select (GPIOA_PUS)	16	R/W	See section	32.3.13/ 1007
E20D	Slew Rate Control Register (GPIOA_SRE)	16	R/W	See section	32.3.14/ 1008
E210	GPIO Pull Resistor Enable Register (GPIOB_PUR)	16	R/W	See section	32.3.1/1000
E211	GPIO Data Register (GPIOB_DR)	16	R/W	Undefined	32.3.2/1000
E212	GPIO Data Direction Register (GPIOB_DDR)	16	R/W	0000h	32.3.3/1001
E213	GPIO Peripheral Enable Register (GPIOB_PER)	16	R/W	See section	32.3.4/1002

Table continues on the next page...

GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E214	GPIO Interrupt Assert Register (GPIOB_IAR)	16	R/W	0000h	32.3.5/1002
E215	GPIO Interrupt Enable Register (GPIOB_IENR)	16	R/W	0000h	32.3.6/1003
E216	GPIO Interrupt Polarity Register (GPIOB_IPOLR)	16	R/W	0000h	32.3.7/1003
E217	GPIO Interrupt Pending Register (GPIOB_IPR)	16	R	0000h	32.3.8/1004
E218	GPIO Interrupt Edge Sensitive Register (GPIOB_IESR)	16	R/W	0000h	32.3.9/1005
E219	GPIO Push-Pull Mode Register (GPIOB_PPMODE)	16	R/W	See section	32.3.10/ 1005
E21A	GPIO Raw Data Register (GPIOB_RAWDATA)	16	R	Undefined	32.3.11/ 1006
E21B	GPIO Drive Strength Control Register (GPIOB_DRIVE)	16	R/W	See section	32.3.12/ 1007
E21C	GPIO Pull Resistor Type Select (GPIOB_PUS)	16	R/W	See section	32.3.13/ 1007
E21D	Slew Rate Control Register (GPIOB_SRE)	16	R/W	See section	32.3.14/ 1008
E220	GPIO Pull Resistor Enable Register (GPIOC_PUR)	16	R/W	See section	32.3.1/1000
E221	GPIO Data Register (GPIOC_DR)	16	R/W	Undefined	32.3.2/1000
E222	GPIO Data Direction Register (GPIOC_DDR)	16	R/W	0000h	32.3.3/1001
E223	GPIO Peripheral Enable Register (GPIOC_PER)	16	R/W	See section	32.3.4/1002
E224	GPIO Interrupt Assert Register (GPIOC_IAR)	16	R/W	0000h	32.3.5/1002
E225	GPIO Interrupt Enable Register (GPIOC_IENR)	16	R/W	0000h	32.3.6/1003
E226	GPIO Interrupt Polarity Register (GPIOC_IPOLR)	16	R/W	0000h	32.3.7/1003
E227	GPIO Interrupt Pending Register (GPIOC_IPR)	16	R	0000h	32.3.8/1004
E228	GPIO Interrupt Edge Sensitive Register (GPIOC_IESR)	16	R/W	0000h	32.3.9/1005
E229	GPIO Push-Pull Mode Register (GPIOC_PPMODE)	16	R/W	See section	32.3.10/ 1005
E22A	GPIO Raw Data Register (GPIOC_RAWDATA)	16	R	Undefined	32.3.11/ 1006
E22B	GPIO Drive Strength Control Register (GPIOC_DRIVE)	16	R/W	See section	32.3.12/ 1007
E22C	GPIO Pull Resistor Type Select (GPIOC_PUS)	16	R/W	See section	32.3.13/ 1007
E22D	Slew Rate Control Register (GPIOC_SRE)	16	R/W	See section	32.3.14/ 1008
E230	GPIO Pull Resistor Enable Register (GPIOD_PUR)	16	R/W	See section	32.3.1/1000
E231	GPIO Data Register (GPIOD_DR)	16	R/W	Undefined	32.3.2/1000
E232	GPIO Data Direction Register (GPIOD_DDR)	16	R/W	0000h	32.3.3/1001
E233	GPIO Peripheral Enable Register (GPIOD_PER)	16	R/W	See section	32.3.4/1002
E234	GPIO Interrupt Assert Register (GPIOD_IAR)	16	R/W	0000h	32.3.5/1002
E235	GPIO Interrupt Enable Register (GPIOD_IENR)	16	R/W	0000h	32.3.6/1003
E236	GPIO Interrupt Polarity Register (GPIOD_IPOLR)	16	R/W	0000h	32.3.7/1003

Table continues on the next page...

GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E237	GPIO Interrupt Pending Register (GPIOD_IPR)	16	R	0000h	32.3.8/1004
E238	GPIO Interrupt Edge Sensitive Register (GPIOD_IESR)	16	R/W	0000h	32.3.9/1005
E239	GPIO Push-Pull Mode Register (GPIOD_PPMODE)	16	R/W	See section	32.3.10/ 1005
E23A	GPIO Raw Data Register (GPIOD_RAWDATA)	16	R	Undefined	32.3.11/ 1006
E23B	GPIO Drive Strength Control Register (GPIOD_DRIVE)	16	R/W	See section	32.3.12/ 1007
E23C	GPIO Pull Resistor Type Select (GPIOD_PUS)	16	R/W	See section	32.3.13/ 1007
E23D	Slew Rate Control Register (GPIOD_SRE)	16	R/W	See section	32.3.14/ 1008
E240	GPIO Pull Resistor Enable Register (GPIOE_PUR)	16	R/W	See section	32.3.1/1000
E241	GPIO Data Register (GPIOE_DR)	16	R/W	Undefined	32.3.2/1000
E242	GPIO Data Direction Register (GPIOE_DDR)	16	R/W	0000h	32.3.3/1001
E243	GPIO Peripheral Enable Register (GPIOE_PER)	16	R/W	See section	32.3.4/1002
E244	GPIO Interrupt Assert Register (GPIOE_IAR)	16	R/W	0000h	32.3.5/1002
E245	GPIO Interrupt Enable Register (GPIOE_IENR)	16	R/W	0000h	32.3.6/1003
E246	GPIO Interrupt Polarity Register (GPIOE_IPOLR)	16	R/W	0000h	32.3.7/1003
E247	GPIO Interrupt Pending Register (GPIOE_IPR)	16	R	0000h	32.3.8/1004
E248	GPIO Interrupt Edge Sensitive Register (GPIOE_IESR)	16	R/W	0000h	32.3.9/1005
E249	GPIO Push-Pull Mode Register (GPIOE_PPMODE)	16	R/W	See section	32.3.10/ 1005
E24A	GPIO Raw Data Register (GPIOE_RAWDATA)	16	R	Undefined	32.3.11/ 1006
E24B	GPIO Drive Strength Control Register (GPIOE_DRIVE)	16	R/W	See section	32.3.12/ 1007
E24C	GPIO Pull Resistor Type Select (GPIOE_PUS)	16	R/W	See section	32.3.13/ 1007
E24D	Slew Rate Control Register (GPIOE_SRE)	16	R/W	See section	32.3.14/ 1008
E250	GPIO Pull Resistor Enable Register (GPIOF_PUR)	16	R/W	See section	32.3.1/1000
E251	GPIO Data Register (GPIOF_DR)	16	R/W	Undefined	32.3.2/1000
E252	GPIO Data Direction Register (GPIOF_DDR)	16	R/W	0000h	32.3.3/1001
E253	GPIO Peripheral Enable Register (GPIOF_PER)	16	R/W	See section	32.3.4/1002
E254	GPIO Interrupt Assert Register (GPIOF_IAR)	16	R/W	0000h	32.3.5/1002
E255	GPIO Interrupt Enable Register (GPIOF_IENR)	16	R/W	0000h	32.3.6/1003
E256	GPIO Interrupt Polarity Register (GPIOF_IPOLR)	16	R/W	0000h	32.3.7/1003
E257	GPIO Interrupt Pending Register (GPIOF_IPR)	16	R	0000h	32.3.8/1004
E258	GPIO Interrupt Edge Sensitive Register (GPIOF_IESR)	16	R/W	0000h	32.3.9/1005

Table continues on the next page...

GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E259	GPIO Push-Pull Mode Register (GPIOF_PPMODE)	16	R/W	See section	32.3.10/ 1005
E25A	GPIO Raw Data Register (GPIOF_RAWDATA)	16	R	Undefined	32.3.11/ 1006
E25B	GPIO Drive Strength Control Register (GPIOF_DRIVE)	16	R/W	See section	32.3.12/ 1007
E25C	GPIO Pull Resistor Type Select (GPIOF_PUS)	16	R/W	See section	32.3.13/ 1007
E25D	Slew Rate Control Register (GPIOF_SRE)	16	R/W	See section	32.3.14/ 1008

32.3.1 GPIO Pull Resistor Enable Register (GPIOx_PUR)

This read/write register is for internal pull resistor enabling and disabling. If the pin is configured as an output, this register is not used. Unimplemented bits read as 0.

The pull resistor is intended only to drive an undriven input pin to a known state. It is characteristically a very weak pull.

Address: Base address + 0h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	PU															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value is as follows:
  - GPIOD_PUR: 001Dh
  - any other GPIOn_PUR: 0000h

GPIOx_PUR field descriptions

Field	Description
PU	Pull Resistor Enable Bits 0 Pull resistor is disabled 1 Pull resistor is enabled

32.3.2 GPIO Data Register (GPIOx_DR)

This register holds data that comes either from the pin or the data bus. In other words, the register is the data interface between the pin and the data bus.

Data written to this register appears on the pins if the pins are configured as GPIO output. Data read from this register is the same as the read state on the pins if those pins are configured as GPIO input.

When the device comes out of reset, GPIO pins are configured as inputs with internal pull disabled. As a result, the reset value of DR's bits is undefined. However, if you configure the GPIO as outputs (the DDR[DD] bit is 1), then the default value of the corresponding DR[D] bit is 0.

Address: Base address + 1h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	D															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

GPIOx_DR field descriptions

Field	Description
D	Data Bits

32.3.3 GPIO Data Direction Register (GPIOx_DDR)

This read/write register configures the state of the pin as either input or output when the pin is configured as GPIO (the corresponding bit in the GPIO peripheral enable register is set to 0). When the register is set to 0, the pin is an input. When the register is set to 1, the pin is an output.

Address: Base address + 2h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	DD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_DDR field descriptions

Field	Description
DD	Data Direction Bits 0 Pin is an input 1 Pin is an output

32.3.4 GPIO Peripheral Enable Register (GPIOx_PER)

This read/write register determines the configuration of the GPIO pins.

When the Peripheral Enable bitfield value in this register is 1, the GPIO module is configured for peripheral mode. In this mode, a peripheral controls the GPIO pin, and the data transfer direction depends on the function of the peripheral.

When the Peripheral Enable bitfield value is 0, the pin is configured for GPIO mode. In this mode, the corresponding GPIO Data Direction register controls the data flow.

If write protection (via the SIM PROT register) is implemented on an individual chip, then this register value cannot be changed after the write protect signal has been asserted.

Address: Base address + 3h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	PE															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value is as follows:
  - GPIOD_PER: 001Fh
  - any other GPIOn_PER: 0000h

GPIOx_PER field descriptions

Field	Description
PE	Peripheral Enable Bits 0 Pin is for GPIO (GPIO mode) 1 Pin is for peripheral (peripheral mode)

32.3.5 GPIO Interrupt Assert Register (GPIOx_IAR)

This read/write register is only for software testing of a software interrupt capability. When the bit is set to 1, an interrupt is asserted. The interrupt is generated continually until this bit is cleared. Clear the bits in the register by writing 0s.

Address: Base address + 4h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	IA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_IAR field descriptions

Field	Description
IA	Interrupt Assert Bits 0 Deassert software interrupt 1 Assert software interrupt

32.3.6 GPIO Interrupt Enable Register (GPIOx_IENR)

This read/write register enables or disables the edge interrupt from each GPIO pin. Set a bit to 1 to enable the interrupt for the associated GPIO pin. The interrupt is recorded in the corresponding GPIO Interrupt Pending register.

Address: Base address + 5h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	IEN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_IENR field descriptions

Field	Description
IEN	Interrupt Enable Bits 0 External Interrupt is disabled 1 External Interrupt is enabled

32.3.7 GPIO Interrupt Polarity Register (GPIOx_IPOLR)

This read/write register is used for polarity detection caused by any external interrupts. The interrupt at the pin is active low when this register is set to 1 (falling edge causes the interrupt). The interrupt seen at the pin is active high when this register is set to 0 (rising edge causes the interrupt).

Address: Base address + 6h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	IPOL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_IPOLR field descriptions

Field	Description
IPOL	Interrupt Polarity Bits 0 Interrupt occurred on rising edge 1 Interrupt occurred on falling edge

32.3.8 GPIO Interrupt Pending Register (GPIOx_IPR)

This read-only register is used to record any incoming interrupts. The user can read this register to determine which pin has caused the interrupt. This register can be cleared by writing 1s into the GPIO Interrupt Edge Sensitive register if the interrupt is caused by a pin, or by writing 0s into the GPIO Interrupt Assert register if the interrupt is caused by software.

Address: Base address + 7h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	IP															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_IPR field descriptions

Field	Description
IP	Interrupt Pending Bits 0 No Interrupt 1 Interrupt occurred

32.3.9 GPIO Interrupt Edge Sensitive Register (GPIOx_IESR)

When an edge is detected by the edge detector circuit and the GPIO Interrupt Enable register's field is set to 1, this register's field records the interrupt. This read/write register clears the corresponding Interrupt Pending bit by writing 1 to the appropriate Interrupt Edge Sensitive bit. Writing 0 to an Interrupt Edge Sensitive bit is ignored.

Address: Base address + 8h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	IES															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_IESR field descriptions

Field	Description
IES	Interrupt Edge-Sensitive Bits 0 No edge detected if read; no effect if writing 1 An edge detected if read; clear corresponding Interrupt Pending bit if writing

32.3.10 GPIO Push-Pull Mode Register (GPIOx_PPMODE)

This register can be used to explicitly set each output driver to either push-pull or open-drain mode. If write protection (via the SIM PROT register) is implemented on an individual device, then this register value cannot be changed after the write protect signal is asserted.

NOTE

Open-drain mode can be used to tri-state any pin on the GPIO port without switching that pin to input mode. This capability is useful for some applications, including a keypad interface.

Address: Base address + 9h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	PPMODE															
Reset	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*

* Notes:
- The reset value of the register for each port is as follows:
  - GPIOA_PPMODE: 00FFh
  - GPIOB_PPMODE: 00FFh
  - GPIOC_PPMODE: FFFFh
  - GPIOD_PPMODE: 001Fh

- GPIOE_PPMODE: 00FFh
- GPIOF_PPMODE: 01FFh

GPIOx_PPMODE field descriptions

Field	Description
PPMODE	Push-Pull Mode Bits 0 Open Drain Mode 1 Push-Pull Mode

32.3.11 GPIO Raw Data Register (GPIOx_RAWDATA)

This read-only register gives the DSC direct access to the logic values on each GPIO pin, even when pins are not in GPIO mode. Values are not clocked and are subject to change at any time. Read several times to ensure a stable value. The reset value of this register depends on the default PIN state.

Address: Base address + Ah offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RAW_DATA															
Write																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

GPIOx_RAWDATA field descriptions

Field	Description
RAW_DATA	Raw Data Bits

32.3.12 GPIO Drive Strength Control Register (GPIOx_DRIVE)

This register can be used to explicitly set the drive strength of each output driver. If write protection (via the SIM PROT register) is implemented on an individual device, then this register value cannot be changed after the write protect signal is asserted.

Address: Base address + Bh offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	DRIVE															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value is as follows:
  - GPIOD_DRIVE: 000Ah
  - any other GPIOn_DRIVE: 0000h

GPIOx_DRIVE field descriptions

Field	Description
DRIVE	Drive Strength Selector Bits 0 Low drive strength 1 High drive strength

32.3.13 GPIO Pull Resistor Type Select (GPIOx_PUS)

This register can be used to explicitly set the pull resistor type for each GPIO.

Address: Base address + Ch offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	PUS															
Reset	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*

* Notes:
- The reset value of the register for each port is as follows:
  - GPIOA_PUS: 00FFh
  - GPIOB_PUS: 00FFh
  - GPIOC_PUS: FFFFh
  - GPIOD_PUS: 001Bh
  - GPIOE_PUS: 00FFh
  - GPIOF_PUS: 01FFh

GPIOx_PUS field descriptions

Field	Description
PUS	Pull Resistor Type Select Bits Note that the pull resistor type is ignored in open-drain mode (PPMODE==0) and, if the pull resistor is enabled, a pullup resistor is used. 0 Pulldown resistor 1 Pullup resistor

32.3.14 Slew Rate Control Register (GPIOx_SRE)

This register determines if output slew rate control is enabled for the associated GPIO port pin.

Address: Base address + Dh offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SRE															
Reset	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*

* Notes:
- The reset value of the register for each port is as follows:
  - GPIOA_SRE: 00FFh
  - GPIOB_SRE: 00FFh
  - GPIOC_SRE: FFFFh
  - GPIOD_SRE: 0015h
  - GPIOE_SRE: 00FFh
  - GPIOF_SRE: 01FFh

GPIOx_SRE field descriptions

Field	Description
SRE	Slew Rate Enable 0 Slew rate is enabled (the turn-on time of the output transistor is faster) 1 Slew rate is disabled (the turn-on time of the output transistor is slower)

32.4 Functional Description

The following block diagram illustrates the logic associated with just one of the bits in each GPIO port. Each GPIO pin can be configured as:

- An input, with or without pull resistor functions
- An edge interrupt
- An output with push-pull or open-drain mode

The GPIO's pull resistor is enabled by writing to the PUR register. The resistor type is selected in the PUS register. When the pin is configured for a peripheral function, the pull resistors are controlled by the PUR register and the direction is specified by the peripheral used. If the pin is set to be an output, the pull resistor is disabled. In open-drain mode, the PUS register is ignored, and only the pullup resistor type is available.

A pin may have several peripheral functions, one of which may be an analog function. To access its analog function, the pin must be in peripheral mode with an analog input enabled. Selecting between an analog peripheral or a digital peripheral is controlled by the GPIO Peripheral Select (GPSn) registers in the System Integration Module (SIM). When the GPIO is in peripheral mode and its analog peripheral function is selected, the digital output buffer and pull resistor are disabled. The digital input buffer is also disconnected from the pin so that the digital input does not respond to analog voltages on the pin.

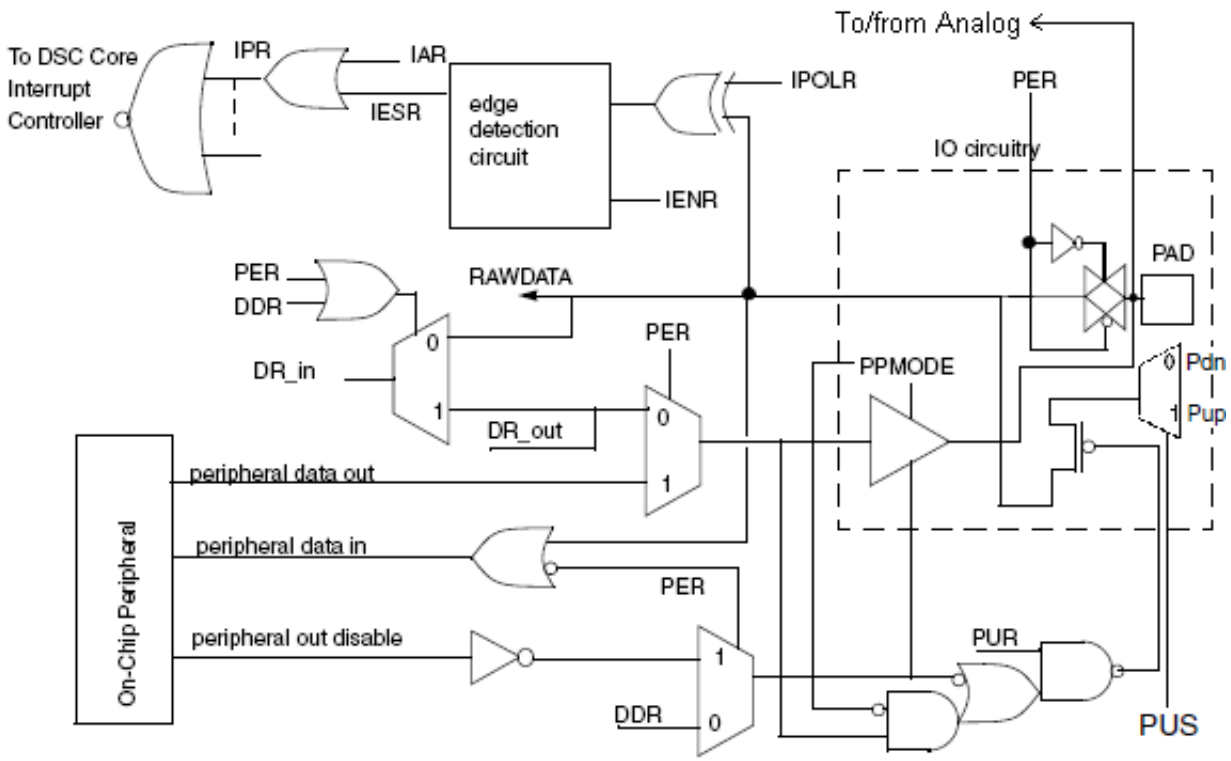


Figure 32-1. Bit View of the GPIO Logic with a Multiplex of Analog Input

32.5 Interrupts

The GPIO module has two types of interrupts:

1. Software interrupt

Write ones to the interrupt assert register to generate the software interrupt. The interrupt pending register records the value of the interrupt assert register. Clear the the interrupt pending register by writing zeroes to the the interrupt assert register.

NOTE

When a software interrupt is asserted, the interrupt polarity register, interrupt edge sensitive register, and the interrupt enable register must be zero to guarantee that the interrupt registered in the interrupt pending register is due only to the interrupt assert register.

2. Hardware interrupt from the pin

   When a GPIO pin is used as an external interrupt source, its IEN bit in the interrupt enable register is set to 1 and the interrupt assert register must be set to 0. The interrupt polarity register must be set to 1 for a falling edge interrupt and to 0 for a rising edge interrupt. When the signal at the pin transitions from high to low or low to high, the value is seen at the interrupt edge sensitive register and recorded by the interrupt pending register.

The interrupt signals in each port are ORed together, presenting only a single interrupt per port to the interrupt controller. The interrupt service routine must then check the contents of the interrupt pending register to determine which pin(s) caused the interrupt. External interrupt sources do not need to remain asserted because the detection mechanism is edge sensitive.

32.6 Clocks and Resets

The GPIO module runs at standard system bus speeds and assumes reset states as defined in the device data sheet. Reset occurs whenever any source of system reset occurs (POR, external reset, COP reset, and so on).

Appendix A
Release Notes

A.1 Revision history

The following table provides a revision history for this document.

Table A-1. Revision history

Rev.	Date	Substantial Changes
2	12/2023	Initial public release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Suitability for use in automotive applications — This NXP product has been qualified for use in automotive applications. If this product is used by customer in the development of, or for incorporation into, products or services (a) used in safety critical applications or (b) in which failure could lead to death, personal injury, or severe physical or environmental damage (such products and services hereinafter referred to as "Critical Applications"), then customer makes the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, safety, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. As such, customer assumes all risk related to use of any products in Critical Applications and NXP and its suppliers shall not be liable for any such use by customer. Accordingly, customer will indemnify and hold NXP harmless from any claims, liabilities, damages and associated costs and expenses (including attorneys' fees) that NXP may incur related to customer's incorporation of any product in a Critical Application.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

CodeWarrior — is a trademark of NXP B.V.

Freescale — is a trademark of NXP B.V.

Kinetis — is a trademark of NXP B.V.

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 12/2023
Document identifier: MWCT2XX2ARM