MC9S12XHZ512

Data Sheet

Covers
MC9S12XHZ384, MC9S12XHZ256

HCS12X Microcontrollers

MC9S12XHZ512
Rev. 1.06
10/2010

freescale.com

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to:

http://freescale.com/

The following revision history table summarizes changes contained in this document. This document contains information for all constituent modules, with the exception of the S12X CPU. For S12X CPU information please refer to CPU12XV2 in the CPU12/CPU12X Reference Manual.

Revision History

Date	Revision Level	Description
January 5, 2006	01.00	New Book
April 20, 2006	01.01	Updated block guide versions
July 28, 2006	01.02	Made minor corrections
January 8, 2007	01.03	Added MC9S12XHZ384 and MC9S12XHZ256
August 20, 2007	01.04	Updated slew rates
November 4, 2008	01.05	Corrected typos on pinout diagram
October 14, 2010	01.06	Added PartID. Minor updates to ECT, SCI, IIC and XGATE sections.

List of Chapters

Table of Contents

Chapter 1
MC9S12XHZ Family Device Overview

Chapter 2
Port Integration Module (S12XHZPIMV1)

Chapter 3
512 Kbyte Flash Module (S12XFTX512K4V3)

Chapter 4
4 Kbyte EEPROM Module (S12XEETX4KV2)

Chapter 5
XGATE (S12XGATEV2)

Chapter 6
Security (S12X9SECV2)

Chapter 7
Clocks and Reset Generator (S12CRGV6)

Chapter 8
Pierce Oscillator (S12XOSCLCPV1)

Chapter 9
Analog-to-Digital Converter (ATD10B16CV4) Block Description

Chapter 10
Liquid Crystal Display (LCD32F4BV1) Block Description

Chapter 11
Motor Controller (MC10B12CV2) Block Description

Chapter 12
Stepper Stall Detector (SSDV1) Block Description

Chapter 13
Inter-Integrated Circuit (IICV3) Block Description

Chapter 14
Freescale’s Scalable Controller Area Network (S12MSCANV3)

Chapter 15
Serial Communication Interface (S12SCIV5)

Chapter 16
Serial Peripheral Interface (S12SPIV4)

Chapter 17
Periodic Interrupt Timer (S12PIT24B4CV1)

Chapter 18
Pulse-Width Modulator (S12PWM8B8CV1)

Chapter 19
Enhanced Capture Timer (ECT16B8CV3)

Chapter 20
Voltage Regulator (S12VREG3V3V5)

Chapter 21
Background Debug Module (S12XBDMV2)

Chapter 22
S12X Debug (S12XDBGV3) Module

Chapter 23
External Bus Interface (S12XEBIV3)

Chapter 24
Interrupt (S12XINTV1)

Chapter 25
Memory Mapping Control (S12XMMCV3)

Appendix A
Electrical Characteristics

Appendix B
Package Information

Appendix C
PCB Layout Guidelines

Appendix D
Ordering Information

Appendix E
Detailed Register Map

Chapter 1
MC9S12XHZ Family Device Overview

1.1 Introduction

Targeted at automotive instrumentation applications, the MC912XHZ family of microcontrollers is a fully pin-compatible extension to the existing MC9S12HZ family. It offers not only a larger memory but also incorporates all the architectural benefits of the new S12X-based family to deliver significantly higher performance. The MC9S12XHZ family retains the low cost, power consumption, EMC and code-size efficiency advantages currently associated with the MC9S12 products.

Based around S12X core, the MC912XHZ family runs 16-bit wide accesses without wait states for all peripherals and memories. The MC912XHZ family also features a new flexible interrupt handler, which allows multilevel nested interrupts.

The MC912XHZ family features the performance boosting XGATE co-processor. The XGATE is programmable in “C” language and runs at twice the bus frequency of the S12. Its instruction set is optimized for data movement, logic and bit manipulation instructions. Any peripheral module can be serviced by the XGATE.

The MC912XHZ family contains up to 512K bytes of Freescale Semiconductor’s industry leading, full automotive qualified Split-Gate Flash memory, with 4K bytes of additional integrated data EEPROM and up to 32K bytes of static RAM.

The MC912XHZ family features a 32x4 liquid crystal display (LCD) controller/driver and a motor pulse width modulator (MC) consisting of up to 24 high current outputs suited to drive six stepper motors with stall detectors (SSD) to simultaneously calibrate the pointer reset position of each motor. It also features two MSCAN modules, each with a FIFO receiver buffer arrangement, and input filters optimized for Gateway applications handling numerous message identifiers.

In addition, the MC912XHZ family is composed of standard on-chip peripherals including two asynchronous serial communications interfaces (SCI0 and SCI1), one serial peripheral interface (SPI), two IIC-bus interface (IIC0 and IIC1), an 8-channel 16-bit enhanced capture timer (ECT), a 16-channel, 10-bit analog-to-digital converter (ADC), and one 8-channel pulse width modulator (PWM).

The inclusion of a PLL circuit allows power consumption and performance to be adjusted to suit operational requirements. The new fast-exit from STOP mode feature can further improve system power consumption. In addition to the I/O ports available in each module, 8 general-purpose I/O pins are available with interrupt and wake-up capability from stop or wait mode.

The MC912XHZ family is available in 112-pin LQFP and 144-pin LQFP packages. The 144-pin LQFP package option provides a full 16-bit wide non-multiplexed external bus interface.

1.1.1 Features

- HCS12X Core
  - 16-bit HCS12X CPU
    - Upward compatible with MC9S12 instruction set
    - Interrupt stacking and programmer's model identical to MC9S12
    - Instruction queue
    - Enhanced indexed addressing
    - Enhanced instruction set
  - EBI (external bus interface)
  - MMC (module mapping control)
  - INT (interrupt controller)
  - DBG (debug module to monitor HCS12X CPU and XGATE bus activity)
  - BDM (background debug mode)
- XGATE (peripheral coprocessor)
  - Parallel processing module off loads the CPU by providing high-speed data processing and transfer
  - Data transfer between Flash EEPROM, RAM, peripheral modules, and I/O ports
- Memory
  - 512K, 384K, 256K byte Flash EEPROM
  - 4K byte EEPROM
  - 32K, 28K, 16K byte RAM
- CRG (clock and reset generator)
  - Low noise/low power Pierce oscillator
  - PLL
  - COP watchdog
  - Real time interrupt
  - Clock monitor
  - Fast wake-up from stop mode
- Analog-to-digital converter
  - 16 channels, 10-bit resolution
  - External conversion trigger capability
- ECT (enhanced capture timer)
  - 16-bit main counter with 8-bit prescaler
  - 8 programmable input capture or output compare channels
  - Four 8-bit or two 16-bit pulse accumulators
- PIT (periodic interrupt timer)
  - Four timers with independent time-out periods
  - Time-out periods selectable between 1 and 2^{24} bus clock cycles

- 8 PWM (pulse-width modulator) channels
  - Programmable period and duty cycle
  - 8-bit 8-channel or 16-bit 4-channel
  - Separate control for each pulse width and duty cycle
  - Center-aligned or left-aligned outputs
  - Programmable clock select logic with a wide range of frequencies
  - Fast emergency shutdown input
- Two 1-Mbps, CAN 2.0 A, B software compatible modules
  - Five receive and three transmit buffers
  - Flexible identifier filter programmable as 2 x 32 bit, 4 x 16 bit, or 8 x 8 bit
  - Four separate interrupt channels for Rx, Tx, error, and wake-up
  - Low-pass filter wake-up function
  - Loop-back for self-test operation
- Two IIC (Inter-IC bus) Modules
  - Compatible with IIC bus standard
  - Multi-master operation
  - Broadcast mode
- Serial interfaces
  - Two asynchronous serial communication interfaces (SCI) with additional LIN support and selectable IrDA 1.4 return-to-zero-inverted (RZI) format with programmable pulse width
  - Synchronous Serial Peripheral Interface (SPI)
- Liquid crystal display (LCD) driver with variable input voltage
  - Configurable for up to 32 frontplanes and 4 backplanes or general-purpose input or output
  - 5 modes of operation allow for different display sizes to meet application requirements
  - Unused frontplane and backplane pins can be used as general-purpose I/O
- PWM motor controller (MC) with 24 high current drivers
  - Each PWM channel switchable between two drivers in an H-bridge configuration
  - Left, right and center aligned outputs
  - Support for sine and cosine drive
  - Dithering
  - Output slew rate control
- Six stepper stall detectors (SSD)
  - Full step control during return to zero
  - Voltage detector and integrator / sigma delta converter circuit
  - 16-bit accumulator register
  - 16-bit modulus down counter

- On-Chip Voltage Regulator
  - Two parallel, linear voltage regulators with bandgap reference
  - Low-voltage detect (LVD) with low-voltage interrupt (LVI)
  - Power-on reset (POR) circuit
  - 3.3-V–5.5-V operation
  - Low-voltage reset (LVR)
  - Ultra low-power wake-up timer
- 144-pin LQFP and 112-pin LQFP packages
  - I/O lines with 5-V input and drive capability
  - Input threshold on external bus interface inputs switchable for 3.3-V or 5-V operation
  - 5-V A/D converter inputs
  - 8 key wake up interrupts with digital filtering and programmable rising/falling edge trigger
- Operation at 80 MHz equivalent to 40-MHz bus speed
- Development support
  - Single-wire background debug™ mode (BDM)
  - Four on-chip hardware breakpoints

1.1.2 Modes of Operation

User modes:

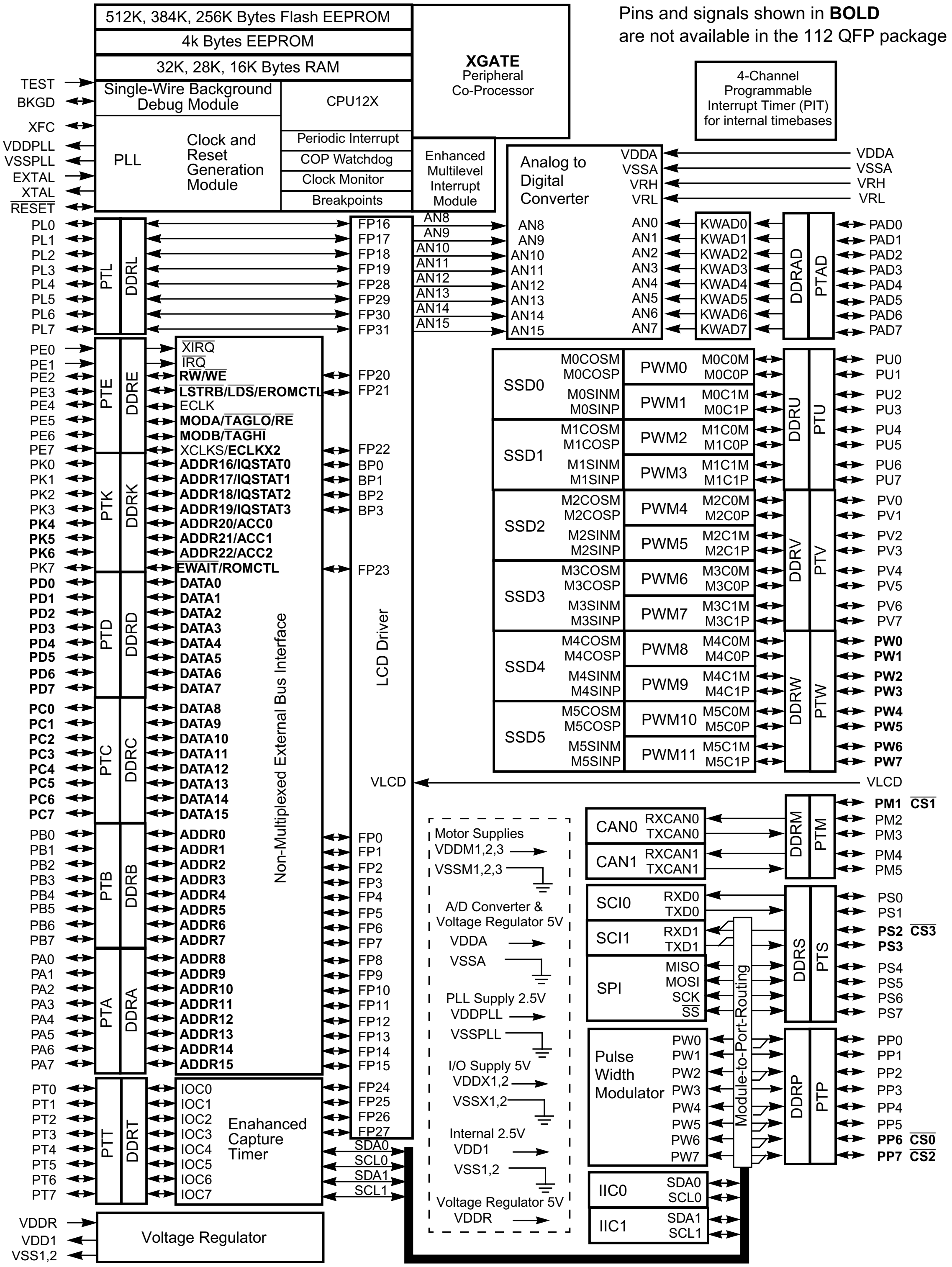
- Normal and emulation operating modes
  - Normal single-chip mode
  - Normal expanded mode
  - Emulation of single-chip mode
  - Emulation of expanded mode
- Special Operating Modes
  - Special single-chip mode with active background debug mode
  - Special test mode (**Freescale use only**)

Low-power modes:

- System stop modes
  - Pseudo stop mode
  - Full stop mode
- System wait mode

1.1.3 Block Diagram

Figure 1-1 shows a block diagram of the MC912XHZ family.



Pins and signals shown in **BOLD** are not available in the 112 QFP package

Figure 1-1. MC9S12XHZ Family Block Diagram

1.1.4 Device Memory Map

Table 1-1 shows the device memory map for the MC912XHZ family.

Unimplemented register space shown in Table 1-1 is not allocated to any module. Writing to these locations have no effect. Read access to these locations returns zero.

Table 1-1. Device Register Memory Map

Address Offset	Module	Size (Bytes)
0x0000–0x0009	PIM (port integration module)	10
0x000A–0x000B	MMC (memory map control)	2
0x000C–0x000D	PIM (port integration module)	2
0x000E–0x000F	EBI (external bus interface)	2
0x0010–0x0017	MMC (memory map control)	8
0x0018–0x0019	Unimplemented	2
0x001A–0x001B	Device ID register	2
0x001C–0x001F	PIM (port integration module)	4
0x0020–0x002F	DBG (debug module)	16
0x0030–0x0031	MMC (memory map control)	2
0x0032–0x0033	PIM (port integration module)	2
0x0034–0x003F	CRG (clock and reset generator)	12
0x0040–0x007F	ECT (enhanced capture timer 16-bit 8-channel)	64
0x0080–0x00AF	ATD (analog-to-digital converter 10-bit 16-channel)	48
0x00B0–0x00BF	INT (interrupt module)	16
0x00C0–0x00C7	IIC0 (inter IC bus)	8
0x00C8–0x00CF	SCI0 (serial communications interface)	8
0x00D0–0x00D7	SCI1 (serial communications interface)	8
0x00D8–0x00DF	SPI (serial peripheral interface)	8
0x00E0–0x00FF	Unimplemented	32
0x0100–0x010F	Flash control registers	16
0x0110–0x011B	EEPROM control registers	12
0x011C–0x011F	MMC (memory map control)	4
0x0120–0x0137	Liquid Crystal Display Driver 32x4 (LCD)	24
0x0138–0x013F	IIC1 (inter IC bus)	8
0x0140–0x017F	CAN0 (scalable CAN)	64
0x0180–0x01BF	CAN1 (scalable CAN)	64
0x01C0–0x01FF	MC (motor controller)	64
0x0200–0x027F	PIM (port integration module)	128
0x0280–0x0287	SSD4 (stepper stall detector)	8
0x0288–0x028F	SSD0 (stepper stall detector)	8
0x0290–0x0297	SSD1 (stepper stall detector)	8
0x0298–0x029F	SSD2 (stepper stall detector)	8

Table 1-1. Device Register Memory Map

Address Offset	Module	Size (Bytes)
0x02A0–0x02A7	SSD3 (stepper stall detector)	8
0x02A8–0x02AF	SSD5 (stepper stall detector)	8
0x02B0–0x02EF	Unimplemented	64
0x02F0–0x02F7	Voltage regulator	8
0x02F8–0x02FF	Unimplemented	8
0x0300–0x0327	PWM (pulse-width modulator 8 channels)	40
0x0328–0x033F	Unimplemented	24
0x0340–0x0367	PIT (periodic interrupt timer)	40
0x0368–0x037F	Unimplemented	24
0x0380–0x03BF	XGATE	64
0x03C0–0x03FF	Unimplemented	64
0x0400–0x07FF	Unimplemented	1024

Figure 1-2 shows the CPU & BDM local address translation to the global memory map. It indicates also the location of the internal resources in the memory map.

Table 1-2. Device Internal Resources

Device	RAMSIZE / RAM_LOW	EEPROMSIZE / EEPROM_LOW	FLASHSIZE0 / FLASH0_LOW	FLASHSIZE1 / FLASH1_HIGH
MC9S12XHZ512	32K / 0x0F_8000	4K / 0x13_F000	256K / 0x7B_FFFF	256K / 0x7C_0000
MC9S12XHZ384	28K / 0x0F_9000	4K / 0x13_F000	128K / 0x79_FFFF	256K / 0x7C_0000
MC9S12XHZ256	16K / 0x0F_C000	4K / 0x13_F000	128K / 0x79_FFFF	128K / 0x7E_0000

Figure 1-3 shows XGATE local address translation to the global memory map. It indicates also the location of used internal resources in the memory map.

Table 1-3. XGATE Resources

Device	XGRAMSIZE / XGRAMLOW	XGFLASHSIZE / XGFLASH_HIGH
MC9S12XHZ512	32K / 0x0F_8000	30K[1] / 0x78_7FFF
MC9S12XHZ384	28K / 0x0F_9000	
MC9S12XHZ256	16K / 0x0F_C000	

[1] This value is calculated by the following formula: (64K - 2K - XGRAMSIZE)

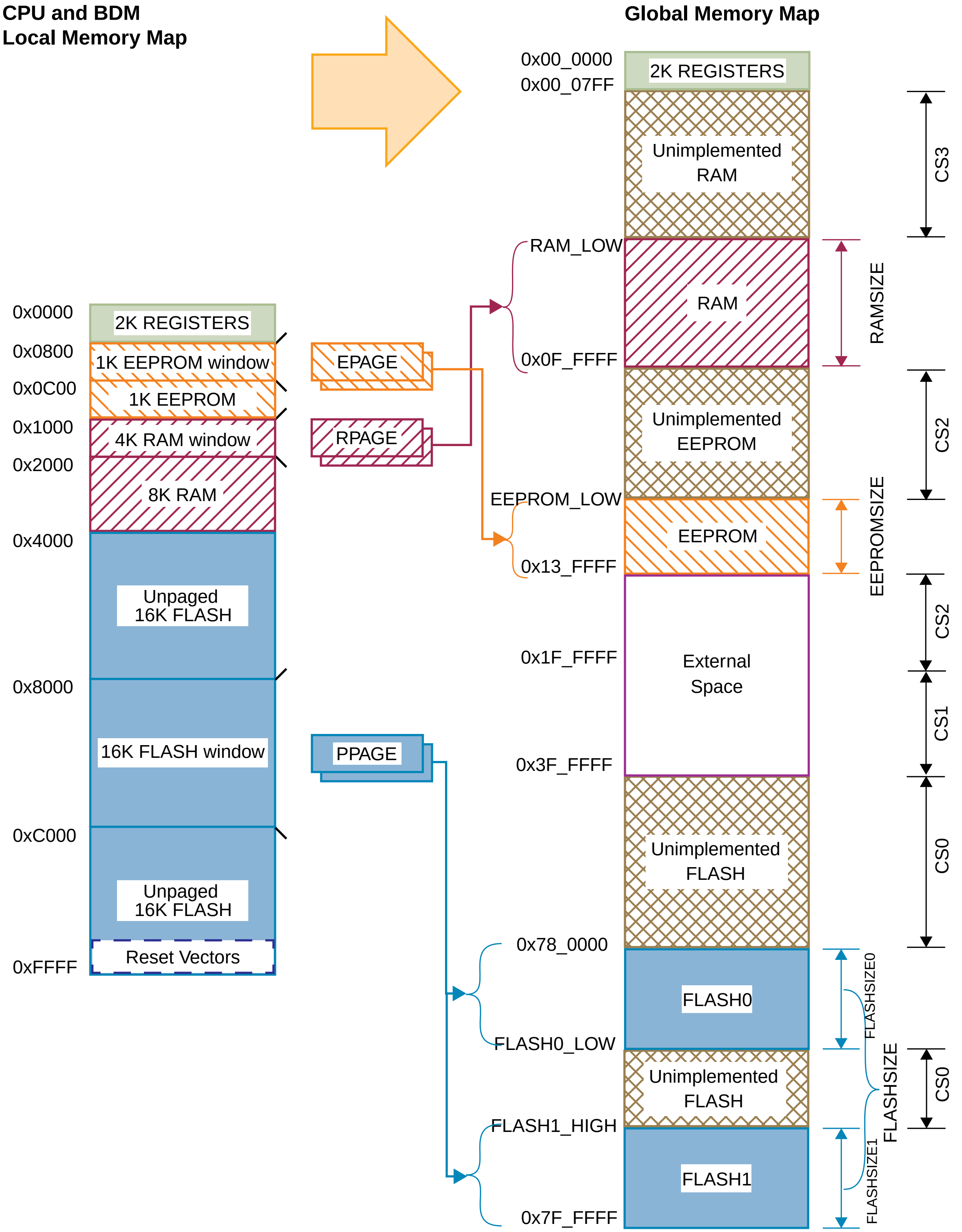


Figure 1-2. MC9S12XHZ Family Global Memory Map

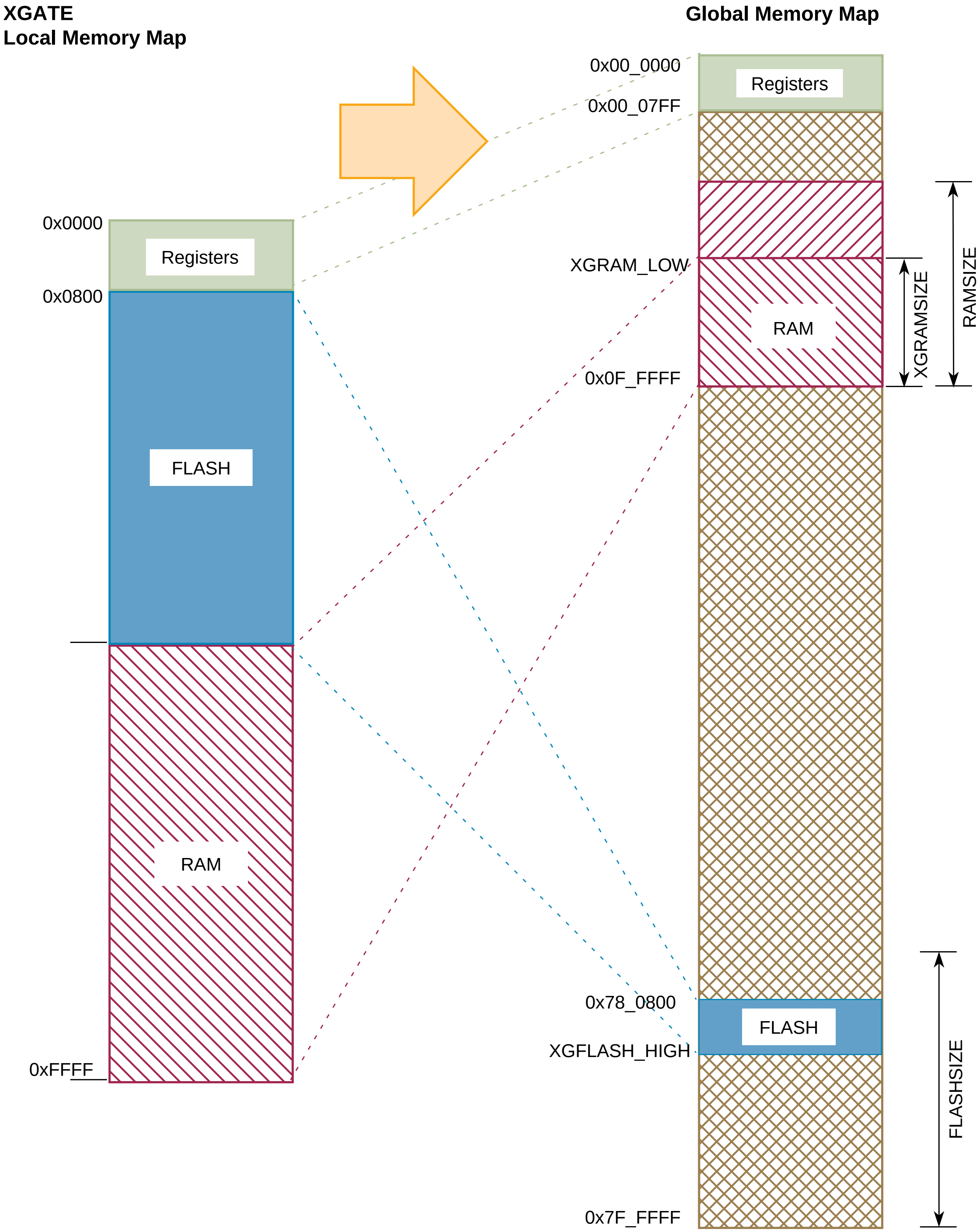


Figure 1-3. XGATE Global Address Mapping

1.1.5 Part ID Assignments

The part ID is located in two 8-bit registers PARTIDH and PARTIDL (addresses 0x001A and 0X001B). The read-only value is a unique part ID for each revision of the chip. Table 1-4 shows the assigned part ID number and mask set number.

Table 1-4. Assigned Part ID Numbers

Device	Mask Set Number	Part ID[1]
MC9S12XHZ512	0M80F 1M80F	0xE400 0xE401
MC9S12XHZ384		
MC9S12XHZ256		

[1] The coding is as follows:
Bit 15-12: Major family identifier
Bit 11-8: Minor family identifier
Bit 7-4: Major mask set revision including fab transfers
Bit 3-0: Minor non-full mask set revision

1.2 Signal Description

This section describes signals that connect off-chip. It includes a pinout diagram, a table of signal properties, and detailed discussion of signals.

1.2.1 Device Pinout

The MC912XHZ family is offered in the following package options:

- 144-pin LQFP with an external bus interface (address/data bus)
- 112-pin LQFP without an external bus interface

Figure 1-4 and Figure 1-5 show the pin assignments.

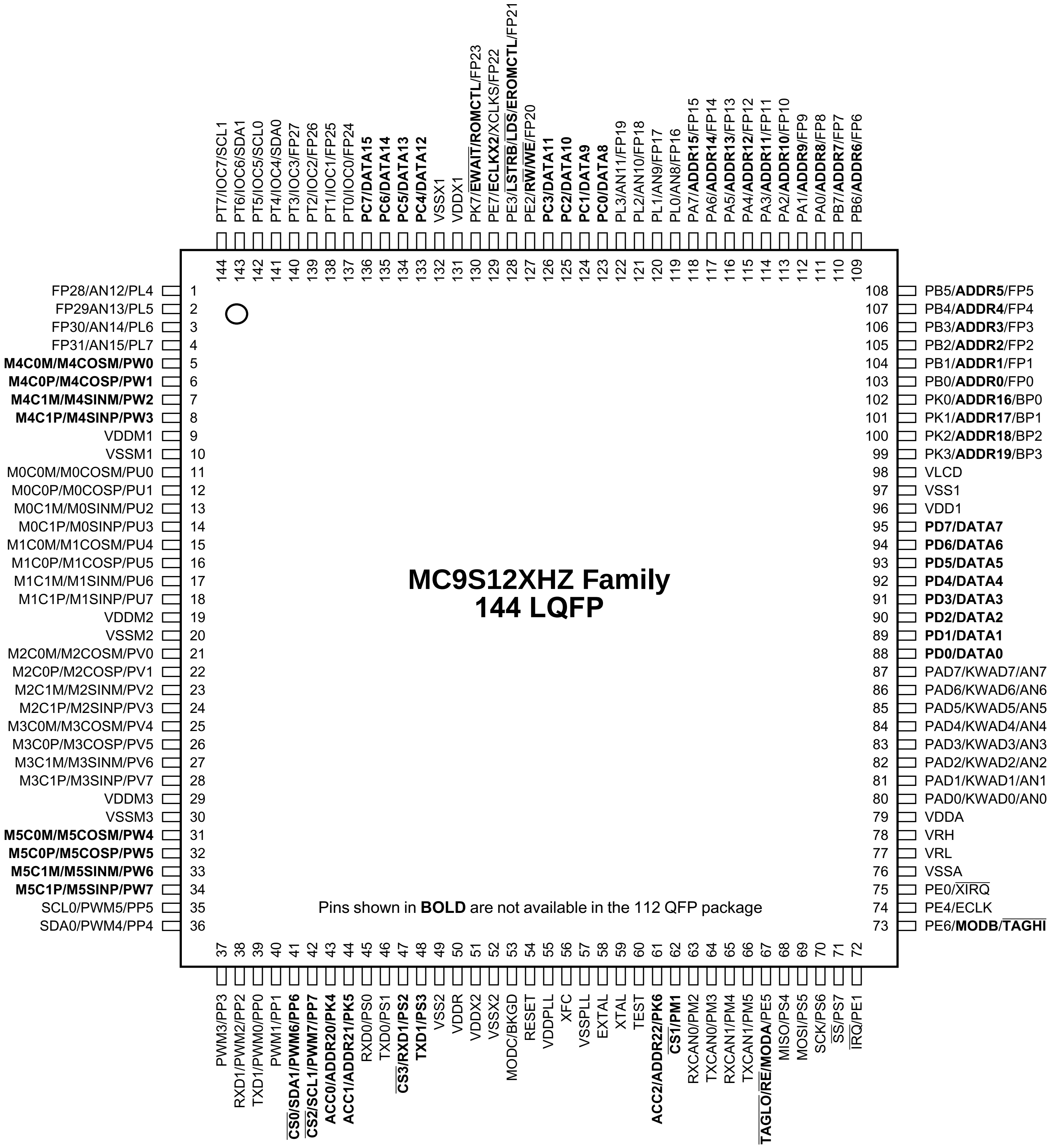


Figure 1-4. MC9S12XHZ Family 144 LQFP Pin Assignment

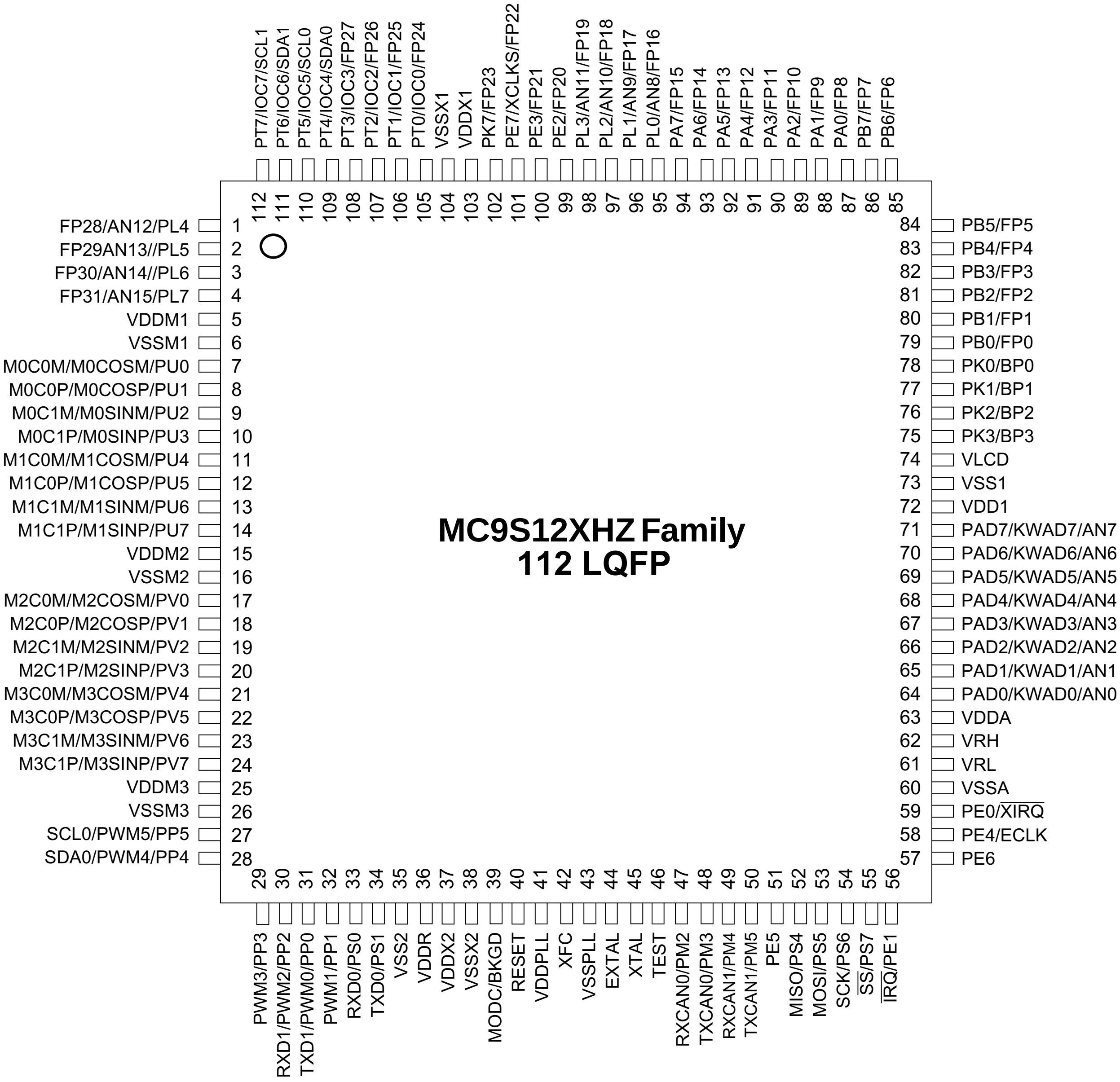


Figure 1-5. MC9S12XHZ Family 112 LQFP Pin Assignment

1.2.2 Signal Properties Summary

Table 1-5 summarizes all pin functions.

Table 1-5. Signal Properties

Pin Name Function 1	Pin Name Function 2	Pin Name Function 3	Pin Name Function 4	Pin Name Function 5	Powered by	Internal Pull Up Resistor		Description
						CTRL	Reset State	
EXTAL	—	—	—	—	V_{DDPLL}	NA	NA	Oscillator pins
XTAL	—	—	—	—	V_{DDPLL}	NA	NA	
$\overline{\mathrm{RESET}}$	—	—	—	—	V_{DDX2}	PULL UP		External reset
TEST	—	—	—	—	NA	NA	NA	Test input - must be tied to VSS in all applications
XFC	—	—	—	—	V_{DDPLL}	NA	NA	PLL loop Filter
BKGD	MODC	—	—	—	V_{DDX2}	Always on	Up	Background debug, mode input
PAD[7:0]	AN[7:0]	KWAD[7:0]	—	—	V_{DDA}	PERAD/ PPSAD	Disabled	Port AD I/O, Analog inputs (ATD), interrupts
PA[7:0]	FP[15:8]	ADDR[15:8]	IVD[15:8]	—	V_{DDX1}	PUCR	Down	Port A I/O, address bus, internal visibility data
PB[7:1]	FP[7:1]	ADDR[7:1]	IVD[7:1]	—	V_{DDX1}	PUCR	Down	Port B I/O, address bus, internal visibility data
PB0	FP0	ADDR0	IVD0	$\overline{\mathrm{UDS}}$	V_{DDX1}	PUCR	Down	Port B I/O, address bus, internal visibility data, upper data strobe
PC[7:0]	—	DATA[15:8]	—	—	V_{DDX1}	PUCR	Disabled	Port C I/O, data bus
PD[7:0]	—	DATA[7:0]	—	—	V_{DDX1}	PUCR	Disabled	Port D I/O, data bus
PE7	FP22	ECLKX2	XCLKS	—	V_{DDX1}	PUCR	Down	Port E I/O, LCD driver, system clock output, clock select
PE6	$\overline{\mathrm{TAGHI}}$	MODB	—	—	V_{DDX2}	While $\overline{\mathrm{RESET}}$ pin is low: Down		Port E I/O, tag high, mode input
PE5	$\overline{\mathrm{TAGLO}}$	MODA	$\overline{\mathrm{RE}}$	—	V_{DDX2}	While $\overline{\mathrm{RESET}}$ pin is low: Down		Port E I/O, tag low, mode input, read enable
PE4	ECLK	—	—	—	V_{DDX2}	PUCR	Down	Port E I/O, bus clock output
PE3	FP21	$\overline{\mathrm{LSTRB}}$	$\overline{\mathrm{LDS}}$	EROMCTL	V_{DDX1}	PUCR	Down	Port E I/O, LCD driver, low byte strobe, EROMON control
PE2	FP20	$\mathrm{R}/\overline{\mathrm{W}}$	$\overline{\mathrm{WE}}$	—	V_{DDX1}	PUCR	Down	Port E I/O, read/write, write enable
PE1	$\overline{\mathrm{IRQ}}$	—	—	—	V_{DDX2}	PUCR	Up	Port E input, maskable interrupt
PE0	$\overline{\mathrm{XIRQ}}$	—	—	—	V_{DDX2}	PUCR	Up	Port E input, non-maskable interrupt

Table 1-5. Signal Properties

Pin Name Function 1	Pin Name Function 2	Pin Name Function 3	Pin Name Function 4	Pin Name Function 5	Powered by	Internal Pull Up Resistor		Description
						CTRL	Reset State	
PK7	FP23	$\overline{\text{ECS}}$	ROMCTL	ROMCTL	V_{DDX1}	PUCR	Down	Port K I/O, emulation chip select, ROM on enable
PK[6:4]	—	ADDR[22:20]	ACC[2:0]	—	V_{DDX2}			Port K I/O, extended address, access source
PK[3:0]	BP[3:0]	ADDR[19:16]	IQSTAT[3:0]	—	V_{DDX1}			Port K I/O, LCD driver, extended address, pipe status
PL[7:4]	FP[31:28]	AN[15:12]	—	—	V_{DDA}	PERL/ PPSL	Down	Port L I/O, LCD drivers, analog inputs (ATD)
PL[3:0]	FP[19:16]	AN[11:8]	—	—	V_{DDX1}			Port L I/O, LCD drivers, analog inputs (ATD)
PM5	TXCAN1	—	—	—	V_{DDX2}	PERM/ PPSM	Disabled	Port M I/O, TX of CAN1
PM4	RXCAN1	—	—	—	V_{DDX2}			Port M I/O, RX of CAN1
PM3	TXCAN0	—	—	—	V_{DDX2}			Port M I/O, TX of CAN0
PM2	RXCAN0	—	—	—	V_{DDX2}			Port M I/O, RX of CAN0
PM1	—	—	$\overline{\text{CS1}}$	—	V_{DDX2}			Port M I/O, chip select 1
PP7	PWM7	SCL1	$\overline{\text{CS2}}$	—	V_{DDX2}	PERP/ PPSP	Disabled	Port P I/O, PWM channel, SCL of IIC1, chip select 2
PP6	PWM6	SDA1	$\overline{\text{CS0}}$	—	V_{DDX2}			Port P I/O, PWM channel, SDA of IIC1, chip select 0
PP5	PWM5	SCL0	—	—	V_{DDX2}			Port P I/O, PWM channel, SCL of IIC0
PP4	PWM4	SDA0	—	—	V_{DDX2}			Port P I/O, PWM channel, SDA of IIC0
PP3	PWM3	—	—	—	V_{DDX2}			Port P I/O, PWM channel
PP2	PWM2	RXD1	—	—	V_{DDX2}			Port P I/O, PWM channel, RXD of SCI1
PP1	PWM1	—	—	—	V_{DDX2}			Port P I/O, PWM channel
PP0	PWM0	TXD1	—	—	V_{DDX2}			Port P I/O, PWM channel, TXD of SCI1
PS7	$\overline{\text{SS}}$	—	—	—	V_{DDX2}	PERS/ PPSS	Disabled	Port S I/O, $\overline{\text{SS}}$ of SPI
PS6	SCK	—	—	—	V_{DDX2}			Port S I/O, SCK of SPI
PS5	MOSI	—	—	—	V_{DDX2}			Port S I/O, MOSI of SPI
PS4	MISO	—	—	—	V_{DDX2}			Port S I/O, MISO of SPI
PS3	TXD1	—	—	—	V_{DDX2}			Port S I/O, TXD of SCI1
PS2	RXD1	—	$\overline{\text{CS3}}$	—	V_{DDX2}			Port S I/O, RXD of SCI1, chip select 3
PS1	TXD0	—	—	—	V_{DDX2}			Port S I/O, TXD of SCI0
PS0	RXD0	—	—	—	V_{DDX2}			Port S I/O, RXD of SCI0

Table 1-5. Signal Properties

Pin Name Function 1	Pin Name Function 2	Pin Name Function 3	Pin Name Function 4	Pin Name Function 5	Powered by	Internal Pull Up Resistor		Description
						CTRL	Reset State	
PT7	IOC7	SCL1	—	—	V_{DDX1}	PERT/ PPST	Disabled	Port T I/O, Timer channels, SCL of IIC1
PT6	IOC6	SDA1	—	—	V_{DDX1}			Port T I/O, Timer channels, SDA of IIC1
PT5	IOC5	SCL0	—	—	V_{DDX1}			Port T I/O, Timer channels, SCL of IIC0
PT4	IOC4	SDA0	—	—	V_{DDX1}			Port T I/O, Timer channels, SDA of IIC0
PT[3:0]	IOC[3:0]	FP[27:24]	—	—	V_{DDX1}	PERT/ PPST	Down	Port T I/O, Timer channels, LCD driver
PU7	M1C1P	M1SINP	—	—	$V_{DDM1,2,3}$	PERU/ PPSU	Disabled	Port U I/O, motor1 coil nodes of MC or SSD1
PU6	M1C1M	M1SINM	—	—	$V_{DDM1,2,3}$			
PU5	M1C0P	M1COSP	—	—	$V_{DDM1,2,3}$			
PU4	M1C0M	M1COSM	—	—	$V_{DDM1,2,3}$			
PU3	M0C1P	M0SINP	—	—	$V_{DDM1,2,3}$			Port U I/O, motor 0 coil nodes of MC or SSD0
PU2	M0C1M	M0SINM	—	—	$V_{DDM1,2,3}$			
PU1	M0C0P	M0COSP	—	—	$V_{DDM1,2,3}$			
PU0	M0C0M	M0COSM	—	—	$V_{DDM1,2,3}$			
PV7	M3C1P	M3SINP	—	—	$V_{DDM1,2,3}$	PERV/ PPSV	Disabled	Port V I/O, motor 3 coil nodes of MC or SSD3
PV6	M3C1M	M3SINM	—	—	$V_{DDM1,2,3}$			
PV5	M3C0P	M3COSP	—	—	$V_{DDM1,2,3}$			
PV4	M3C0M	M3COSM	—	—	$V_{DDM1,2,3}$			
PV3	M2C1P	M2SINP	—	—	$V_{DDM1,2,3}$			Port V I/O, motor 2 coil nodes of MC or SSD2
PV2	M2C1M	M2SINM	—	—	$V_{DDM1,2,3}$			
PV1	M2C0P	M2COSP	—	—	$V_{DDM1,2,3}$			
PV0	M2C0M	M2COSM	—	—	$V_{DDM1,2,3}$			
PW7	M5C1P	M5SINP	—	—	$V_{DDM1,2,3}$	PERW/ PPSW	Disabled	Port W I/O, motor 5 coil nodes of MC or SSD5
PW6	M5C1M	M5SINM	—	—	$V_{DDM1,2,3}$			
PW5	M5C0P	M5COSP	—	—	$V_{DDM1,2,3}$			
PW4	M5C0M	M5COSM	—	—	$V_{DDM1,2,3}$			
PW3	M4C1P	M4SINP	—	—	$V_{DDM1,2,3}$			Port W I/O, motor 4 coil nodes of MC or SSD4
PW2	M4C1M	M4SINM	—	—	$V_{DDM1,2,3}$			
PW1	M4C0P	M4COSP	—	—	$V_{DDM1,2,3}$			
PW0	M4C0M	M4COSM	—	—	$V_{DDM1,2,3}$			

Table 1-6. Power and Ground

Mnemonic	Nominal Voltage	Description
V_{LCD}	5.0 V	Voltage reference pin for the LCD driver.
V_{DD1}	2.5 V	Internal power and ground generated by internal regulator. These also allow an external source to supply the core V_{DD}/V_{SS} voltages and bypass the internal voltage regulator.
V_{SS1} V_{SS2}	0V	
V_{DDR}	5.0 V	External power and ground, supply to pin drivers and internal voltage regulator.
V_{SSR}	0 V	
V_{DDX1} V_{DDX2}	5.0 V	External power and ground, supply to pin drivers.
V_{SSX1} V_{SSX2}	0 V	
V_{DDA}	5.0 V	Operating voltage and ground for the analog-to-digital converter and the reference for the internal voltage regulator, allows the supply voltage to the A/D to be bypassed independently.
V_{SSA}	0 V	
V_{RH}	5.0 V	Reference voltage high for the ATD converter.
V_{RL}	0 V	Reference voltage low for the ATD converter.
V_{DDPLL}	2.5 V	Provides operating voltage and ground for the phased-locked Loop. This allows the supply voltage to the PLL to be bypassed independently. Internal power and ground generated by internal regulator.
V_{SSPLL}	0 V	
$V_{DDM1,2,3}$	5.0 V	Provides operating voltage and ground for motor 0, 1, 2 and 3.
$V_{SSM1,2,3}$	0 V	

NOTE

All V_{SS} pins must be connected together in the application. Because fast signal transitions place high, short-duration current demands on the power supply, use bypass capacitors with high-frequency characteristics and place them as close to the MCU as possible. Bypass requirements depend on MCU pin load.

1.2.3 Detailed Signal Descriptions

1.2.3.1 EXTAL, XTAL — Oscillator Pins

EXTAL and XTAL are the crystal driver and external clock pins. On reset all the device clocks are derived from the EXTAL input frequency. XTAL is the crystal output.

1.2.3.2 $\overline{\text{RESET}}$ — External Reset Pin

The $\overline{\text{RESET}}$ pin is an active low bidirectional control signal. It acts as an input to initialize the MCU to a known start-up state, and an output when an internal MCU function causes a reset.The $\overline{\text{RESET}}$ pin has an internal pullup device.

1.2.3.3 TEST — Test Pin

This input only pin is reserved for test. This pin has a pulldown device.

NOTE

The TEST pin must be tied to V_{SS} in all applications.

1.2.3.4 XFC — PLL Loop Filter Pin

Please ask your Freescale representative for the interactive application note to compute PLL loop filter elements. Any current leakage on this pin must be avoided.

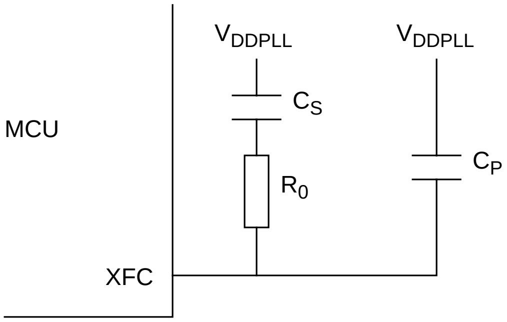


Figure 1-6. PLL Loop Filter Connections

1.2.3.5 BKGD / MODC — Background Debug and Mode Pin

The BKGD/MODC pin is used as a pseudo-open-drain pin for the background debug communication. It is used as a MCU operating mode select pin during reset. The state of this pin is latched to the MODC bit at the rising edge of $\overline{\text{RESET}}$. The BKGD pin has a pullup device.

1.2.3.6 PAD[7:0] / AN[7:0] / KWAD[7:0] — Port AD I/O Pins [7:0]

PAD7–PAD0 are general-purpose input or output pins and analog inputs for the analog-to-digital converter. They can be configured to generate an interrupt causing the MCU to exit STOP or WAIT mode.

1.2.3.7 PA[7:0] / ADDR[15:8] / IVD[15:8] — Port A I/O Pins

PA[7:0] are general-purpose input or output pins. In MCU expanded modes of operation, these pins are used for the external address bus. In MCU emulation modes of operation, these pins are used for external address bus and internal visibility read data.

1.2.3.8 PB[7:1] / ADDR[7:1] / IVD[7:1] — Port B I/O Pins

PB[7:1] are general-purpose input or output pins. In MCU expanded modes of operation, these pins are used for the external address bus. In MCU emulation modes of operation, these pins are used for external address bus and internal visibility read data.

1.2.3.9 PB0 / ADDR0 / $\overline{\text{UDS}}$ / IVD[0] — Port B I/O Pin 0

PB0 is a general-purpose input or output pin. In MCU expanded modes of operation, this pin is used for the external address bus ADDR0 or as upper data strobe signal. In MCU emulation modes of operation, this pin is used for external address bus ADDR0 and internal visibility read data IVD0.

1.2.3.10 PC[7:0] / DATA [15:8] — Port C I/O Pins

PC[7:0] are general-purpose input or output pins. In MCU expanded modes of operation, these pins are used for the external data bus.

The input voltage thresholds for PC[7:0] can be configured to reduced levels, to allow data from an external 3.3-V peripheral to be read by the MCU operating at 5.0 V. The input voltage thresholds for PC[7:0] are configured to reduced levels out of reset in expanded and emulation modes. The input voltage thresholds for PC[7:0] are configured to 5-V levels out of reset in normal modes.

1.2.3.11 PD[7:0] / DATA [7:0] — Port D I/O Pins

PD[7:0] are general-purpose input or output pins. In MCU expanded modes of operation, these pins are used for the external data bus.

The input voltage thresholds for PD[7:0] can be configured to reduced levels, to allow data from an external 3.3-V peripheral to be read by the MCU operating at 5.0 V. The input voltage thresholds for PD[7:0] are configured to reduced levels out of reset in expanded and emulation modes. The input voltage thresholds for PC[7:0] are configured to 5-V levels out of reset in normal modes.

1.2.3.12 PE7 / FP22 / ECLKX2 / XCLKS — Port E I/O Pin 7

PE7 is a general-purpose input or output pin. The pin can be configured as frontplane segment driver output FP22 of the LCD module or as the internal system clock ECLKX2.

The XCLKS is an input signal which controls whether a crystal in combination with the internal loop controlled (low power) Pierce oscillator is used or whether full swing Pierce oscillator/external clock circuitry is used.

The XCLKS signal selects the oscillator configuration during reset low phase while a clock quality check is ongoing. This is the case for:

- Power on reset or low-voltage reset
- Clock monitor reset
- Any reset while in self-clock mode or full stop mode

The selected oscillator configuration is frozen with the rising edge of reset.

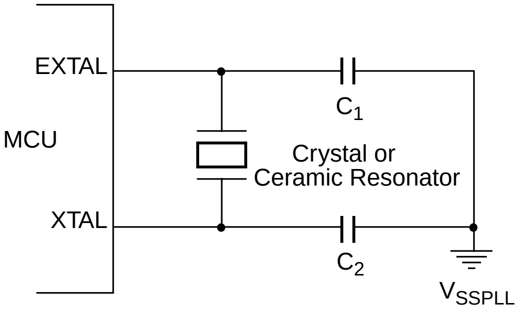


Figure 1-7. Loop Controlled Pierce Oscillator Connections (PE7 = 0)

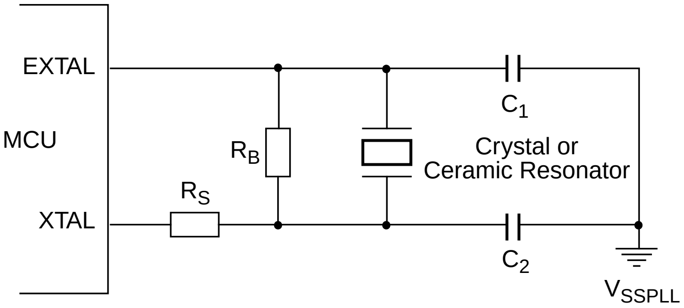


Figure 1-8. Full Swing Pierce Oscillator Connections (PE7 = 1)

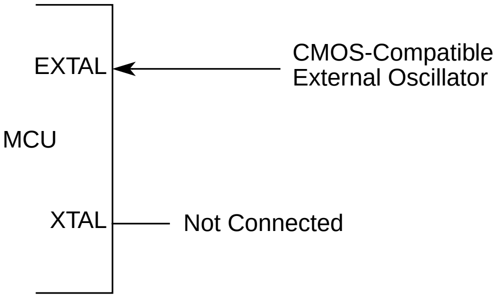


Figure 1-9. External Clock Connections (PE7 = 1)

1.2.3.13 PE6 / MODB / $\overline{\text{TAGHI}}$ — Port E I/O Pin 6

PE6 is a general-purpose input or output pin. It is used as a MCU operating mode select pin during reset. The state of this pin is latched to the MODB bit at the rising edge of $\overline{\text{RESET}}$. This pin is an input with a pull-down device which is only active when $\overline{\text{RESET}}$ is low. $\overline{\text{TAGHI}}$ is used to tag the high half of the instruction word being read into the instruction queue.

The input voltage threshold for PE6 can be configured to reduced levels, to allow data from an external 3.3-V peripheral to be read by the MCU operating at 5.0 V. The input voltage threshold for PE6 is configured to reduced levels out of reset in expanded and emulation modes.

1.2.3.14 PE5 / MODA / $\overline{\text{TAGLO}}$ / $\overline{\text{RE}}$ — Port E I/O Pin 5

PE5 is a general-purpose input or output pin. It is used as a MCU operating mode select pin during reset. The state of this pin is latched to the MODA bit at the rising edge of $\overline{\text{RESET}}$. This pin is shared with the read enable $\overline{\text{RE}}$ output. This pin is an input with a pull-down device which is only active when $\overline{\text{RESET}}$ is low. $\overline{\text{TAGLO}}$ is used to tag the low half of the instruction word being read into the instruction queue.

The input voltage threshold for PE5 can be configured to reduced levels, to allow data from an external 3.3-V peripheral to be read by the MCU operating at 5.0 V. The input voltage threshold for PE5 is configured to reduced levels out of reset in expanded and emulation modes.

1.2.3.15 PE4 / ECLK — Port E I/O Pin 4

PE4 is a general-purpose input or output pin. It can be configured to drive the internal bus clock ECLK. ECLK can be used as a timing reference.

1.2.3.16 PE3 / FP21 / $\overline{\text{LSTRB}}$ / $\overline{\text{LDS}}$ / EROMCTL— Port E I/O Pin 3

PE3 is a general-purpose input or output pin. It can be configured as frontplane segment driver output FP21 of the LCD module. In MCU expanded modes of operation, $\overline{\text{LSTRB}}$ or $\overline{\text{LDS}}$ can be used for the low byte strobe function to indicate the type of bus access. At the rising edge of $\overline{\text{RESET}}$ the state of this pin is latched to the EROMON bit.

1.2.3.17 PE2 / FP20 / R/$\overline{\text{W}}$ / $\overline{\text{WE}}$—Port E I/O Pin 2

PE2 is a general-purpose input or output pin. It can be configured as frontplane segment driver output FP20 of the LCD module. In MCU expanded modes of operations, this pin drives the read/write output signal or write enable output signal for the external bus. It indicates the direction of data on the external bus.

1.2.3.18 PE1 / $\overline{\text{IRQ}}$ — Port E Input Pin 1

PE1 is a general-purpose input pin and the maskable interrupt request input that provides a means of applying asynchronous interrupt requests. This will wake up the MCU from stop or wait mode.

1.2.3.19 PE0 / $\overline{\text{XIRQ}}$ — Port E Input Pin 0

PE0 is a general-purpose input pin and the non-maskable interrupt request input that provides a means of applying asynchronous interrupt requests. This will wake up the MCU from stop or wait mode.

1.2.3.20 PK7 / FP23 / $\overline{\text{EWAIT}}$ / ROMCTL — Port K I/O Pin 7

PK7 is a general-purpose input or output pin. It can be configured as frontplane segment driver output FP23 of the LCD module. During MCU emulation modes and normal expanded modes of operation, this pin is used to enable the Flash EEPROM memory in the memory map (ROMCTL). At the rising edge of $\overline{\text{RESET}}$, the state of this pin is latched to the ROMON bit. The $\overline{\text{EWAIT}}$ input signal maintains the external bus access until the external device is ready to capture data (write) or provide data (read).

The input voltage threshold for PK7 can be configured to reduced levels, to allow data from an external 3.3-V peripheral to be read by the MCU operating at 5.0 V. The input voltage threshold for PK7 is configured to reduced levels out of reset in expanded and emulation modes.

1.2.3.21 PK[6:4] / ADDR[22:20] / ACC[2:0] — Port K I/O Pin [6:4]

PK[6:4] are general-purpose input or output pins. During MCU expanded modes of operation, the ACC[2:0] signals are used to indicate the access source of the bus cycle. This pins also provide the expanded addresses ADDR[22:20] for the external bus. In Emulation modes ACC[2:0] is available and is time multiplexed with the high addresses

1.2.3.22 PK[3:0] / BP[3:0] / ADDR[19:16] / IQSTAT[3:0] — Port K I/O Pins [3:0]

PK3-PK0 are general-purpose input or output pins. The pins can be configured as backplane segment driver outputs BP3–BP0 of the LCD module. In MCU expanded modes of operation, these pins provide the expanded address ADDR[19:16] for the external bus and carry instruction pipe information.

1.2.3.23 PL[7:4] / FP[31:28] / AN[15:12] — Port L I/O Pins [7:4]

PL7–PL4 are general-purpose input or output pins. They can be configured as frontplane segment driver outputs FP31–FP28 of the LCD module or analog inputs for the analog-to-digital converter.

1.2.3.24 PL[3:0] / FP[19:16] / AN[11:8] — Port L I/O Pins [3:0]

PL3–PL0 are general-purpose input or output pins. They can be configured as frontplane segment driver outputs FP19–FP16 of the LCD module or analog inputs for the analog-to-digital converter.

1.2.3.25 PM5 / TXCAN1 — Port M I/O Pin 5

PM5 is a general-purpose input or output pin. It can be configured as the transmit pin TXCAN1 of the scalable controller area network controller 1 (CAN1)

1.2.3.26 PM4 / RXCAN1 — Port M I/O Pin 4

PM4 is a general-purpose input or output pin. It can be configured as the receive pin RXCAN1 of the scalable controller area network controller 1 (CAN1)

1.2.3.27 PM3 / TXCAN0 — Port M I/O Pin 3

PM3 is a general-purpose input or output pin. It can be configured as the transmit pin TXCAN0 of the scalable controller area network controller 0 (CAN0)

1.2.3.28 PM2 / RXCAN0 — Port M I/O Pin 2

PM2 is a general-purpose input or output pin. It can be configured as the receive pin RXCAN0 of the scalable controller area network controller 0 (CAN0).

1.2.3.29 PM1 / $\overline{\text{CS1}}$ — Port M I/O Pin 1

PM1 is a general-purpose input or output pin. It can be configured to provide a chip-select output.

1.2.3.30 PP7 / PWM7 / SCL1 / $\overline{\text{CS2}}$ — Port P I/O Pin 7

PP7 is a general-purpose input or output pin. It can be configured as pulse width modulator (PWM) channel output PWM7 or the serial clock pin SCL1 of the inter-IC bus interface 1 (IIC1). It can be configured to provide a chip-select output.

1.2.3.31 PP6 / PWM6 / SDA1 / $\overline{\text{CS0}}$ — Port P I/O Pin 6

PP6 is a general-purpose input or output pin. It can be configured as pulse width modulator (PWM) channel output PWM6 or the serial data pin SDA1 of the inter-IC bus interface 1 (IIC1). It can be configured to provide a chip-select output.

1.2.3.32 PP5 / PWM5 / SCL0 — Port P I/O Pin 5

PP5 is a general-purpose input or output pin. It can be configured as pulse width modulator (PWM) channel output PWM5 or the serial clock pin SCL0 of the inter-IC bus interface 0 (IIC0).

1.2.3.33 PP4 / PWM4 / SDA0 — Port P I/O Pin 4

PP4 is a general-purpose input or output pin. It can be configured as pulse width modulator (PWM) channel output PWM4 or the serial data pin SDA0 of the inter-IC bus interface 0 (IIC0).

1.2.3.34 PP3 / PWM3 — Port P I/O Pin 3

PP3 is a general-purpose input or output pin. It can be configured as pulse width modulator (PWM) channel output PWM3.

1.2.3.35 PP2 / PWM2 / RXD1 — Port P I/O Pin 2

PP2 is a general-purpose input or output pin. It can be configured as pulse width modulator (PWM) channel output PWM2 or the receive pin RXD1 of the serial communication interface 1 (SCI1).

1.2.3.36 PP1 / PWM1 — Port P I/O Pin 1

PP1 is a general-purpose input or output pin. It can be configured as pulse width modulator (PWM) channel output PWM1.

1.2.3.37 PP0 / PWM0 / TXD1 — Port P I/O Pin 0

PP0 is a general-purpose input or output pin. It can be configured as pulse width modulator (PWM) channel output PWM0 or the transmit pin TXD1 of the serial communication interface 1 (SCI1).

1.2.3.38 PS7 / $\overline{\text{SS}}$ — Port S I/O Pin 7

PS7 is a general-purpose input or output pin. It can be configured as slave select pin $\overline{\text{SS}}$ of the serial peripheral interface (SPI).

1.2.3.39 PS6 / SCK — Port S I/O Pin 6

PS6 is a general-purpose input or output pin. It can be configured as serial clock pin SCK of the serial peripheral interface (SPI).

1.2.3.40 PS5 / MOSI — Port S I/O Pin 5

PS5 is a general-purpose input or output pin. It can be configured as the master output (during master mode) or slave input (during slave mode) pin MOSI of the serial peripheral interface (SPI).

1.2.3.41 PS4 / MISO — Port S I/O Pin 4

PS4 is a general-purpose input or output pin. It can be configured as master input (during master mode) or slave output (during slave mode) pin MISO for the serial peripheral interface (SPI).

1.2.3.42 PS3 / TXD1 — Port S I/O Pin 3

PS3 is a general-purpose input or output pin. It can be configured as transmit pin TXD1 of the serial communication interface 1 (SCI1).

1.2.3.43 PS2 / RXD1 / $\overline{\text{CS2}}$ — Port S I/O Pin 2

PS2 is a general-purpose input or output pin. It can be configured as receive pin RXD1 of the serial communication interface 1 (SCI1). It can be configured to provide a chip-select output.

1.2.3.44 PS1 / TXD0 — Port S I/O Pin 1

PS1 is a general-purpose input or output pin. It can be configured as transmit pin TXD0 of the serial communication interface 0 (SCI0).

1.2.3.45 PS0 / RXD0 — Port S I/O Pin 0

PS0 is a general-purpose input or output pin. It can be configured as receive pin RXD0 of the serial communication interface 0 (SCI0).

1.2.3.46 PT7 / IOC7 / SCL1 — Port T I/O Pin 7

PT7 is a general-purpose input or output pin. It can be configured as input capture or output compare pin IOC7 of the enhanced capture timer (ECT) or the serial clock pin SCL1 of the inter-IC bus interface 1 (IIC1).

1.2.3.47 PT6 / IOC6 / SDA1 — Port T I/O Pin 6

PT6 is a general-purpose input or output pin. It can be configured as input capture or output compare pin IOC6 of the enhanced capture timer (ECT) or the serial data pin SDA1 of the inter-IC bus interface 1 (IIC1).

1.2.3.48 PT5 / IOC5 / SCL0 — Port T I/O Pin 5

PT5 is a general-purpose input or output pin. It can be configured as input capture or output compare pin IOC5 of the enhanced capture timer (ECT) or the serial clock pin SCL0 of the inter-IC bus interface 0 (IIC0).

1.2.3.49 PT4 / IOC4 / SDA0 — Port T I/O Pin 4

PT4 is a general-purpose input or output pin. It can be configured as input capture or output compare pin IOC4 of the enhanced capture timer (ECT) or the serial data pin SDA0 of the inter-IC bus interface 0 (IIC0).

1.2.3.50 PT[3:0] / IOC[3:0] / FP[27:24] — Port T I/O Pins [3:0]

PT3–PT0 are general-purpose input or output pins. They can be configured as input capture or output compare pins IOC3–IOC0 of the enhanced capture timer (ECT). They can be configured as frontplane segment driver outputs FP27–FP24 of the LCD module.

1.2.3.51 PU[7:4] / M1C1(SIN)P, M1C1(SIN)M, M1C0(COS)P, M1C0(COS)M — Port U I/O Pins [7:4]

PU7–PU4 are general-purpose input or output pins. They can be configured as high current PWM output pins which can be used for motor drive or to measure the back EMF to calibrate the pointer reset position. These pins interface to the coils of motor 1.

1.2.3.52 PU[3:0] / M0C1(SIN)P, M0C1(SIN)M, M0C0(COS)P, M0C0(COS)M — Port U I/O Pins [3:0]

PU3–PU0 are general-purpose input or output pins. They can be configured as high current PWM output pins which can be used for motor drive or to measure the back EMF to calibrate the pointer reset position. These pins interface to the coils of motor 0.

1.2.3.53 PV[7:4] / M3C1(SIN)P, M3C1(SIN)M, M3C0(COS)P, M3C0(COS)M — Port V I/O Pins [7:4]

PV7–PV4 are general-purpose input or output pins. They can be configured as high current PWM output pins which can be used for motor drive or to measure the back EMF to calibrate the pointer reset position. These pins interface to the coils of motor 3.

1.2.3.54 PV[3:0] / M2C1(SIN)P, M2C1(SIN)M, M2C0(COS)P, M2C0(COS)M — Port V I/O Pins [3:0]

PV3–PV0 are general-purpose input or output pins. They can be configured as high current PWM output pins which can be used for motor drive or to measure the back EMF to calibrate the pointer reset position. These pins interface to the coils of motor 2.

1.2.3.55 PW[7:4] / M5C1(SIN)P, M5C1(SIN)M, M5C0(COS)P, M5C0(COS)M — Port W I/O Pins [7:4]

PW7–PW4 are general-purpose input or output pins. They can be configured as high current PWM output pins which can be used for motor drive or to measure the back EMF to calibrate the pointer reset position. These pins interface to the coils of motor 5.

1.2.3.56 PW[3:0] / M4C1(SIN)P, M4C1(SIN)M, M4C0(COS)P, M4C0(COS)M — Port W I/O Pins [3:0]

PW3–PW0 are general-purpose input or output pins. They can be configured as high current PWM output pins which can be used for motor drive or to measure the back EMF to calibrate the pointer reset position. These pins interface to the coils of motor 4.

1.2.4 Power Supply Pins

Power and ground pins are described below.

NOTE

All V_{SS} pins must be connected together in the application.

1.2.4.1 V_{DDR} — External Power Pin

V_{DDR} is the power supply pin for the internal voltage regulator.

1.2.4.2 V_{DDX1}, V_{DDX2}, V_{SSX1}, V_{SSX2} — External Power and Ground Pins

External power and ground for I/O drivers. Because fast signal transitions place high, short-duration current demands on the power supply, use bypass capacitors with high-frequency characteristics and place them as close to the MCU as possible. Bypass requirements depend on how heavily the MCU pins are loaded.

V_{DDX1} and V_{DDX2} as well as V_{SSX1} and V_{SSX2} are not internally connected.

1.2.4.3 V_{DD1}, V_{SS1}, V_{SS2} — Internal Logic Power Pins

Power is supplied to the MCU through V_{DD} and V_{SS}. Because fast signal transitions place high, short-duration current demands on the power supply, use bypass capacitors with high-frequency characteristics and place them as close to the MCU as possible. This 2.5-V supply is derived from the internal voltage regulator. There is no static load on those pins allowed.

V_{SS1} and V_{SS2} are internally connected.

1.2.4.4 V_{DDA}, V_{SSA} — Power Supply Pins for ATD and VREG

V_{DDA}, V_{SSA} are the power supply and ground pins for the voltage regulator and the analog-to-digital converter.

1.2.4.5 V_{RH}, V_{RL} — ATD Reference Voltage Input Pins

V_{RH} and V_{RL} are the voltage reference pins for the analog-to-digital converter.

1.2.4.6 V_{DDPLL}, V_{SSPLL} — Power Supply Pins for PLL

Provides operating voltage and ground for the oscillator and the phased-locked loop. This allows the supply voltage to the oscillator and PLL to be bypassed independently. This 2.5-V voltage is generated by the internal voltage regulator.

1.2.4.7 V_{DDM1}, V_{DDM2}, V_{DDM3} — Power Supply Pins for Motor 0 to 3

V_{DDM1}, V_{DDM2} and V_{DDM3} are the supply pins for the ports U, V and W. V_{DDM1}, V_{DDM2} and V_{DDM3} are internally connected.

1.2.4.8 V_{SSM1}, V_{SSM2}, V_{SSM3} — Ground Pins for Motor 0 to 3

V_{SSM1}, V_{SSM2} and V_{SSM3} are the ground pins for the ports U, V and W. V_{SSM1}, V_{SSM2} and V_{SSM3} are internally connected.

1.2.4.9 V_{LCD} — Power Supply Reference Pin for LCD driver

V_{LCD} is the voltage reference pin for the LCD driver. Adjusting the voltage on this pin will change the display contrast.

1.3 System Clock Description

The clock and reset generator module (CRG) provides the internal clock signals for the core and all peripheral modules. Figure 1-10 shows the clock connections from the CRG to all modules.

Consult the CRG block description chapter for details on clock generation.

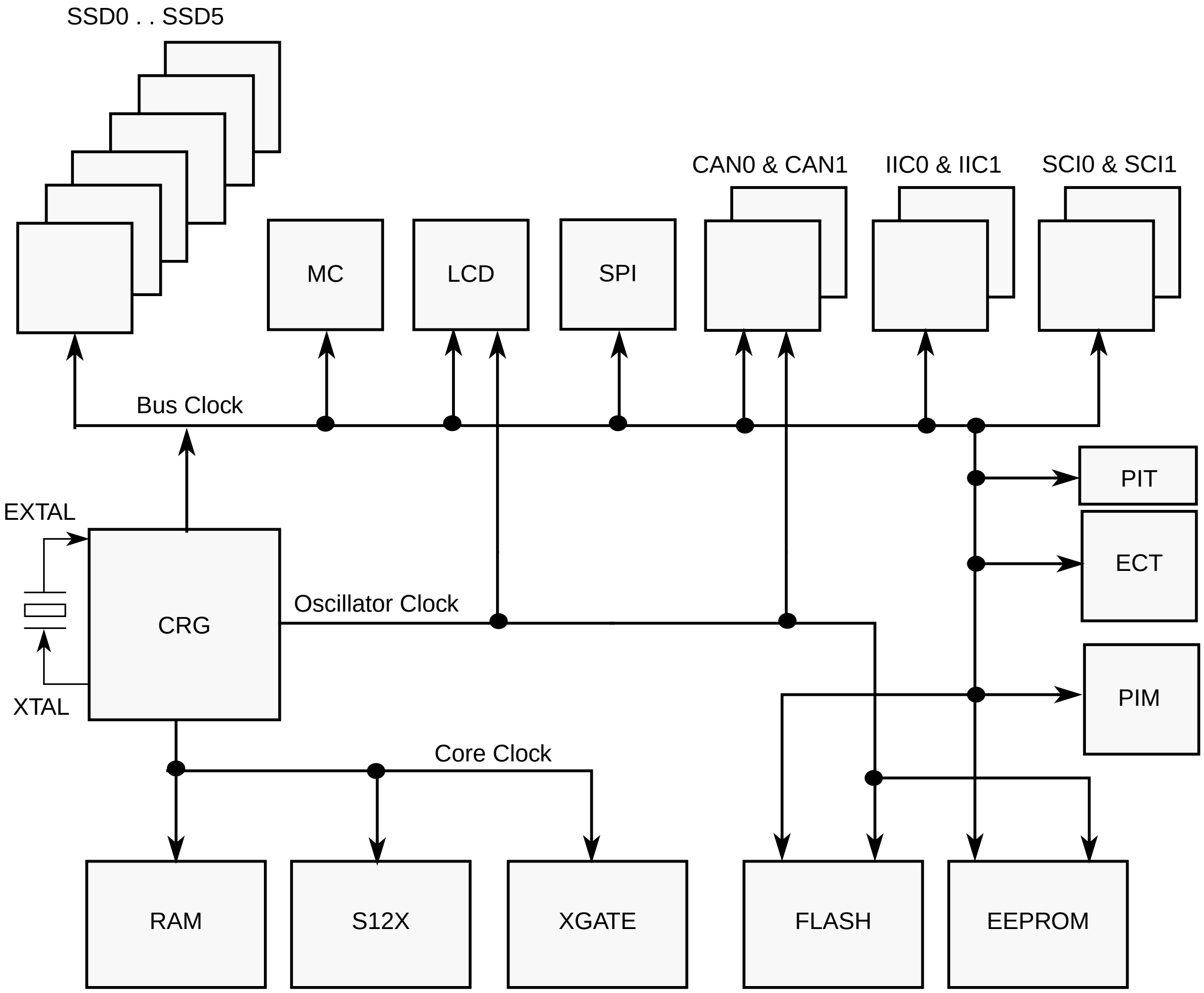


Figure 1-10. Clock Connections

The MCU's system clock can be supplied in several ways enabling a range of system operating frequencies to be supported:

- The on-chip phase locked loop (PLL)
- the PLL self clocking
- the oscillator

The clock generated by the PLL or oscillator provides the main system clock frequencies core clock and bus clock. As shown in Figure 1-10, this system clocks are used throughout the MCU to drive the core, the memories, and the peripherals.

The program Flash memory and the EEPROM are supplied by the bus clock and the oscillator clock.The oscillator clock is used as a time base to derive the program and erase times for the NVM's. Consult the FTX512k4 and EETX4K block description chapters for more details on the operation of the NVM's.

The CAN modules may be configured to have their clock sources derived either from the bus clock or directly from the oscillator clock. This allows the user to select its clock based on the required jitter performance. Consult MSCAN block description for more details on the operation and configuration of the CAN blocks.

In order to ensure the presence of the clock the MCU includes an on-chip clock monitor connected to the output of the oscillator. The clock monitor can be configured to invoke the PLL self-clocking mode or to generate a system reset if it is allowed to time out as a result of no oscillator clock being present.

In addition to the clock monitor, the MCU also provides a clock quality checker which performs a more accurate check of the clock. The clock quality checker counts a predetermined number of clock edges within a defined time window to insure that the clock is running. The checker can be invoked following specific events such as on wake-up or clock monitor failure.

1.4 Chip Configuration Summary

The MCU can operate in six different modes. The different modes, the state of ROMCTL and EROMCTL signal on rising edge of $\overline{\text{RESET}}$, and the security state of the MCU affects the following device characteristics:

- External bus interface configuration
- Flash in memory map, or not
- Debug features enabled or disabled

The operating mode out of reset is determined by the states of the MODC, MODB, and MODA signals during reset (see Table 1-7). The MODC, MODB, and MODA bits in the MODE register show the current operating mode and provide limited mode switching during operation. The states of the MODC, MODB, and MODA signals are latched into these bits on the rising edge of $\overline{\text{RESET}}$.

In normal expanded mode and in emulation modes the ROMON bit and the EROMON bit in the MMCCTL1 register defines if the on chip flash memory is the memory map, or not. (See Table 1-7.) For a detailed description of the ROMON and EROMON bits refer to the S12X_MMC Bblock description chapter.

The state of the ROMCTL signal is latched into the ROMON bit in the MMCCTL1 register on the rising edge of $\overline{\text{RESET}}$. The state of the EROMCTL signal is latched into the EROMON bit in the MISC register on the rising edge of $\overline{\text{RESET}}$.

Table 1-7. Chip Modes and Data Sources

Chip Modes	BKGD = MODC	PE6 = MODB	PE5 = MODA	PK7 = ROMCTL	PE3 = EROMCTL	Data Source[1]
Normal single chip	1	0	0	X	X	Internal
Special single chip	0	0	0			
Emulation single chip	0	0	1	X	0	Emulation memory
				X	1	Internal Flash
Normal expanded	1	0	1	0	X	External application
				1	X	Internal Flash
Emulation expanded	0	1	1	0	X	External application
				1	0	Emulation memory
				1	1	Internal Flash
Special test	0	1	0	0	X	External application
				1	X	Internal Flash

[1] Internal means resources inside the MCU are read/written.
Internal Flash means Flash resources inside the MCU are read/written.
Emulation memory means resources inside the emulator are read/written (PRU registers, Flash replacement, RAM, EEPROM, and register space are always considered internal).
External application means resources residing outside the MCU are read/written.

The configuration of the oscillator can be selected using the XCLKS signal (see Table 1-8). For a detailed description please refer to the CRG block description chapter.

Table 1-8. Clock Selection Based on PE7

PE7 = XCLKS	Description
0	Loop controlled Pierce oscillator selected
1	Full swing Pierce oscillator or external clock source selected

1.5 Modes of Operation

1.5.1 User Modes

1.5.1.1 Normal Expanded Mode

Ports K, A, and B are configured as a 23-bit address bus, ports C and D are configured as a 16-bit data bus, and port E provides bus control and status signals. This mode allows 16-bit external memory and peripheral devices to be interfaced to the system. The fastest external bus rate is divide by 2 from the internal bus rate.

1.5.1.2 Normal Single-Chip Mode

There is no external bus in this mode. The processor program is executed from internal memory. Ports A, B,C,D, K, and most pins of port E are available as general-purpose I/O.

1.5.1.3 Special Single-Chip Mode

This mode is used for debugging single-chip operation, boot-strapping, or security related operations. The background debug module BDM is active in this mode. The CPU executes a monitor program located in an on-chip ROM. BDM firmware is waiting for additional serial commands through the BKGD pin. There is no external bus after reset in this mode.

1.5.1.4 Emulation of Expanded Mode

Developers use this mode for emulation systems in which the users target application is normal expanded mode. Code is executed from external memory or from internal memory depending on the state of ROMON and EROMON bit. In this mode the internal operation is visible on external bus interface.

1.5.1.5 Emulation of Single-Chip Mode

Developers use this mode for emulation systems in which the user’s target application is normal single-chip mode. Code is executed from external memory or from internal memory depending on the state of ROMON and EROMON bit. In this mode the internal operation is visible on external bus interface.

1.5.1.6 Special Test Mode

Freescale internal use only.

1.5.2 Low-Power Modes

The microcontroller features two main low-power modes. Consult the respective block description chapter for information on the module behavior in system stop, system pseudo stop, and system wait mode. An important source of information about the clock system is the Clock and Reset Generator (CRG) block description chapter.

1.5.2.1 System Stop Modes

The system stop modes are entered if the CPU executes the STOP instruction and the XGATE doesn’t execute a thread and the XGFACT bit in the XGMCTL register is cleared. Depending on the state of the PSTP bit in the CLKSEL register the MCU goes into pseudo stop mode or full stop mode. Please refer to CRG block description chapter. Asserting $\overline{\text{RESET}}$, $\overline{\text{XIRQ}}$, $\overline{\text{IRQ}}$ or any other interrupt ends the system stop modes.

1.5.2.2 Pseudo Stop Mode

In this mode the clocks are stopped but the oscillator is still running and the real time interrupt (RTI) or watchdog (COP) submodule can stay active. Other peripherals are turned off. This mode consumes more current than the system stop mode, but the wake up time from this mode is significantly shorter.

1.5.2.3 Full Stop Mode

The oscillator is stopped in this mode. All clocks are switched off. All counters and dividers remain frozen.

1.5.2.4 System Wait Mode

This mode is entered when the CPU executes the WAI instruction. In this mode the CPU will not execute instructions. The internal CPU clock is switched off. All peripherals and the XGATE can be active in system wait mode. For further power consumption savings, the peripherals can individually turn off their local clocks. Asserting $\overline{\text{RESET}}$, $\overline{\text{XIRQ}}$, $\overline{\text{IRQ}}$ or any other interrupt that has not been masked ends system wait mode.

1.5.3 Freeze Mode

The enhanced capture timer, pulse width modulator, analog-to-digital converter, the periodic interrupt timer and the XGATE module provide a software programmable option to freeze the module status during the background debug module is active. This is useful when debugging application software. For detailed description of the behavior of the ATD, ECT, PWM, XGATE and PIT when the background debug module is active consult the corresponding module block description chapters.

1.6 Resets and Interrupts

Consult the S12XCPU block description chapter for information on exception processing.

1.6.1 Vectors

Table 1-9 lists all interrupt sources and vectors in the default order of priority. The interrupt module (S12XINT) provides an interrupt vector base register (IVBR) to relocate the vectors. Associated with each I-bit maskable service request is a configuration register. It selects if the service request is enabled, the service request priority level and whether the service request is handled either by the S12X CPU or by the XGATE module.

Table 1-9. Interrupt Vector Locations (Sheet 1 of 3)

Vector Address[1]	XGATE Channel ID[2]	Interrupt Source	CCR Mask	Local Enable
0xFFFE	—	System reset or illegal access reset	None	None
0xFFFC	—	Clock monitor reset	None	PLLCTL (CME, SCME)
0xFFFA	—	COP watchdog reset	None	COP rate select
Vector base + 0xF8	—	Unimplemented instruction trap	None	None
Vector base+ 0xF6	—	SWI	None	None
Vector base+ 0xF4	—	$\overline{\text{XIRQ}}$	X Bit	None
Vector base+ 0xF2	—	$\overline{\text{IRQ}}$	I bit	IRQCR (IRQEN)
Vector base+ 0xF0	0x78	Real time interrupt	I bit	CRGINT (RTIE)
Vector base+ 0xEE	0x77	Enhanced capture timer channel 0	I bit	TIE (C0I)
Vector base + 0xEC	0x76	Enhanced capture timer channel 1	I bit	TIE (C1I)
Vector base+ 0xEA	0x75	Enhanced capture timer channel 2	I bit	TIE (C2I)
Vector base+ 0xE8	0x74	Enhanced capture timer channel 3	I bit	TIE (C3I)
Vector base+ 0xE6	0x73	Enhanced capture timer channel 4	I bit	TIE (C4I)
Vector base+ 0xE4	0x72	Enhanced capture timer channel 5	I bit	TIE (C5I)
Vector base + 0xE2	0x71	Enhanced capture timer channel 6	I bit	TIE (C6I)
Vector base+ 0xE0	0x70	Enhanced capture timer channel 7	I bit	TIE (C7I)
Vector base+ 0xDE	0x6F	Enhanced capture timer overflow	I bit	TSRC2 (TOF)
Vector base+ 0xDC	0x6E	Pulse accumulator A overflow	I bit	PACTL (PAOVI)
Vector base + 0xDA	0x6D	Pulse accumulator input edge	I bit	PACTL (PAI)
Vector base + 0xD8	0x6C	SPI	I bit	SPCR1 (SPIE, SPTIE)
Vector base+ 0xD6	0x6B	SCI0	I bit	SCI0CR2 (TIE, TCIE, RIE, ILIE)
Vector base + 0xD4	0x6A	SCI1	I bit	SCI1CR2 (TIE, TCIE, RIE, ILIE)
Vector base + 0xD2	0x69	ATD	I bit	ATDCTL2 (ASCIE)
Vector base + 0xD0	0x68	Reserved	I bit	Reserved
Vector base + 0xCE	0x67	Port AD	I bit	PIEAD (PIEAD7 - PIEAD0)
Vector base + 0xCC	0x66	Reserved	I bit	Reserved
Vector base + 0xCA	0x65	Modulus down counter underflow	I bit	MCCTL(MCZI)
Vector base + 0xC8	0x64	Pulse accumulator B overflow	I bit	PBCTL(PBOVI)
Vector base + 0xC6	0x63	CRG PLL lock	I bit	CRGINT(LOCKIE)
Vector base + 0xC4	0x62	CRG self-clock mode	I bit	CRGINT (SCMIE)
Vector base + 0xC2	0x61	Reserved	I bit	Reserved
Vector base + 0xC0	0x60	IIC0 bus	I bit	IB0CR (IBIE)
Vector base + 0xBE	0x5F	Reserved	I bit	Reserved

Table 1-9. Interrupt Vector Locations (Sheet 2 of 3)

Vector Address[1]	XGATE Channel ID[2]	Interrupt Source	CCR Mask	Local Enable
Vector base + 0xBC	0x5E	Reserved	I bit	Reserved
Vector base + 0xBA	0x5D	EEPROM	I bit	ECNFG (CCIE, CBEIE)
Vector base + 0xB8	0x5C	FLASH	I bit	FCNFG (CCIE, CBEIE)
Vector base + 0xB6	0x5B	CAN0 wake-up	I bit	CAN0RIER (WUPIE)
Vector base + 0xB4	0x5A	CAN0 errors	I bit	CAN0RIER (CSCIE, OVRIE)
Vector base + 0xB2	0x59	CAN0 receive	I bit	CAN0RIER (RXFIE)
Vector base + 0xB0	0x58	CAN0 transmit	I bit	CAN0TIER (TXEIE[2:0])
Vector base + 0xAE	0x57	CAN1 wake-up	I bit	CAN1RIER (WUPIE)
Vector base + 0xAC	0x56	CAN1 errors	I bit	CAN1RIER (CSCIE, OVRIE)
Vector base + 0xAA	0x55	CAN1 receive	I bit	CAN1RIER (RXFIE)
Vector base + 0xA8	0x54	CAN1 transmit	I bit	CAN1TIER (TXEIE[2:0])
Vector base + 0xA6	0x53	Reserved	I bit	Reserved
Vector base + 0xA4	0x52	Reserved	I bit	Reserved
Vector base + 0xA2	0x51	SSD4	I bit	MDC4CTL (MCZIE, AOVIE)
Vector base + 0xA0	0x50	SSD0	I bit	MDC0CTL (MCZIE, AOVIE)
Vector base + 0x9E	0x4F	SSD1	I bit	MDC1CTL (MCZIE, AOVIE)
Vector base+ 0x9C	0x4E	SSD2	I bit	MDC2CTL (MCZIE, AOVIE)
Vector base+ 0x9A	0x4D	SSD3	I bit	MDC3CTL (MCZIE, AOVIE)
Vector base + 0x98	0x4C	SSD5	I bit	MDC5CTL (MCZIE, AOVIE)
Vector base + 0x96	0x4B	Motor Control Timer Overflow	I bit	MCCTL1 (MCOCIE)
Vector base + 0x94	0x4A	Reserved	I bit	Reserved
Vector base + 0x92	0x49	Reserved	I bit	Reserved
Vector base + 0x90	0x48	Reserved	I bit	Reserved
Vector base + 0x8E	0x47	Reserved	I bit	Reserved
Vector base+ 0x8C	0x46	PWM emergency shutdown	I bit	PWMSDN (PWMIE)
Vector base + 0x8A	0x45	Reserved	I bit	Reserved
Vector base + 0x88	0x44	Reserved	I bit	Reserved
Vector base + 0x86	0x43	Reserved	I bit	Reserved
Vector base + 0x84	0x42	Reserved	I bit	Reserved
Vector base + 0x82	0x41	IIC1 Bus	I bit	IB1CR (IBIE)
Vector base + 0x80	0x40	Low-voltage interrupt (LVI)	I bit	VREGCTRL (LVIE)
Vector base + 0x7E	0x3F	Autonomous periodical interrupt (API)	I bit	VREGAPICTRL (APIE)
Vector base + 0x7C	0x3E	Reserved	I bit	Reserved
Vector base + 0x7A	0x3D	Periodic interrupt timer channel 0	I bit	PITINTE (PINTE0)

Table 1-9. Interrupt Vector Locations (Sheet 3 of 3)

Vector Address[1]	XGATE Channel ID[2]	Interrupt Source	CCR Mask	Local Enable
Vector base + 0x78	0x3C	Periodic interrupt timer channel 1	I bit	PITINTE (PINTE1)
Vector base + 0x76	0x3B	Periodic interrupt timer channel 2	I bit	PITINTE (PINTE2)
Vector base + 0x74	0x3A	Periodic interrupt timer channel 3	I bit	PITINTE (PINTE3)
Vector base + 0x72	0x39	XGATE software trigger 0	I bit	XGMCTL (XGIE)
Vector base + 0x70	0x38	XGATE software trigger 1	I bit	XGMCTL (XGIE)
Vector base + 0x6E	0x37	XGATE software trigger 2	I bit	XGMCTL (XGIE)
Vector base + 0x6C	0x36	XGATE software trigger 3	I bit	XGMCTL (XGIE)
Vector base + 0x6A	0x35	XGATE software trigger 4	I bit	XGMCTL (XGIE)
Vector base + 0x68	0x34	XGATE software trigger 5	I bit	XGMCTL (XGIE)
Vector base + 0x66	0x33	XGATE software trigger 6	I bit	XGMCTL (XGIE)
Vector base + 0x64	0x32	XGATE software trigger 7	I bit	XGMCTL (XGIE)
Vector base + 0x62	—	XGATE software error interrupt	I bit	XGMCTL (XGIE)
Vector base + 0x60	—	S12XCPU RAM access violation	I bit	RAMWPC (AVIE)
Vector base+ 0x12 to Vector base + 0x5E	—	Reserved	—	Reserved
Vector base + 0x10	—	Spurious interrupt	—	None

[1] 16 bits vector address based

[2] For detailed description of XGATE channel ID refer to XGATE block description chapter

1.6.2 Effects of Reset

When a reset occurs, MCU registers and control bits are changed to known start-up states. Refer to the respective module block description chapters for register reset states.

1.6.2.1 I/O Pins

Refer to the PIM block description chapter for reset configurations of all peripheral module ports.

1.6.2.2 Memory

The RAM array is not initialized out of reset.

1.7 COP Configuration

The COP timeout rate bits CR[2:0] and the WCOP bit in the COPCTL register are loaded on rising edge of $\overline{\text{RESET}}$ from the Flash control register FCTL (0x0107) located in the Flash EEPROM block. See Table 1-10 and Table 1-11 for coding. The FCTL register is loaded from the Flash configuration field byte at global address 0x7FFF0E during the reset sequence

NOTE

If the MCU is secured the COP timeout rate is always set to the longest period (CR[2:0] = 111) after COP reset.

Table 1-10. Initial COP Rate Configuration

NV[2:0] in FCTL Register	CR[2:0] in COPCTL Register
000	111
001	110
010	101
011	100
100	011
101	010
110	001
111	000

Table 1-11. Initial WCOP Configuration

NV[3] in FCTL Register	WCOP in COPCTL Register
1	0
0	1

1.8 ATD External Trigger Input Connection

The ATD_10B16C module includes four external trigger inputs ETRIG0, ETRIG1, ETRIG2, and ETRIG3. The external trigger feature allows the user to synchronize ATD conversion to external trigger events. Table 1-12 shows the connection of the external trigger inputs on MC9S12XHZ Family.

Table 1-12. ATD External Trigger Sources

External Trigger Input	Connectivity
ETRIG0	Pulse width modulator channel 1
ETRIG1	Pulse width modulator channel 3
ETRIG2	Periodic interrupt timer hardware trigger 0
ETRIG3	Periodic interrupt timer hardware trigger 1

Consult the ATD_10B16C block description chapter for information about the analog-to-digital converter module.

Chapter 2
Port Integration Module (S12XHZPIMV1)

2.1 Introduction

The port integration module establishes the interface between the peripheral modules including the non-multiplexed external bus interface module (S12X_EBI) and the I/O pins for all ports. It controls the electrical pin properties as well as the signal prioritization and multiplexing on shared pins.

This section covers:

- Port A, B and K associated with S12X_EBI module and the LCD driver
- Port C and D associated with S12X_EBI module
- Port E associated with S12X_EBI module, the IRQ, XIRQ interrupt inputs, and the LCD driver
- Port AD associated with ATD module (channels 7 through 0) and keyboard wake-up interrupts
- Port L connected to the LCD driver and ATD (channels 15 through 8) modules
- Port M connected to 2 CAN modules
- Port P connected to 1 SCI, 2 IIC and PWM modules
- Port S connected to 2 SCI and 1 SPI modules
- Port T connected to 2 IIC, 1 ECT and LCD driver modules
- Port U, V and W associated with PWM motor control and stepper stall detect modules

Each I/O pin can be configured by several registers: input/output selection, drive strength reduction, enable and select of pull resistors, wired-or mode selection, interrupt enable, and/or status flags.

2.1.1 Features

A standard port has the following minimum features:

- Input/output selection
- 5-V output drive with two selectable drive strength (or slew rates)
- 5-V digital and analog input
- Input with selectable pull-up or pull-down device

Optional features:

- Open drain for wired-OR connections
- Interrupt input with glitch filtering

2.1.2 Block Diagram

Figure 2-1 is a block diagram of the S12XHZPIM

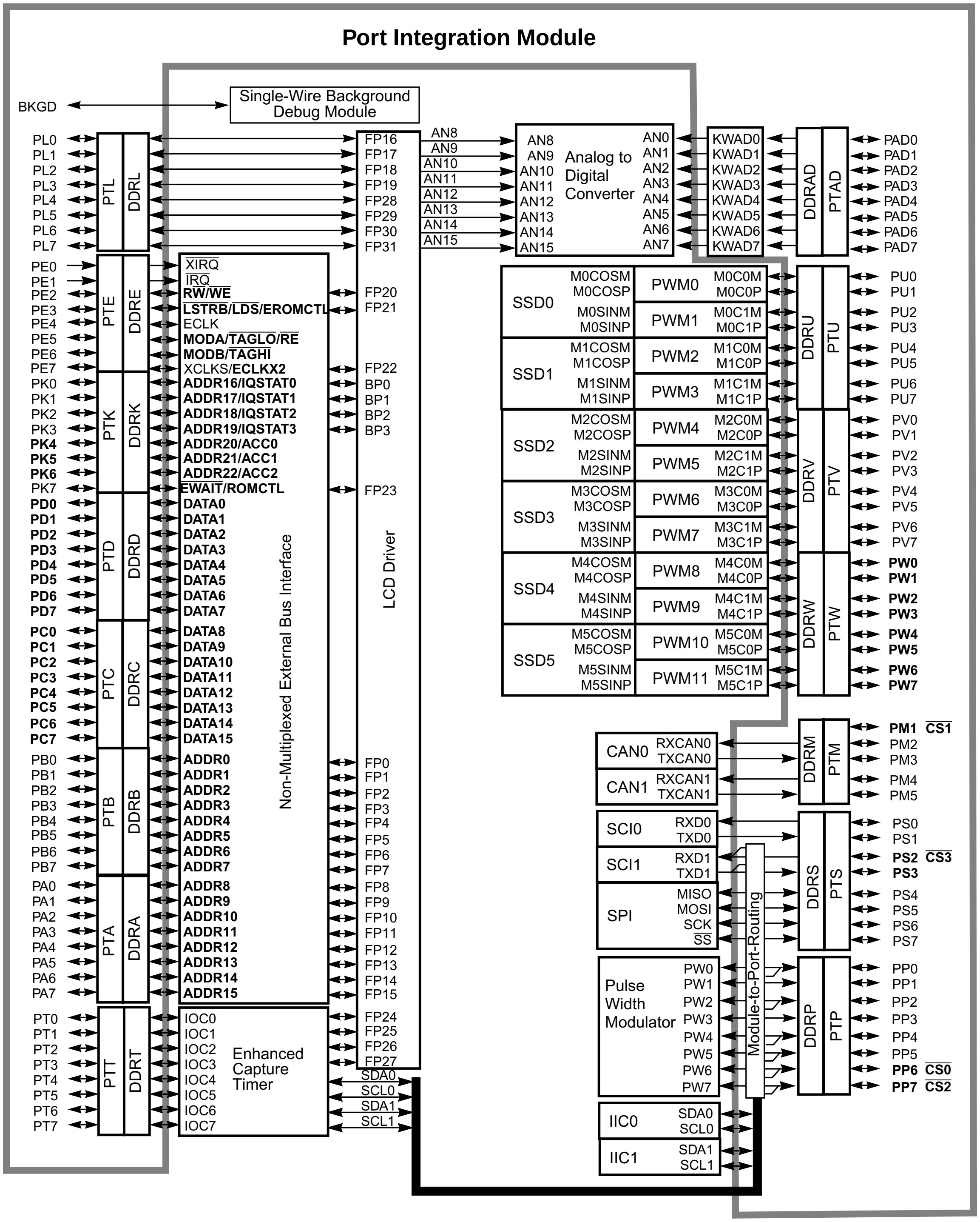


Figure 2-1. S12XHZPIM Block Diagram

2.2 External Signal Description

This section lists and describes the signals that connect off chip.

Table 2-1 shows all the pins and their functions that are controlled by the S12XHZPIM. The order in which the pin functions are listed represents the functions priority (top – highest priority, bottom – lowest priority).

Table 2-1. Detailed Signal Descriptions (Sheet 1 of 6)

Port	Pin Name	Pin Function and Priority	I/O	Description	Pin Function after Reset
—	BKGD	MODC	I	MODC input during $\overline{\text{RESET}}$	BKGD
		BKGD	I/O	S12X_BDM communication pin	
A	PA[7:0]	ADDR[15:8] mux IVD[15:8]	O	High-order external bus address output (multiplexed with IVIS data)	Mode dependent
		FP[15:8]	O	LCD frontplane driver	
		GPIO	I/O	General-purpose I/O	
B	PB[7:1]	ADDR[7:1] mux IVD[7:1]	O	Low-order external bus address output (multiplexed with IVIS data)	Mode dependent
		FP[7:1]	O	LCD frontplane driver	
		GPIO	I/O	General-purpose I/O	
	PB[0]	ADDR0 mux IVD0	O	Low-order external bus address output (multiplexed with IVIS data)	
		$\overline{\text{UDS}}$	O	Upper data strobe	
		FP[0]	O	LCD frontplane driver	
		GPIO	I/O	General-purpose I/O	
C	PC[7:0]	DATA[15:8]	I/O	High-order bidirectional data input/output Configurable for reduced input threshold	Mode dependent
		GPIO	I/O	General-purpose I/O	
D	PD[7:0]	DATA[7:0]	I/O	Low-order bidirectional data input/output Configurable for reduced input threshold	Mode dependent
		GPIO	I/O	General-purpose I/O	

Table 2-1. Detailed Signal Descriptions (Sheet 2 of 6)

Port	Pin Name	Pin Function and Priority	I/O	Description	Pin Function after Reset
E	PE[7]	XCLKS	I	External clock selection input during $\overline{\text{RESET}}$	Mode dependent
		ECLKX2	O	Free-running clock output at Core Clock rate (ECLK x 2)	
		FP[22]	O	LCD frontplane driver	
		GPIO	I/O	General-purpose I/O	
	PE[6]	MODB	I	MODB input during $\overline{\text{RESET}}$	
		$\overline{\text{TAGHI}}$	I	Instruction tagging low pin Configurable for reduced input threshold	
		GPIO	I/O	General-purpose I/O	
	PE[5]	MODA	I	MODA input during $\overline{\text{RESET}}$	
		$\overline{\text{RE}}$	O	Read enable signal	
		$\overline{\text{TAGLO}}$	I	Instruction tagging low pin Configurable for reduced input threshold	
		GPIO	I/O	General-purpose I/O	
	PE[4]	ECLK	O	Free-running clock output at the Bus Clock rate or programmable divided in normal modes	
		GPIO	I/O	General-purpose I/O	
	PE[3]	EROMCTL	I	EROMON bit control input during $\overline{\text{RESET}}$	
		$\overline{\text{LSTRB}}$	O	Low strobe bar output	
		$\overline{\text{LDS}}$	O	Lower data strobe	
		FP[21]	O	LCD frontplane driver	
		GPIO	I/O	General-purpose I/O	
	PE[2]	R/$\overline{\text{W}}$	O	Read/write output for external bus	
		$\overline{\text{WE}}$	O	Write enable	
		FP[20]	O	LCD frontplane driver	
		GPIO	I/O	General-purpose I/O	
	PE[1]	$\overline{\text{IRQ}}$	I	Maskable level or falling edge-sensitive interrupt input	
		GPIO	I/O	General-purpose I/O	
	PE[0]	$\overline{\text{XIRQ}}$	I	Non-maskable level-sensitive interrupt input	
		GPIO	I/O	General-purpose I/O	
K	PK[7]	$\overline{\text{ROMCTL}}$	I	ROMON bit control input during $\overline{\text{RESET}}$	Mode dependent
		EWAIT	I	External Wait signal Configurable for reduced input threshold	
		FP[23]	O	LCD frontplane driver	
		GPIO	I/O	General-purpose I/O	
	PK[6:4]	ADDR[22:20] mux ACC[2:0]	O	Extended external bus address output (multiplexed with master access output)	
		GPIO	I/O	General-purpose I/O	
	PK[3:0]	ADDR[19:16] mux IQSTAT[3:0]	O	Extended external bus address output (multiplexed with instruction pipe status bits)	
		BP[3:0]	O	LCD backplane driver	
		GPIO	I/O	General-purpose I/O	
AD	PAD[7:0]	AN[7:0]	I	Analog-to-digital converter input channel	GPIO
		KWAD[7:0]	I	Keyboard wake-up interrupt	
		GPIO	I/O	General-purpose I/O	

Table 2-1. Detailed Signal Descriptions (Sheet 3 of 6)

Port	Pin Name	Pin Function and Priority	I/O	Description	Pin Function after Reset
L	PL[7:4]	FP[31:28]	O	LCD frontplane driver	GPIO
		AN[15:12]	I	Analog-to-digital converter input channel	
		GPIO	I/O	General-purpose I/O	
	PL[3:0]	FP[19:16]	O	LCD frontplane driver	
		AN[11:8]	I	Analog-to-digital converter input channel	
		GPIO	I/O	General-purpose I/O	
M	PM[5]	TXCAN1	O	MSCAN1 transmit pin	GPIO
		GPIO	I/O	General-purpose I/O	
	PM[4]	RXCAN1	I	MSCAN1 receive pin	
		GPIO	I/O	General-purpose I/O	
	PM[3]	TXCAN0	O	MSCAN0 transmit pin	
		GPIO	I/O	General-purpose I/O	
	PM[2]	RXCAN0	I	MSCAN0 receive pin	
		GPIO	O	General-purpose I/O	
	PM[1]	$\overline{\text{CS1}}$	O	Chip select 1	
		GPIO	I/O	General-purpose I/O	
P	PP[7]	$\overline{\text{CS2}}$	O	Chip select 2	GPIO
		PWM7	I/O	Pulse-width modulator channel 7 and emergency shutdown input	
		SCL1	I/O	Inter-integrated circuit 1 serial clock line	
		GPIO	I/O	General-purpose I/O	
	PP[6]	$\overline{\text{CS0}}$	O	Chip select 0	
		PWM6	O	Pulse-width modulator channel 6	
		SDA1	I/O	Inter-integrated circuit 1 serial data line	
		GPIO	I/O	General-purpose I/O	
	PP[5]	PWM5	I/O	Pulse-width modulator channel 5 and emergency shutdown input	
		SCL0	I/O	Inter-integrated circuit 0 serial clock line	
		GPIO	I/O	General-purpose I/O	
	PP[4]	PWM4	O	Pulse-width modulator channel 4	
		SDA0	I/O	Inter-integrated circuit 0 serial data line	
		GPIO	I/O	General-purpose I/O	
	PP[3]	PWM3	O	Pulse-width modulator channel 3	
		GPIO	I/O	General-purpose I/O	
	PP[2]	PWM2	O	Pulse-width modulator channel 2	
		RXD1	I	Serial communication interface 1 receive pin	
		GPIO	I/O	General-purpose I/O	
	PP[1]	PWM1	O	Pulse-width modulator channel 1	
		GPIO	I/O	General-purpose I/O	
	PP[0]	PWM0	O	Pulse-width modulator channel 0	
		TXD1	O	Serial communication interface 1 transmit pin	
		GPIO	I/O	General-purpose I/O	

Table 2-1. Detailed Signal Descriptions (Sheet 4 of 6)

Port	Pin Name	Pin Function and Priority	I/O	Description	Pin Function after Reset
S	PS[7]	$\overline{\text{SS}}$	I/O	Serial peripheral interface slave select input/output in master mode, input in slave mode	GPIO
		GPIO	I/O	General-purpose I/O	
	PS[6]	SCK	I/O	Serial peripheral interface serial clock pin	
		GPIO	I/O	General-purpose I/O	
	PS[5]	MOSI	I/O	Serial peripheral interface master out/slave in pin	
		GPIO	I/O	General-purpose I/O	
	PS[4]	MISO	I/O	Serial peripheral interface master in/slave out pin	
		GPIO	I/O	General-purpose I/O	
	PS[3]	TXD1	O	Serial communication interface 1 transmit pin	
		GPIO	I/O	General-purpose I/O	
	PS[2]	$\overline{\text{CS3}}$	O	Chip select 3	
		RXD1	I	Serial communication interface 1 receive pin	
		GPIO	I/O	General-purpose I/O	
	PS[1]	TXD0	O	Serial communication interface 0 transmit pin	
		GPIO	I/O	General-purpose I/O	
	PS[0]	RXD0	I	Serial communication interface 0 receive pin	
		GPIO	I/O	General-purpose I/O	
T	PT[7]	IOC7	I/O	Timer channel	GPIO
		SCL1	I/O	Inter-integrated circuit 1 serial clock line	
		GPIO	I/O	General-purpose I/O	
	PT[6]	IOC7	I/O	Timer channel	
		SDA1	I/O	Inter-integrated circuit 1 serial data line	
		GPIO	I/O	General-purpose I/O	
	PT[5]	IOC5	I/O	Timer channel	
		SCL0	I/O	Inter-integrated circuit 0 serial clock line	
		GPIO	I/O	General-purpose I/O	
	PT[4]	IOC4	I/O	Timer channel	
		SDA0	I/O	Inter-integrated circuit 0 serial data line	
		GPIO	I/O	General-purpose I/O	
	PT[3:0]	FP[27:24]	I/O	LCD frontplane driver	
		IOC[3:0]	I/O	Timer channel	
		GPIO	I/O	General-purpose I/O	

Table 2-1. Detailed Signal Descriptions (Sheet 5 of 6)

Port	Pin Name	Pin Function and Priority	I/O	Description	Pin Function after Reset
U	PU[7]	M1SINP	O	SSD1 Sine+ Node	GPIO
		M1C1P	O	PWM motor controller channel 3	
		GPIO	I/O	General-purpose I/O	
	PU[6]	M1SINM	O	SSD1 Sine- Node	
		M1C1M	O	PWM motor controller channel 3	
		GPIO	I/O	General-purpose I/O	
	PU[5]	M1COSP	O	SSD1 Cosine+ Node	
		M1C0P	O	PWM motor controller channel 2	
		GPIO	I/O	General-purpose I/O	
	PU[4]	M1COSM	O	SSD1 Cosine- Node	
		M1C0M	O	PWM motor controller channel 2	
		GPIO	I/O	General-purpose I/O	
	PU[3]	M0SINP	O	SSD0 Sine+ Node	
		M0C1P	O	PWM motor controller channel 1	
		GPIO	I/O	General-purpose I/O	
	PU[2]	M0SINM	O	SSD0 Sine- Node	
		M0C1M	O	PWM motor controller channel 1	
		GPIO	I/O	General-purpose I/O	
	PU[1]	M0COSP	O	SSD0 Cosine+ Node	
		M0C0P	O	PWM motor controller channel 0	
		GPIO	I/O	General-purpose I/O	
	PU[0]	M0COSM	O	SSD0 Cosine- Node	
		M0C0M	O	PWM motor controller channel 0	
		GPIO	I/O	General-purpose I/O	
V	PV[7]	M3SINP	O	SSD3 sine+ node	GPIO
		M3C1P	O	PWM motor controller channel 7	
		GPIO	I/O	General-purpose I/O	
	PV[6]	M3SINM	O	SSD3 sine- node	
		M3C1M	O	PWM motor controller channel 7	
		GPIO	I/O	General-purpose I/O	
	PV[5]	M3COSP	O	SSD3 cosine+ node	
		M3C0P	O	PWM motor controller channel 6	
		GPIO	I/O	General-purpose I/O	
	PV[4]	M3COSM	O	SSD3 cosine- node	
		M3C0M	O	PWM motor controller channel 6	
		GPIO	I/O	General-purpose I/O	
	PV[3]	M2SINP	O	SSD2 sine+ node	
		M2C1P	O	PWM motor controller channel 5	
		GPIO	I/O	General-purpose I/O	
	PV[2]	M2SINM	O	SSD2 sine- node	
		M2C1M	O	PWM motor controller channel 5	
		GPIO	I/O	General-purpose I/O	
	PV[1]	M2COSP	O	SSD2 cosine+ node	
		M2C0P	O	PWM motor controller channel 4	
		GPIO	I/O	General-purpose I/O	
	PV[0]	M2COSM	O	SSD2 cosine- node	
		M2C0M	O	PWM motor controller channel 4	
		GPIO	I/O	General-purpose I/O	

Table 2-1. Detailed Signal Descriptions (Sheet 6 of 6)

Port	Pin Name	Pin Function and Priority	I/O	Description	Pin Function after Reset
W	PW[7]	M5SINP	O	SSD5 sine+ node	GPIO
		M5C1P	O	PWM motor controller channel 11	
		GPIO	I/O	General-purpose I/O	
	PW[6]	M5SINM	O	SSD5 sine- node	
		M5C1M	O	PWM motor controller channel 11	
		GPIO	I/O	General-purpose I/O	
	PW[5]	M5COSP	O	SSD5 cosine+ node	
		M5C0P	O	PWM motor controller channel 10	
		GPIO	I/O	General-purpose I/O	
	PW[4]	M5COSM	O	SSD5 cosine- node	
		M5C0M	O	PWM motor controller channel 10	
		GPIO	I/O	General-purpose I/O	
	PW[3]	M4SINP	O	SSD4 sine+ node	
		M4C1P	O	PWM motor controller channel 9	
		GPIO	I/O	General-purpose I/O	
	PW[2]	M4SINM	O	SSD4 sine- node	
		M4C1M	O	PWM motor controller channel 9	
		GPIO	I/O	General-purpose I/O	
	PW[1]	M4COSP	O	SSD4 cosine+ node	
		M4C0P	O	PWM motor controller channel 8	
		GPIO	I/O	General-purpose I/O	
	PW[0]	M4COSM	O	SSD4 cosine- node	
		M4C0M	O	PWM motor controller channel 8	
		GPIO	I/O	General-purpose I/O	

2.3 Memory Map and Register Definition

This section provides a detailed description of all registers. Table 2-2 is a standard memory map of port integration module.

Table 2-2. S12XHZPIM Memory Map

Address Offset	Use	Access
0x0000	Port A I/O Register (PTA)	R/W
0x0001	Port B I/O Register (PTB)	R/W
0x0002	Port A Data Direction Register (DDRA)	R/W
0x0003	Port B Data Direction Register (DDRB)	R/W
0x0004	Port C I/O Register (PTC)	R/W
0x0005	Port D I/O Register (PTD)	R/W
0x0006	Port C Data Direction Register (DDRC)	R/W
0x0007	Port D Data Direction Register (DDRD)	R/W
0x0008	Port E I/O Register (PTE)	R/W
0x0009	Port E Data Direction Register (DDRE)	R/W
0x000A - 0x000B	Non-PIM address range	—
0x000C	Pull Up/Down Control Register (PUCR)	R/W
0x000D	Reduced Drive Register (RDRIV)	R/W
0x000E - 0x001B	Non-PIM address range	—
0x001C	ECLK Control Register (ECLKCR)	R/W
0x001D	Reserved	—
0x001E	IRQ Control Register (IRQCR)	R/W
0x001F	Slew Rate Control Register (SRCR)	R/W
0x0020 - 0x0031	Non-PIM address range	—
0x0032	Port K I/O Register (PTK)	R/W
0x0033	Port K Data Direction Register (DDRK)	R/W
0x0034 - 0x01FF	Non-PIM address range	—
0x0200	Port T I/O Register (PTT)	R/W
0x0201	Port T Input Register (PTIT)	R
0x0202	Port T Data Direction Register (DDRT)	R/W
0x0203	Port T Reduced Drive Register (RDRT)	R/W
0x0204	Port T Pull Device Enable Register (PERT)	R/W
0x0205	Port T Polarity Select Register (PPST)	R/W
0x0206	Port T Wired-OR Mode Register (WOMT)	R/W
0x0207	Port T Slew Rate Register (SRRT)	R/W

Table 2-2. S12XHZPIM Memory Map (continued)

Address Offset	Use	Access
0x0208	Port S I/O Register (PTS)	R/W
0x0209	Port S Input Register (PTIS)	R
0x020A	Port S Data Direction Register (DDRS)	R/W
0x020B	Port S Reduced Drive Register (RDRS)	R/W
0x020C	Port S Pull Device Enable Register (PERS)	R/W
0x020D	Port S Polarity Select Register (PPSS)	R/W
0x020E	Port S Wired-OR Mode Register (WOMS)	R/W
0x020F	Port S Slew Rate Register (SRRS)	R/W
0x0210	Port M I/O Register (PTM)	R/W
0x0211	Port M Input Register (PTIM)	R
0x0212	Port M Data Direction Register (DDRM)	R/W
0x0213	Port M Reduced Drive Register (RDRM)	R/W
0x0214	Port M Pull Device Enable Register (PERM)	R/W
0x0215	Port M Polarity Select Register (PPSM)	R/W
0x0216	Port M Wired-OR Mode Register (WOMM)	R/W
0x0217	Port M Slew Rate Register (SRRM)	R/W
0x0218	Port P I/O Register (PTP)	R/W
0x0219	Port P Input Register (PTIP)	R
0x021A	Port P Data Direction Register (DDRP)	R/W
0x021B	Port P Reduced Drive Register (RDRP)	R/W
0x021C	Port P Pull Device Enable Register (PERP)	R/W
0x021D	Port P Polarity Select Register (PPSP)	R/W
0x021E	Port P Wired-OR Mode Register (WOMP)	R/W
0x021F	Port P Slew Rate Register (SRRP)	R/W
0x0220 - 0x022F	Reserved	—
0x0230	Port L I/O Register (PTL)	R/W
0x0231	Port L Input Register (PTIL)	R
0x0232	Port L Data Direction Register (DDRL)	R/W
0x0233	Port L Reduced Drive Register (RDRL)	R/W
0x0234	Port L Pull Device Enable Register (PERL)	R/W
0x0235	Port L Polarity Select Register (PPSL)	R/W
0x0236	Reserved	—
0x0237	Port L Slew Rate Register (SRRL)	R/W
0x0238	Port U I/O Register (PTU)	R/W
0x0239	Port U Input Register (PTIU)	R
0x023A	Port U Data Direction Register (DDRU)	R/W
0x023B	Port U Slew Rate Register (SRRU)	R/W
0x023C	Port U Pull Device Enable Register (PERU)	R/W
0x023D	Port U Polarity Select Register (PPSU)	R/W
0x023E - 0x023F	Reserved	—

Table 2-2. S12XHZPIM Memory Map (continued)

Address Offset	Use	Access
0x0240	Port V I/O Register (PTV)	R/W
0x0241	Port V Input Register (PTIV)	R
0x0242	Port V Data Direction Register (DDRV)	R/W
0x0243	Port V Slew Rate Register (SRRV)	R/W
0x0244	Port V Pull Device Enable Register (PERV)	R/W
0x0245	Port V Polarity Select Register (PPSV)	R/W
0x0246 - 0x0247	Reserved	—
0x0248	Port W I/O Register (PTW)	R/W
0x0249	Port W Input Register (PTIW)	R
0x024A	Port W Data Direction Register (DDRW)	R/W
0x024B	Port W Slew Rate Register (SRRW)	R/W
0x024C	Port W Pull Device Enable Register (PERW)	R/W
0x024D	Port W Polarity Select Register (PPSW)	R/W
0x024E - 0x0250	Reserved	—
0x0251	Port AD I/O Register (PTAD)	R/W
0x0252	Reserved	—
0x0253	Port AD Input Register (PTIAD)	R
0x0254	Reserved	—
0x0255	Port AD Data Direction Register (DDRAD)	R/W
0x0256	Reserved	—
0x0257	Port AD Reduced Drive Register (RDRAD)	R/W
0x0258	Reserved	—
0x0259	Port AD Pull Device Enable Register (PERAD)	R/W
0x025A	Reserved	—
0x025B	Port AD Polarity Select Register (PPSAD)	R/W
0x025C	Reserved	—
0x025D	Port AD Interrupt Enable Register (PIEAD)	R/W
0x025E	Reserved	—
0x025F	Port AD Interrupt Flag Register (PIFAD)	R/W
0x0260 - 0x027F	Reserved	—

2.3.1 Port A and Port B

Port A and port B are associated with the external address bus outputs ADDR15-ADDR0, the external read visibility IVD15-IVD0 and the liquid crystal display (LCD) driver. Each pin is assigned to these functions according to the following priority: LCD Driver > XEBI > general-purpose I/O.

If the corresponding LCD frontplane drivers are enabled (and LCD module is enabled), the FP[15:0] outputs of the LCD module are available on port B and port A pins.

Refer to the LCD block description chapter for information on enabling and disabling the LCD and its frontplane drivers.Refer to the S12X_EBI block description chapter for information on external bus.

During reset, port A and port B pins are configured as inputs with pull down.

2.3.1.1 Port A I/O Register (PTA)

Module Base + 0x0051

	7	6	5	4	3	2	1	0
R W	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
XEBI:	ADDR15 mux IVD15	ADDR14 mux IVD14	ADDR13 mux IVD13	ADDR12 mux IVD12	ADDR11 mux IVD11	ADDR10 mux IVD10	ADDR9 mux IVD9	ADDR8 mux IVD8
LCD:	FP15	FP14	FP13	FP12	FP11	FP10	FP9	FP8
Reset	0	0	0	0	0	0	0	0

Figure 2-2. Port A I/O Register (PTA)

Read: Anytime. Write: Anytime.

If the associated data direction bit (DDRAx) is set to 1 (output), a read returns the value of the I/O register bit.

If the associated data direction bit (DDRAx) is set to 0 (input) and the LCD frontplane driver is enabled (and LCD module is enabled), the associated I/O register bit (PTAx) reads “1”.

If the associated data direction bit (DDRAx) is set to 0 (input) and the LCD frontplane driver is disabled (or LCD module is disabled), a read returns the value of the pin.

2.3.1.2 Port B I/O Register (PTB)

Module Base + 0x0051

	7	6	5	4	3	2	1	0
R W	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
XEBI:	ADDR7 mux IVD7	ADDR6 mux IVD6	ADDR5 mux IVD5	ADDR4 mux IVD4	ADDR3 mux IVD3	ADDR2 mux IVD2	ADDR1 mux IVD1	ADDR0 mux IVD0 or $\overline{\text{UDS}}$
LCD:	FP7	FP6	FP5	FP4	FP3	FP2	FP1	FP0
Reset	0	0	0	0	0	0	0	0

Figure 2-3. Port B I/O Register (PTB)

Read: Anytime. Write: Anytime.

If the associated data direction bit (DDRBx) is set to 1 (output), a read returns the value of the I/O register bit.

If the associated data direction bit (DDRBx) is set to 0 (input) and the LCD frontplane driver is enabled (and LCD module is enabled), the associated I/O register bit (PTBx) reads “1”.

If the associated data direction bit (DDRBx) is set to 0 (input) and the LCD frontplane driver is disabled (or LCD module is disabled), a read returns the value of the pin.

2.3.1.3 Port A Data Direction Register (DDRA)

Module Base + 0x0055

	7	6	5	4	3	2	1	0
R W	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
Reset	0	0	0	0	0	0	0	0

Figure 2-4. Port A Data Direction Register (DDRA)

Read: Anytime. Write: Anytime.

This register configures port pins PA[7:0] as either input or output.If a LCD frontplane driver is enabled (and LCD module is enabled), it outputs an analog signal to the corresponding pin and the associated Data Direction Register bit has no effect. If a LCD frontplane driver is disabled (or LCD module is disabled), the corresponding Data Direction Register bit reverts to control the I/O direction of the associated pin.

Table 2-3. DDRA Field Descriptions

Field	Description
7:0 DDRA[7:0]	**Data Direction Port A** 0 Associated pin is configured as input. 1 Associated pin is configured as output.

2.3.1.4 Port B Data Direction Register (DDRB)

Module Base + 0x0055

	7	6	5	4	3	2	1	0
R W	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
Reset	0	0	0	0	0	0	0	0

Figure 2-5. Port B Data Direction Register (DDRB)

Read: Anytime. Write: Anytime.

This register configures port pins PB[7:0] as either input or output.If a LCD frontplane driver is enabled (and LCD module is enabled), it outputs an analog signal to the corresponding pin and the associated Data Direction Register bit has no effect. If a LCD frontplane driver is disabled (or LCD module is disabled), the corresponding Data Direction Register bit reverts to control the I/O direction of the associated pin.

Table 2-4. DDRB Field Descriptions

Field	Description
7:0 DDRB[7:0]	**Data Direction Port B** 0 Associated pin is configured as input. 1 Associated pin is configured as output.

2.3.2 Port C and Port D

Port C and port D pins can be used for either general-purpose I/O or the external data bus input/outputs DATA15-DATA0. Refer to the S12X_EBI block description chapter for information on external bus.

2.3.2.1 Port C I/O Register (PTC)

Module Base + 0x0051

	7	6	5	4	3	2	1	0
R W	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
XEBI:	DATA15	DATA14	DATA13	DATA12	DATA11	DATA10	DATA9	DATA8
Reset	0	0	0	0	0	0	0	0

Figure 2-6. Port C I/O Register (PTC)

Read: Anytime. Write: Anytime.

If the data direction bit of the associated I/O pin (DDRCx) is set to 1 (output), a write to the corresponding I/O Register bit sets the value to be driven to the Port C pin. If the data direction bit of the associated I/O pin (DDRCx) is set to 0 (input), a write to the corresponding I/O Register bit takes place but has no effect on the Port C pin.

If the associated data direction bit (DDRCx) is set to 1 (output), a read returns the value of the I/O register bit. If the associated data direction bit (DDRCx) is set to 0 (input), a read returns the value of the pin.

2.3.2.2 Port D I/O Register (PTD)

Module Base + 0x0051

	7	6	5	4	3	2	1	0
R W	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
XEBI:	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0
Reset	0	0	0	0	0	0	0	0

Figure 2-7. Port D I/O Register (PTD)

Read: Anytime. Write: Anytime.

If the data direction bit of the associated I/O pin (DDRDx) is set to 1 (output), a write to the corresponding I/O Register bit sets the value to be driven to the Port D pin. If the data direction bit of the associated I/O pin (DDRDx) is set to 0 (input), a write to the corresponding I/O Register bit takes place but has no effect on the Port D pin.

If the associated data direction bit (DDRDx) is set to 1 (output), a read returns the value of the I/O register bit. If the associated data direction bit (DDRDx) is set to 0 (input), a read returns the value of the pin.

2.3.2.3 Port C Data Direction Register (DDRC)

Module Base + 0x0055

	7	6	5	4	3	2	1	0
R W	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
Reset	0	0	0	0	0	0	0	0

Figure 2-8. Port C Data Direction Register (DDRC)

Read: Anytime. Write: Anytime.

This register configures port pins PC[7:0] as either input or output.

Table 2-5. DDRC Field Descriptions

Field	Description
7:0 DDRC[7:0]	**Data Direction Port C** 0 Associated pin is configured as input. 1 Associated pin is configured as output.

2.3.2.4 Port D Data Direction Register (DDRD)

Module Base + 0x0055

	7	6	5	4	3	2	1	0
R W	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
Reset	0	0	0	0	0	0	0	0

Figure 2-9. Port D Data Direction Register (DDRD)

Read: Anytime. Write: Anytime.

This register configures port pins PD[7:0] as either input or output.

Table 2-6. DDRD Field Descriptions

Field	Description
7:0 DDRD[7:0]	**Data Direction Port D** 0 Associated pin is configured as input. 1 Associated pin is configured as output.

2.3.3 Port E

Port E pins can be used for either general-purpose I/O, or the liquid crystal display (LCD) driver, or the external bus control outputs R/$\overline{\text{W}}$, $\overline{\text{WE}}$, $\overline{\text{LSTRB}}$, $\overline{\text{LDS}}$ and $\overline{\text{RE}}$, the free running clock outputs ECLK and ECLKX2, or the inputs $\overline{\text{TAGHI}}$, $\overline{\text{TAGLO}}$, MODA, MODB, EROMCTL, XCLKS and interrupts $\overline{\text{IRQ}}$ and $\overline{\text{XIRQ}}$. Refer to the LCD block description chapter for information on enabling and disabling the LCD and its frontplane drivers. Refer to the S12X_EBI block description chapter for information on external bus.

Port E pin PE[7] can be used for either general-purpose I/O, or as the free-running clock ECLKX2 output running at the core clock rate, or the frontplane driver FP22. The clock ECLKX2 output is always enabled in emulation modes.

Port E pin PE[4] can be used for either general-purpose I/O or as the free-running clock ECLK output running at the bus clock rate or at the programmed divided clock rate. The clock output is always enabled in emulation modes.

Port E pin PE[1] can be used for either general-purpose input or as the level- or falling edge-sensitive $\overline{\text{IRQ}}$ interrupt inpu. $\overline{\text{IRQ}}$ will be enabled by setting the IRQEN configuration bit and clearing the I-bit in the CPU's condition code register. It is inhibited at reset so this pin is initially configured as a simple input with a pull-up.

Port E pin PE[0] can be used for either general-purpose input or as the level-sensitive $\overline{\text{XIRQ}}$ interrupt input. $\overline{\text{XIRQ}}$ can be enabled by clearing the X-bit in the CPU's condition code register. It is inhibited at reset so this pin is initially configured as a high-impedance input with a pull-up.

2.3.3.1 Port E I/O Register (PTE)

Module Base + 0x0051

	7	6	5	4	3	2	1	0
R W	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
XEBI:	XCLKS[1] or ECLKX2	MODB[1] or $\overline{\text{TAGHI}}$	MODA[1] or $\overline{\text{TAGLO}}$ or $\overline{\text{RE}}$	ECLK	EROMCTL[1] or $\overline{\text{LSTRB}}$ or $\overline{\text{LDS}}$	R/$\overline{\text{W}}$ or $\overline{\text{WE}}$	$\overline{\text{IRQ}}$	$\overline{\text{XIRQ}}$
LCD:	FP22				FP21	FP20		
Reset	0	0	0	0	0	0	—[2]	—[2]

Figure 2-10. Port E I/O Register (PTE)

[1] Function active when $\overline{\text{RESET}}$ asserted.

[2] These registers are reset to zero. Two bus clock cycles after reset release the register values are updated with the associated pin values.

Read: Anytime. Write: Anytime.

If the associated data direction bit (DDREx) is set to 1 (output), a read returns the value of the I/O register bit.

If the associated data direction bit (DDREx) is set to 0 (input) and the LCD frontplane driver is enabled (and LCD module is enabled), the associated I/O register bit (PTEx) reads “1”.

If the associated data direction bit (DDREx) is set to 0 (input) and the LCD frontplane driver is disabled (or LCD module is disabled), a read returns the value of the pin.

2.3.3.2 Port E Data Direction Register (DDRE)

Module Base + 0x0055

	7	6	5	4	3	2	1	0
R	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	0	0
W							▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Reserved or Unimplemented

Figure 2-11. Port E Data Direction Register (DDRE)

Read: Anytime. Write: Anytime.

This register configures port pins PE[7:0] as either input or output.If a LCD frontplane driver is enabled (and LCD module is enabled), it outputs an analog signal to the corresponding pin and the associated Data Direction Register bit has no effect. If a LCD frontplane driver is disabled (or LCD module is disabled), the corresponding Data Direction Register bit reverts to control the I/O direction of the associated pin.

Table 2-7. DDRE Field Descriptions

Field	Description
7:2 DDRE[7:2]	**Data Direction Port E** 0 Associated pin is configured as input. 1 Associated pin is configured as output.

2.3.4 Port K

Port K pins can be used for either general-purpose I/O, or the liquid crystal display (LCD) driver, or the external address bus outputs ADDR22-ADDR16 muxed with master access output ACC2-ACC0 and instruction pipe signals IQSTAT3-IQSTAT0, or inputs $\overline{\text{EWAIT}}$ and ROMCTL. Refer to the LCD block description chapter for information on enabling and disabling the LCD and its frontplane drivers. Refer to the S12X_EBI block description chapter for information on external bus.

2.3.4.1 Port K I/O Register (PTK)

Module Base + 0x0051

	7	6	5	4	3	2	1	0
R W	PTK7	PTK6	PTK5	PTK4	PTK3	PTK2	PTK1	PTK0
XEBI:	ROMCTL[1] or $\overline{\text{EWAIT}}$	ADDR22 or ACC2	ADDR21 or ACC1	ADDR20 or ACC0	ADDR19 or IQSTAT3	ADDR18 or IQSTAT2	ADDR17 or IQSTAT1	ADDR16 or IQSTAT0
LCD:	FP23				BP3	BP2	BP1	BP0
Reset	0	0	0	0	0	0	0	0

Figure 2-12. Port K I/O Register (PTK)

[1] Function active when $\overline{\text{RESET}}$ asserted.

Read: Anytime. Write: Anytime.

If the associated data direction bit (DDRKx) is set to 1 (output), a read returns the value of the I/O register bit.

If the associated data direction bit (DDRKx) is set to 0 (input) and the LCD frontplane driver is enabled (and LCD module is enabled), the associated I/O register bit (PTKx) reads “1”.

If the associated data direction bit (DDRKx) is set to 0 (input) and the LCD frontplane driver is disabled (or LCD module is disabled), a read returns the value of the pin.

2.3.4.2 Port K Data Direction Register (DDRK)

Module Base + 0x0055

	7	6	5	4	3	2	1	0
R W	DDRK7	DDRK6	DDRK5	DDRK4	DDRK3	DDRK2	DDRK1	DDRK0
Reset	0	0	0	0	0	0	0	0

Figure 2-13. Port K Data Direction Register (DDRK)

Read: Anytime. Write: Anytime.

This register configures port pins PK[7:0] as either input or output.If a LCD frontplane driver is enabled (and LCD module is enabled), it outputs an analog signal to the corresponding pin and the associated Data Direction Register bit has no effect. If a LCD frontplane driver is disabled (or LCD module is disabled), the corresponding Data Direction Register bit reverts to control the I/O direction of the associated pin.

Table 2-8. DDRK Field Descriptions

Field	Description
7:0 DDRK[7:0]	**Data Direction Port K** 0 Associated pin is configured as input. 1 Associated pin is configured as output.

2.3.5 Miscellaneous registers

2.3.5.1 Pull Up/Down Control Register (PUCR)

Module Base + 0x0055

	7	6	5	4	3	2	1	0
R	PUPKE	BKGPE	0	PUPEE	PUPDE	PUPCE	PUPBE	PUPAE
W								
Reset	1	1	0	1	0	0	1	1

Figure 2-14. Pull Up/Down Control Register (PUCR)

Read: Anytime. Write: Anytime except BKPUE which is writable in special test mode only.

This register is used to enable pull up/down devices for the associated ports A, B, C, D, E and K. Pull up/down devices are assigned on a per-port basis and apply to any pin in the corresponding port currently configured as an input.

Table 2-9. PUCR Field Descriptions

Field	Description
7 PUPKE	**Pull-down Port K Enable** 0 Port K pull-down devices are disabled. 1 Enable pull-down devices for Port K input pins.
6 BKPUE	**BKGD Pin Pull-up Enable** 0 BKGD pull-up device is disabled. 1 Enable pull-up device on BKGD pin.
4 PUPEE	**Pull Port E Enable** 0 Port E pull-down devices on pins 7, 4–2 are disabled. Port E pull-up devices on pins 1–0 are disabled. 1 Enable pull-down devices for Port E input pins 7, 4–2. Enable pull-up devices for Port E input pins 1–0. **Note:** Bits 5 and 6 of Port E have pull-down devices which are only enabled during reset. This bit has no effect on these pins.
3 PUPDE	**Pull-up Port D Enable** 0 Port D pull-up devices are disabled. 1 Enable pull-up devices for all Port D input pins.
2 PUPCE	**Pull-up Port C Enable** 0 Port C pull-up devices are disabled. 1 Enable pull-up devices for all Port C.
1 PUPBE	**Pull-down Port B Enable** 0 Port B pull-down devices are disabled. 1 Enable pull-down devices for all Port B input pins.
0 PUPAE	**Pull-down Port A Enable** 0 Port A pull-down devices are disabled. 1 Enable pull-down devices for all Port A input pins.

2.3.5.2 Reduced Drive Register (RDRIV)

Module Base + 0x0055

	7	6	5	4	3	2	1	0
R	RDPK	0	0	RDPE	RDPD	RDPC	RDPB	RDPA
W		▨	▨					
Reset	0	0	0	0	0	0	0	0

▨ = Reserved or Unimplemented

Figure 2-15. Reduced Drive Register (RDRIV)

Read: Anytime. Write: Anytime.

This register is used to select reduced drive for the pins associated with ports A, B, C, D, E, and K. If enabled, the pins drive at about 1/6 of the full drive strength. The reduced drive function is independent of which function is being used on a particular pin.

The reduced drive functionality does not take effect on the pins in emulation modes.

Table 2-10. RDRIV Field Descriptions

Field	Description
7 RDPK	**Reduced Drive of Port K** 0 All port K output pins have full drive enabled. 1 All port K output pins have reduced drive enabled.
4 RDPE	**Reduced Drive of Port E** 0 All port E output pins have full drive enabled. 1 All port E output pins have reduced drive enabled.
3 RDPD	**Reduced Drive of Port D** 0 All port D output pins have full drive enabled. 1 All port D output pins have reduced drive enabled.
2 RDPC	**Reduced Drive of Port C** 0 All Port C output pins have full drive enabled. 1 All port C output pins have reduced drive enabled.
1 RDPB	**Reduced Drive of Port B** 0 All port B output pins have full drive enabled. 1 All port B output pins have reduced drive enabled.
0 RDPA	**Reduced Drive of Port A** 0 All Port A output pins have full drive enabled. 1 All port A output pins have reduced drive enabled.

2.3.5.3 ECLK Control Register (ECLKCR)

Module Base + 0x0055

	7	6	5	4	3	2	1	0
R	NECLK	NCLKX2	0	0	0	0	EDIV1	EDIV0
W			▨	▨	▨	▨		
Reset	0[1]	1	0	0	0	0	0	0

▨ = Reserved or Unimplemented

Figure 2-16. ECLK Control Register (ECLKCR)

[1] NECLK reset value is 1 in emulation single-chip and normal single-chip modes.

Read: Anytime. Write: Anytime.

The ECLKCTL register is used to control the availability of the free-running clocks and the free-running clock divider.

Table 2-11. ECLKCTL Field Descriptions

Field	Description
7 NECLK	**No ECLK** — This bit controls the availability of a free-running clock on the ECLK pin. Clock output is always active in emulation modes and if enabled in all other operating modes. 0 ECLK enabled 1 ECLK disabled
6 NCLKX2	**No ECLKX2** — This bit controls the availability of a free-running clock on the ECLKX2 pin. This clock has a fixed rate of twice the internal bus clock. Clock output is always active in emulation modes and if enabled in all other operating modes. 0 ECLKX2 is enabled 1 ECLKX2 is disabled
1–0 EDIV[1:0]	**Free-Running ECLK Divider** — These bits determine the rate of the free-running clock on the ECLK pinr. The usage of the bits is shown in Table 2-12. Divider is always disabled in emulation modes and active as programmed in all other operating modes.

Table 2-12. Free-Running ECLK Clock Rate

EDIV[1:0]	Rate of Free-Running ECLK
00	ECLK = Bus clock rate
01	ECLK = Bus clock rate divided by 2
10	ECLK = Bus clock rate divided by 3
11	ECLK = Bus clock rate divided by 4

2.3.5.4 IRQ Control Register (IRQCR)

Module Base + 0x0055

	7	6	5	4	3	2	1	0
R	IRQE	IRQEN	0	0	0	0	0	0
W			▨	▨	▨	▨	▨	▨
Reset	0	1	0	0	0	0	0	0

▨ = Reserved or Unimplemented

Figure 2-17. Port E Data Direction Register (DDRE)

Read: See individual bit descriptions below.

Write: See individual bit descriptions below.

Table 2-13. IRQCR Field Descriptions

Field	Description
7 IRQE	**IRQ Select Edge Sensitive Only** Special modes: Read or write anytime. Normal and emulation modes: Read anytime, write once. 0 $\overline{\text{IRQ}}$ configured for low level recognition. 1 $\overline{\text{IRQ}}$ configured to respond only to falling edges. Falling edges on the $\overline{\text{IRQ}}$ pin will be detected anytime IRQE = 1. The edge detector is cleared only upon the servicing of the $\overline{\text{IRQ}}$ interrupt or a reset .
6 IRQEN	**External IRQ Enable** Read or write anytime. 0 External $\overline{\text{IRQ}}$ pin is disconnected from interrupt logic. 1 External $\overline{\text{IRQ}}$ pin is connected to interrupt logic.

2.3.5.5 Slew Rate Control Register (SRCR)

Module Base + 0x0055

	7	6	5	4	3	2	1	0
R	SRRK	0	0	SRRE	SRRD	SRRC	SRRB	SRRA
W		▨	▨					
Reset	0	0	0	0	0	0	0	0

▨ = Reserved or Unimplemented

Figure 2-18. Slew Rate Control Register (SRCR)

Read: Anytime. Write: Anytime.

This register enables the slew rate control and disables the digital input buffer for the pins associated with ports A, B, C, D, E, and K.

Table 2-14. SRCR Field Descriptions

Field	Description
7 SRRK	**Slew Rate of Port K** 0 Disables slew rate control and enables digital input buffer for all port K pins. 1 Enables slew rate control and disables digital input buffer for all port K pins.
4 SRRE	**Slew Rate of Port E** 0 Disables slew rate control and enables digital input buffer for all port E pins. 1 Enables slew rate control and disables digital input buffer for all port E pins.
3 SRRD	**Slew Rate of Port D** 0 Disables slew rate control and enables digital input buffer for all port D pins. 1 Enables slew rate control and disables digital input buffer for all port D pins.
2 SRRC	**Slew Rate of Port C** 0 Disables slew rate control and enables digital input buffer for all port C pins. 1 Enables slew rate control and disables digital input buffer for all port C pins.
1 SRRB	**Slew Rate of Port B** 0 Disables slew rate control and enables digital input buffer for all port B pins. 1 Enables slew rate control and disables digital input buffer for all port B pins.
0 SRRA	**Slew Rate of Port A** 0 Disables slew rate control and enables digital input buffer for all port A pins. 1 Enables slew rate control and disables digital input buffer for all port A pins.

2.3.6 Port AD

Port AD is associated with the analog-to-digital converter (ATD) and keyboard wake-up (KWU) interrupts . Each pin is assigned to these modules according to the following priority: ATD > KWU > general-purpose I/O.

For the pins of port AD to be used as inputs, the corresponding bits of the ATDDIEN1 register in the ATD module must be set to 1 (digital input buffer is enabled). The ATDDIEN1 register does not affect the port AD pins when they are configured as outputs.

Refer to the ATD block description chapter for information on the ATDDIEN1 register.

During reset, port AD pins are configured as high-impedance analog inputs (digital input buffer is disabled).

2.3.6.1 Port AD I/O Register (PTAD)

Module Base + 0x0051

	7	6	5	4	3	2	1	0
R W	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
KWU:	KWAD7	KWAD6	KWAD5	KWAD4	KWAD3	KWAD2	KWAD1	KWAD0
ATD:	AN7	AN6	AN55	AN4	AN3	AN2	AN1	AN0
Reset	0	0	0	0	0	0	0	0

Figure 2-19. Port AD I/O Register (PTAD)

Read: Anytime. Write: Anytime.

If the data direction bit of the associated I/O pin (DDRADx) is set to 1 (output), a write to the corresponding I/O Register bit sets the value to be driven to the Port AD pin. If the data direction bit of the associated I/O pin (DDRADx) is set to 0 (input), a write to the corresponding I/O Register bit takes place but has no effect on the Port AD pin.

If the associated data direction bit (DDRADx) is set to 1 (output), a read returns the value of the I/O register bit.

If the associated data direction bit (DDRADx) is set to 0 (input) and the associated ATDDIEN1 bits is set to 0 (digital input buffer is disabled), the associated I/O register bit (PTADx) reads “1”.

If the associated data direction bit (DDRADx) is set to 0 (input) and the associated ATDDIEN1 bits is set to 1 (digital input buffer is enabled), a read returns the value of the pin.

2.3.6.2 Port AD Input Register (PTIAD)

Module Base + 0x0053

	7	6	5	4	3	2	1	0
R	PTIAD7	PTIAD6	PTIAD5	PTIAD4	PTIAD3	PTIAD2	PTIAD1	PTIAD0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	1	1	1	1	1	1	1	1

▨ = Reserved or Unimplemented

Figure 2-20. Port AD Input Register (PTIAD)

Read: Anytime. Write: Never; writes to these registers have no effect.

If the ATDDIEN1 bit of the associated I/O pin is set to 0 (digital input buffer is disabled), a read returns a 1. If the ATDDIEN1 bit of the associated I/O pin is set to 1 (digital input buffer is enabled), a read returns the status of the associated pin.

2.3.6.3 Port AD Data Direction Register (DDRAD)

Module Base + 0x0055

	7	6	5	4	3	2	1	0
R W	DDRAD7	DDRAD6	DDRAD5	DDRAD4	DDRAD3	DDRAD2	DDRAD1	DDRAD0
Reset	0	0	0	0	0	0	0	0

Figure 2-21. Port AD Data Direction Register (DDRAD)

Read: Anytime. Write: Anytime.

This register configures port pins PAD[7:0] as either input or output.

If a data direction bit is 0 (pin configured as input), then a read value on PTADx depends on the associated ATDDIEN1 bit. If the associated ATDDIEN1 bit is set to 1 (digital input buffer is enabled), a read on PTADx returns the value on port AD pin. If the associated ATDDIEN1 bit is set to 0 (digital input buffer is disabled), a read on PTADx returns a 1.

Table 2-15. DDRAD Field Descriptions

Field	Description
7:0 DDRAD[7:0]	**Data Direction Port AD** 0 Associated pin is configured as input. 1 Associated pin is configured as output.

2.3.6.4 Port AD Reduced Drive Register (RDRAD)

Module Base + 0x0057

	7	6	5	4	3	2	1	0
R W	RDRAD7	RDRAD6	RDRAD5	RDRAD4	RDRAD3	RDRAD2	RDRAD1	RDRAD0
Reset	0	0	0	0	0	0	0	0

Figure 2-22. Port AD Reduced Drive Register (RDRAD)

Read: Anytime. Write: Anytime.

This register configures the drive strength of configured output pins as either full or reduced. If a pin is configured as input, the corresponding Reduced Drive Register bit has no effect.

Table 2-16. RDRAD Field Descriptions

Field	Description
7:0 RDRAD[7:0]	**Reduced Drive Port A** 0 Full drive strength at output. 1 Associated pin drives at about 1/3 of the full drive strength.

2.3.6.5 Port AD Pull Device Enable Register (PERAD)

Module Base + 0x0059

	7	6	5	4	3	2	1	0
R W	PERAD7	PERAD6	PERAD5	PERAD4	PERAD3	PERAD2	PERAD1	PERAD0
Reset	0	0	0	0	0	0	0	0

Figure 2-23. Port AD Pull Device Enable Register (PERAD)

Read: Anytime. Write: Anytime.

This register configures whether a pull-up or a pull-down device is activated on configured input pins. If a pin is configured as output, the corresponding Pull Device Enable Register bit has no effect.

Table 2-17. PERAD Field Descriptions

Field	Description
7:0 PERAD[7:0]	**Pull Device Enable Port AD** 0 Pull-up or pull-down device is disabled. 1 Pull-up or pull-down device is enabled.

2.3.6.6 Port AD Polarity Select Register (PPSAD)

Module Base + 0x005B

	7	6	5	4	3	2	1	0
R W	PPSAD7	PPSAD6	PPSAD5	PPSAD4	PPSAD3	PPSAD2	PPSAD1	PPSAD0
Reset	0	0	0	0	0	0	0	0

Figure 2-24. Port AD Polarity Select Register (PPSAD)

Read: Anytime. Write: Anytime.

The Port AD Polarity Select Register serves a dual purpose by selecting the polarity of the active interrupt edge as well as selecting a pull-up or pull-down device if enabled (PERADx = 1). The Port AD Polarity Select Register is effective only when the corresponding Data Direction Register bit is set to 0 (input).

In pull-down mode (PPSADx = 1), a rising edge on a port AD pin sets the corresponding PIFADx bit. In pull-up mode (PPSADx = 0), a falling edge on a port AD pin sets the corresponding PIFADx bit.

Table 2-18. PPSAD Field Descriptions

Field	Description
7:0 PPSAD[7:0]	**Polarity Select Port AD** 0 A pull-up device is connected to the associated port AD pin, and detects falling edge for interrupt generation. 1 A pull-down device is connected to the associated port AD pin, and detects rising edge for interrupt generation.

2.3.6.7 Port AD Interrupt Enable Register (PIEAD)

Module Base + 0x005D

	7	6	5	4	3	2	1	0
R W	PIEAD7	PIEAD6	PIEAD5	PIEAD4	PIEAD3	PIEAD2	PIEAD1	PIEAD0
Reset	0	0	0	0	0	0	0	0

Figure 2-25. Port AD Interrupt Enable Register (PIEAD)

Read: Anytime. Write: Anytime.

This register disables or enables on a per pin basis the edge sensitive external interrupt associated with port AD.

Table 2-19. PIEAD Field Descriptions

Field	Description
7:0 PIEAD[7:0]	**Interrupt Enable Port AD** 0 Interrupt is disabled (interrupt flag masked). 1 Interrupt is enabled.

2.3.6.8 Port AD Interrupt Flag Register (PIFAD)

Module Base + 0x005F

	7	6	5	4	3	2	1	0
R W	PIFAD7	PIFAD6	PIFAD5	PIFAD4	PIFAD3	PIFAD2	PIFAD1	PIFAD0
Reset	0	0	0	0	0	0	0	0

Figure 2-26. Port AD Interrupt Flag Register (PIFAD)

Read: Anytime. Write: Anytime.

Each flag is set by an active edge on the associated input pin. The active edge could be rising or falling based on the state of the corresponding PPSADx bit. To clear each flag, write “1” to the corresponding PIFADx bit. Writing a “0” has no effect.

NOTE

If the ATDDIEN1 bit of the associated pin is set to 0 (digital input buffer is disabled), active edges can not be detected.

Table 2-20. PIFAD Field Descriptions

Field	Description
7:0 PIFAD[7:0]	**Interrupt Flags Port AD** 0 No active edge pending. Writing a “0” has no effect. 1 Active edge on the associated bit has occurred (an interrupt will occur if the associated enable bit is set). Writing a “1” clears the associated flag.

2.3.7 Port L

Port L is associated with the analog-to-digital converter (ATD) and the liquid crystal display (LCD) driver.

If the ATD module is enabled, the AN[15:8] inputs of ATD module are available on port L pins PL[7:0].

If the corresponding LCD frontplane drivers are enabled, the FP[31:28] and FP[19:16] outputs of LCD module are available on port L pins PL[7:0] and the general purpose I/Os are disabled.

For the pins of port L to be used as inputs, the corresponding LCD frontplane drivers must be disabled and the associated ATDDIEN0 register in the ATD module must be set to 1 (digital input buffer is enabled). The ATDDIEN0 register does not affect the port L pins when they are configured as outputs.

Refer to the LCD block description chapter for information on enabling and disabling the LCD and its frontplane drivers. Refer to the ATD block description chapter for information on the ATDDIEN0 register.

During reset, port L pins are configured as inputs with pull down.

2.3.7.1 Port L I/O Register (PTL)

Module Base + 0x0030

	7	6	5	4	3	2	1	0
R W	PTL7	PTL6	PTL5	PTL4	PTL3	PTL2	PTL1	PTL0
ATD:	AN15	AN14	AN13	AN12	AN11	AN10	AN9	AN8
LCD:	1	1	1	1	1	1	1	1
Reset	0	0	0	0	0	0	0	0

Figure 2-27. Port L I/O Register (PTL)

Read: Anytime. Write: Anytime.

If the data direction bit of the associated I/O pin (DDRLx) is set to 1 (output), a write to the corresponding I/O Register bit sets the value to be driven to the Port L pin. If the data direction bit of the associated I/O pin (DDRLx) is set to 0 (input), a write to the corresponding I/O Register bit takes place but has no effect on the Port L pin.

If the associated data direction bit (DDRLx) is set to 1 (output), a read returns the value of the I/O register bit.

If the associated data direction bit (DDRLx) is set to 0 (input) and the associated ATDDIEN0 bits is set to 0 (digital input buffer is disabled), the associated I/O register bit (PTLx) reads “1”.

If the associated data direction bit (DDRLx) is set to 0 (input), the associated ATDDIEN0 bit is set to 1 (digital input buffer is enabled), and the LCD frontplane driver is enabled (and LCD module is enabled), the associated I/O register bit (PTLx) reads “1”.

If the associated data direction bit (DDRLx) is set to 0 (input), the associated ATDDIEN0 bit is set to 1 (digital input buffer is enabled), and the LCD frontplane driver is disabled (or LCD module is disabled), a read returns the value of the pin.

2.3.7.2 Port L Input Register (PTIL)

Module Base + 0x0031

	7	6	5	4	3	2	1	0
R	PTIL7	PTIL6	PTIL5	PTIL4	PTIL3	PTIL2	PTIL1	PTIL0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	1	1	1	1	1	1	1	1

▨ = Reserved or Unimplemented

Figure 2-28. Port L Input Register (PTIL)

Read: Anytime. Write: Never, writes to this register have no effect.

If the LCD frontplane driver of an associated I/O pin is enabled (and LCD module is enabled) or the associated ATDDIEN0 bit is set to 0 (digital input buffer is disabled), a read returns a 1.

If the LCD frontplane driver of an associated I/O pin is disabled (or LCD module is disabled) and the associated ATDDIEN0 bit is set to 1 (digital input buffer is enabled), a read returns the status of the associated pin.

2.3.7.3 Port L Data Direction Register (DDRL)

Module Base + 0x0032

	7	6	5	4	3	2	1	0
R W	DDRL7	DDRL6	DDRL5	DDRL4	DDRL3	DDRL2	DDRL1	DDRL0
Reset	0	0	0	0	0	0	0	0

Figure 2-29. Port L Data Direction Register (DDRL)

Read: Anytime. Write: Anytime.

This register configures port pins PL[7:0] as either input or output.

If a LCD frontplane driver is enabled (and LCD module is enabled), it outputs an analog signal to the corresponding pin and the associated Data Direction Register bit has no effect. If a LCD frontplane driver is disabled (or LCD module is disabled), the corresponding Data Direction Register bit reverts to control the I/O direction of the associated pin.

Table 2-21. DDRL Field Descriptions

Field	Description
7:0 DDRL[7:0]	**Data Direction Port L** 0 Associated pin is configured as input. 1 Associated pin is configured as output.

2.3.7.4 Port L Reduced Drive Register (RDRL)

Module Base + 0x0033

	7	6	5	4	3	2	1	0
R W	RDRL7	RDRL6	RDRL5	RDRL4	RDRL3	RDRL2	RDRL1	RDRL0
Reset	0	0	0	0	0	0	0	0

Figure 2-30. Port L Reduced Drive Register (RDRL)

Read: Anytime. Write: Anytime.

This register configures the drive strength of configured output pins as either full or reduced. If a pin is configured as input, the corresponding Reduced Drive Register bit has no effect.

Table 2-22. RDRL Field Descriptions

Field	Description
7:0 RDRL[7:0]	**Reduced Drive Port L** 0 Full drive strength at output. 1 Associated pin drives at about 1/3 of the full drive strength.

2.3.7.5 Port L Pull Device Enable Register (PERL)

Module Base + 0x0034

	7	6	5	4	3	2	1	0
R W	PERL7	PERL6	PERL5	PERL4	PERL3	PERL2	PERL1	PERL0
Reset	1	1	1	1	1	1	1	1

Figure 2-31. Port L Pull Device Enable Register (PERL)

Read:Anytime. Write:Anytime.

This register configures whether a pull-up or a pull-down device is activated on configured input pins. If a pin is configured as output, the corresponding Pull Device Enable Register bit has no effect.

Table 2-23. PERL Field Descriptions

Field	Description
7:0 PERL[7:0]	**Pull Device Enable Port L** 0 Pull-up or pull-down device is disabled. 1 Pull-up or pull-down device is enabled.

2.3.7.6 Port L Polarity Select Register (PPSL)

Module Base + 0x0035

	7	6	5	4	3	2	1	0
R W	PPSL7	PPSL6	PPSL5	PPSL4	PPSL3	PPSL2	PPSL1	PPSL0
Reset	1	1	1	1	1	1	1	1

Figure 2-32. Port L Polarity Select Register (PPSL)

Read: Anytime. Write: Anytime.

The Port L Polarity Select Register selects whether a pull-down or a pull-up device is connected to the pin. The Port L Polarity Select Register is effective only when the corresponding Data Direction Register bit is set to 0 (input) and the corresponding Pull Device Enable Register bit is set to 1.

Table 2-24. PPSL Field Descriptions

Field	Description
7:0 PPSL[7:0]	**Pull Select Port L** 0 A pull-up device is connected to the associated port L pin. 1 A pull-down device is connected to the associated port L pin.

2.3.7.7 Port L Slew Rate Register (SRRL)

Module Base + 0x003B

	7	6	5	4	3	2	1	0
R W	SRRL7	SRRL6	SRRL5	SRRL4	SRRL3	SRRL2	SRRL1	SRRL0
Reset	0	0	0	0	0	0	0	0

Figure 2-33. Port L Slew Rate Register (SRRL)

Read: Anytime. Write: Anytime.

This register enables the slew rate control and disables the digital input buffer for port pins PL[7:0].

Table 2-25. SRRL Field Descriptions

Field	Description
7:0 SRRL[7:0]	**Slew Rate Port L** 0 Disables slew rate control and enables digital input buffer. 1 Enables slew rate control and disables digital input buffer.

2.3.8 Port M

Port M is associated with the chip select 1 and the Freescale's scalable controller area network (CAN1 and CAN0) modules. Each pin is assigned to these modules according to the following priority: $\overline{\text{CS1}}$/CAN1/CAN0 > general-purpose I/O.

When the CAN1 module is enabled, PM[5:4] pins become TXCAN1 (transmitter) and RXCAN1 (receiver) pins for the CAN1 module. When the CAN0 module is enabled, PM[3:2] pins become TXCAN0 (transmitter) and RXCAN0 (receiver) pins for the CAN0 module. Refer to the MSCAN block description chapter for information on enabling and disabling the CAN module.

During reset, port M pins are configured as high-impedance inputs.

2.3.8.1 Port M I/O Register (PTM)

Module Base + 0x0010

	7	6	5	4	3	2	1	0
R	0	0	PTM5	PTM4	PTM3	PTM2	PTM1	0
W	▨	▨						▨
CAN0/CAN1:			TXCAN1	RXCAN1	TXCAN0	RXCAN0		
Chip Select:							$\overline{\text{CS1}}$	
Reset	0	0	0	0	0	0	0	0

▨ = Reserved or Unimplemented

Figure 2-34. Port M I/O Register (PTM)

Read: Anytime. Write: Anytime.

If the associated data direction bit (DDRMx) is set to 1 (output), a read returns the value of the I/O register bit. If the associated data direction bit (DDRMx) is set to 0 (input), a read returns the value of the pin.

2.3.8.2 Port M Input Register (PTIM)

Module Base + 0x0011

	7	6	5	4	3	2	1	0
R	0	0	PTIM5	PTIM4	PTIM3	PTIM2	PTIM1	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	u	u	u	u	u	0

▨ = Reserved or Unimplemented

u = Unaffected by reset

Figure 2-35. Port M Input Register (PTIM)

Read: Anytime. Write: Never, writes to this register have no effect.

This register always reads back the status of the associated pins.

2.3.8.3 Port M Data Direction Register (DDRM)

Module Base + 0x0012

	7	6	5	4	3	2	1	0
R	0	0	DDRM5	DDRM4	DDRM3	DDRM2	DDRM1	0
W	▨	▨						▨
Reset	0	0	0	0	0	0	0	0

▨ = Reserved or Unimplemented

Figure 2-36. Port M Data Direction Register (DDRM)

Read: Anytime. Write: Anytime.

This register configures port pins PM[5:1] as either input or output.

When a CAN module is enabled, the corresponding transmitter (TXCANx) pin becomes an output, the corresponding receiver (RXCANx) pin becomes an input, and the associated Data Direction Register bits have no effect. If a CAN module is disabled, the corresponding Data Direction Register bit reverts to control the I/O direction of the associated pin.

Table 2-26. DDRM Field Descriptions

Field	Description
5:1 DDRM[5:1]	**Data Direction Port M** 0 Associated pin is configured as input. 1 Associated pin is configured as output.

2.3.8.4 Port M Reduced Drive Register (RDRM)

Module Base + 0x0013

	7	6	5	4	3	2	1	0
R	0	0	RDRM5	RDRM4	RDRM3	RDRM2	RDRM1	0
W	▨	▨						▨
Reset	0	0	0	0	0	0	0	0

▨ = Reserved or Unimplemented

Figure 2-37. Port M Reduced Drive Register (RDRM)

Read: Anytime. Write: Anytime.

This register configures the drive strength of configured output pins as either full or reduced. If a pin is configured as input, the corresponding Reduced Drive Register bit has no effect.

Table 2-27. RDRM Field Descriptions

Field	Description
5:1 RDRM[5:1]	**Reduced Drive Port M** 0 Full drive strength at output 1 Associated pin drives at about 1/3 of the full drive strength.

2.3.8.5 Port M Pull Device Enable Register (PERM)

Module Base + 0x0014

	7	6	5	4	3	2	1	0
R	0	0	PERM5	PERM4	PERM3	PERM2	PERM1	0
W	▨	▨						▨
Reset	0	0	0	0	0	0	0	0

▨ = Reserved or Unimplemented

Figure 2-38. Port M Pull Device Enable Register (PERM)

Read: Anytime. Write: Anytime.

This register configures whether a pull-up or a pull-down device is activated on configured input or wired-or output pins. If a pin is configured as push-pull output, the corresponding Pull Device Enable Register bit has no effect.

Table 2-28. PERM Field Descriptions

Field	Description
5:1 PERM[5:1]	**Pull Device Enable Port M** 0 Pull-up or pull-down device is disabled. 1 Pull-up or pull-down device is enabled.

2.3.8.6 Port M Polarity Select Register (PPSM)

Module Base + 0x0015

	7	6	5	4	3	2	1	0
R	0	0	PPSM5	PPSM4	PPSM3	PPSM2	PPSM1	0
W	▨	▨						▨
Reset	0	0	0	0	0	0	0	0

▨ = Reserved or Unimplemented

Figure 2-39. Port M Polarity Select Register (PPSM)

Read: Anytime. Write: Anytime.

The Port M Polarity Select Register selects whether a pull-down or a pull-up device is connected to the pin. The Port M Polarity Select Register is effective only when the corresponding Data Direction Register bit is set to 0 (input) and the corresponding Pull Device Enable Register bit is set to 1.

If a CAN module is enabled, a pull-up device can be activated on the receiver pin, and on the transmitter pin if the corresponding wired-OR mode bit is set. Pull-down devices can not be activated on CAN pins.

Table 2-29. PPSM Field Descriptions

Field	Description
5:1 PPSM[5:1]	**Pull Select Port M** 0 A pull-up device is connected to the associated port M pin. 1 A pull-down device is connected to the associated port M pin.

2.3.8.7 Port M Wired-OR Mode Register (WOMM)

Module Base + 0x0016

	7	6	5	4	3	2	1	0
R	0	0	WOMM5	WOMM4	WOMM3	WOMM2	WOMM1	0
W	▨	▨						▨
Reset	0	0	0	0	0	0	0	0

▨ = Reserved or Unimplemented

Figure 2-40. Port M Wired-OR Mode Register (WOMM)

Read: Anytime. Write: Anytime.

This register selects whether a port M output is configured as push-pull or wired-or. When a Wired-OR Mode Register bit is set to 1, the corresponding output pin is driven active low only (open drain) and a high level is not driven. A Wired-OR Mode Register bit has no effect if the corresponding pin is configured as an input.

These bits apply also to the CAN transmitter and allow a multipoint connection of several serial modules.

Table 2-30. WOMM Field Descriptions

Field	Description
5:1 WOMM[5:1]	**Wired-OR Mode Port M** 0 Output buffers operate as push-pull outputs. 1 Output buffers operate as open-drain outputs.

2.3.8.8 Port M Slew Rate Register (SRRM)

Module Base + 0x003B

	7	6	5	4	3	2	1	0
R	0	0	SRRM5	SRRM4	SRRM3	SRRM2	SRRM1	0
W	▨	▨						▨
Reset	0	0	0	0	0	0	0	0

▨ = Reserved or Unimplemented

Figure 2-41. Port M Slew Rate Register (SRRM)

Read: Anytime. Write: Anytime.

This register enables the slew rate control and disables the digital input buffer for port pins PM[5:1].

Table 2-31. SRRM Field Descriptions

Field	Description
5:1 SRRM[5:1]	**Slew Rate Port M** 0 Disables slew rate control and enables digital input buffer. 1 Enables slew rate control and disables digital input buffer.

2.3.9 Port P

Port P is associated with the chip selects 0 and 2, the Pulse Width Modulator (PWM), the serial communication interface (SCI1) and the Inter-IC bus (IIC0 and IIC1) modules. Each pin is assigned to these modules according to the following priority: $\overline{\text{CS0}}$/$\overline{\text{CS2}}$ > PWM > SCI1/IIC1/IIC0 > general-purpose I/O.

When a PWM channel is enabled, the corresponding pin becomes a PWM output with the exception of PP[5] which can be PWM input or output. Refer to the PWM block description chapter for information on enabling and disabling the PWM channels.

When the IIC1 module is enabled and MODRR1 is clear, PP[7:6] pins become SCL1 and SDA1 respectively as long as the corresponding PWM channels are disabled. When the IIC0 module is enabled and MODRR0 is clear, PP[5:4] pins become SCL0 and SDA0 respectively as long as the corresponding PWM channels are disabled. Refer to the IIC block description chapter for information on enabling and disabling the IIC bus.

When the SCI1 receiver and transmitter are enabled and MODRR2 is clear, the PP[2] and PP[0] pins become RXD1 and TXD1 respectively as long as the corresponding PWM channels are disabled. Refer to the SCI block description chapter for information on enabling and disabling the SCI receiver and transmitter.

During reset, port P pins are configured as high-impedance inputs.

2.3.9.1 Port P I/O Register (PTP)

Module Base + 0x0018

	7	6	5	4	3	2	1	0
R W	PTP7	PTP6	PTP5	PTP4	PTP3	PTP2	PTP1	PTP0
SCI1/ IIC1/IIC0:	SCL1	SDA1	SCL0	SDA0		RXD1		TXD1
PWM:	PWM7	PWM6	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0
Chip Select:	$\overline{\text{CS2}}$	$\overline{\text{CS0}}$						
Reset	0	0	0	0	0	0	0	0

Figure 2-42. Port P I/O Register (PTP)

Read: Anytime. Write: Anytime.

If the associated data direction bit (DDRPx) is set to 1 (output), a read returns the value of the I/O register bit. If the associated data direction bit (DDRPx) is set to 0 (input), a read returns the value of the pin.

The PWM function takes precedence over the general-purpose I/O function if the associated PWM channel is enabled. The PWM channels 6-0 are outputs if the respective channels are enabled. PWM channel 7 can be an output, or an input if the shutdown feature is enabled.

The IIC function takes precedence over the general-purpose I/O function if the IIC bus is enabled and the corresponding PWM channels remain disabled. The SDA and SCL pins are bidirectional with outputs configured as open-drain.

If enabled, the SCI1 transmitter takes precedence over the general-purpose I/O function, and the corresponding TXD1 pin is configured as an output. If enabled, the SCI1 receiver takes precedence over the general-purpose I/O function, and the corresponding RXD1 pin is configured as an input.

2.3.9.2 Port P Input Register (PTIP)

Module Base + 0x0019

	7	6	5	4	3	2	1	0
R	PTIP7	PTIP6	PTIP5	PTIP4	PTIP3	PTIP2	PTIP1	PTIP0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	u	u	u	u	u	u	u	u

▨ = Reserved or Unimplemented u = Unaffected by reset

Figure 2-43. Port P I/O Register (PTP)

Read: Anytime. Write: Never, writes to this register have no effect.

This register always reads back the status of the associated pins.

2.3.9.3 Port P Data Direction Register (DDRP)

Module Base + 0x001A

	7	6	5	4	3	2	1	0
R W	DDRP7	DDRP6	DDRP5	DDRP4	DDRP3	DDRP2	DDRP1	DDRP0
Reset	0	0	0	0	0	0	0	0

Figure 2-44. Port P Data Direction Register (DDRP)

Read: Anytime. Write: Anytime.

This register configures port pins PP[7:0] as either input or output.

If a PWM channel is enabled, the corresponding pin is forced to be an output and the associated Data Direction Register bit has no effect. Channel 5 can also force the corresponding pin to be an input if the shutdown feature is enabled.

When an IIC bus is enabled, the corresponding pins become the SCL and SDA bidirectional pins respectively as long as the corresponding PWM channels are disabled. The associated Data Direction Register bits have no effect.

When the SCI1 transmitter is enabled, the PP[0] pin becomes the TXD1 output pin and the associated Data Direction Register bit has no effect. When the SCI1 receiver is enabled, the PP[2] pin becomes the RXD1 input pin and the associated Data Direction Register bit has no effect.

If the PWM, IIC0, IIC1 and SCI1 functions are disabled, the corresponding Data Direction Register bit reverts to control the I/O direction of the associated pin.

Table 2-32. DDRP Field Descriptions

Field	Description
7:0 DDRP[7:0]	**Data Direction Port P** 0 Associated pin is configured as input. 1 Associated pin is configured as output.

2.3.9.4 Port P Reduced Drive Register (RDRP)

Module Base + 0x001B

	7	6	5	4	3	2	1	0
R W	RDRP7	RDRP6	RDRP5	RDRP4	RDRP3	RDRP2	RDRP1	RDRP0
Reset	0	0	0	0	0	0	0	0

Figure 2-45. Port P Reduced Drive Register (RDRP)

Read:Anytime. Write:Anytime.

This register configures the drive strength of configured output pins as either full or reduced. If a pin is configured as input, the corresponding Reduced Drive Register bit has no effect.

Table 2-33. RDRP Field Descriptions

Field	Description
7:0 RDRP[7:0]	**Reduced Drive Port P** 0 Full drive strength at output. 1 Associated pin drives at about 1/3 of the full drive strength.

2.3.9.5 Port P Pull Device Enable Register (PERP)

Module Base + 0x001C

	7	6	5	4	3	2	1	0
R W	PERP7	PERP6	PERP5	PERP4	PERP3	PERP2	PERP1	PERP0
Reset	0	0	0	0	0	0	0	0

Figure 2-46. Port P Pull Device Enable Register (PERP)

Read: Anytime. Write: Anytime.

This register configures whether a pull-up or a pull-down device is activated on configured input or wired-or (open drain) output pins. If a pin is configured as push-pull output, the corresponding Pull Device Enable Register bit has no effect.

Table 2-34. PERP Field Descriptions

Field	Description
7:0 PERP[7:0]	**Pull Device Enable Port P** 0 Pull-up or pull-down device is disabled. 1 Pull-up or pull-down device is enabled.

2.3.9.6 Port P Polarity Select Register (PPSP)

Module Base + 0x001D

	7	6	5	4	3	2	1	0
R W	PPSP7	PPSP6	PPSP5	PPSP4	PPSP3	PPSP2	PPSP1	PPSP0
Reset	0	0	0	0	0	0	0	0

Figure 2-47. Port P Polarity Select Register (PPSP)

Read: Anytime. Write: Anytime.

The Port P Polarity Select Register selects whether a pull-down or a pull-up device is connected to the pin. The Port P Polarity Select Register is effective only when the corresponding Data Direction Register bit is set to 0 (input) and the corresponding Pull Device Enable Register bit is set to 1.

If an IIC module is enabled, a pull-up device can be activated on either the SCL or SDA pins. Pull-down devices can not be activated on IIC pins.

Table 2-35. PPSP Field Descriptions

Field	Description
7:0 PPSP[7:0]	**Polarity Select Port P** 0 A pull-up device is connected to the associated port P pin. 1 A pull-down device is connected to the associated port P pin.

2.3.9.7 Port P Wired-OR Mode Register (WOMP)

Module Base + 0x001E

	7	6	5	4	3	2	1	0
R	WOMP7	WOMP6	WOMP5	WOMP4	0	WOMP2	0	WOMPO
W					▨		▨	
Reset	0	0	0	0	0	0	0	0

▨ = Reserved or Unimplemented

Figure 2-48. Port P Wired-OR Mode Register (WOMP)

Read: Anytime. Write: Anytime.

This register selects whether a port P output is configured as push-pull or wired-or. When a Wired-OR Mode Register bit is set to 1, the corresponding output pin is driven active low only (open drain) and a high level is not driven. A Wired-OR Mode Register bit has no effect if the corresponding pin is configured as an input.

If IIC is enabled and the corresponding PWM channels are disabled, the pins are configured as wired-or and the corresponding Wired-OR Mode Register bits have no effect.

Table 2-36. WOMP Field Descriptions

Field	Description
7:4 WOMP[7:4]	**Wired-OR Mode Port P** 0 Output buffers operate as push-pull outputs. 1 Output buffers operate as open-drain outputs.
2 WOMP2	**Wired-OR Mode Port P** 0 Output buffers operate as push-pull outputs. 1 Output buffers operate as open-drain outputs.
0 WOMP0	**Wired-OR Mode Port P** 0 Output buffers operate as push-pull outputs. 1 Output buffers operate as open-drain outputs.

2.3.9.8 Port P Slew Rate Register (SRRP)

Module Base + 0x003B

	7	6	5	4	3	2	1	0
R W	SRRP7	SRRP6	SRRP5	SRRP4	SRRP3	SRRP2	SRRP1	SRRP0
Reset	0	0	0	0	0	0	0	0

Figure 2-49. Port P Slew Rate Register (SRRP)

Read: Anytime. Write: Anytime.

This register enables the slew rate control and disables the digital input buffer for port pins PP[7:0].

Table 2-37. SRRP Field Descriptions

Field	Description
7:0 SRRP[7:0]	**Slew Rate Port P** 0 Disables slew rate control and enables digital input buffer. 1 Enables slew rate control and disables digital input buffer.

2.3.10 Port S

Port S is associated with the chip select 3, the serial peripheral interface (SPI) and the serial communication interface (SCI0). Each pin is assigned to these modules according to the following priority: $\overline{CS3}$ > SPI/SCI1/SCI0 > general-purpose I/O.

When the SPI is enabled, the PS[7:4] pins become $\overline{SS}$, SCK, MOSI, and MISO respectively. Refer to the SPI block description chapter for information on enabling and disabling the SPI.

When the SCI0 receiver and transmitter are enabled, the PS[1:0] pins become TXD0 and RXD0 respectively. When the SCI1 receiver and transmitter are enabled and MODRR2 is set, the PS[3:2] pins become TXD1 and RXD1 respectively. Refer to the SCI block description chapter for information on enabling and disabling the SCI receiver and transmitter.

During reset, port S pins are configured as high-impedance inputs.

2.3.10.1 Port S I/O Register (PTS)

Module Base + 0x0008

	7	6	5	4	3	2	1	0
R W	PTS7	PTS6	PTS5	PTS4	PTS3	PTS2	PTS1	PTS0
SPI/ SCI1/SCI0:	$\overline{SS}$	SCK	MOSI	MISO	TXD1	RXD1	TXD0	RXD0
Chip Select:						$\overline{CS3}$		
Reset	0	0	0	0	0	0	0	0

▨ = Reserved or Unimplemented

Figure 2-50. Port S I/O Register (PTS)

Read: Anytime. Write: Anytime.

If the associated data direction bit (DDRSx) is set to 1 (output), a read returns the value of the I/O register bit. If the associated data direction bit (DDRSx) is set to 0 (input), a read returns the value of the pin.

The SPI function takes precedence over the general-purpose I/O function if the SPI is enabled.

If enabled, the SCI0(1) transmitter takes precedence over the general-purpose I/O function, and the corresponding TXD0(1) pin is configured as an output. If enabled, the SCI0(1) receiver takes precedence over the general-purpose I/O function, and the corresponding RXD0(1) pin is configured as an input.

2.3.10.2 Port S Input Register (PTIS)

Module Base + 0x0009

	7	6	5	4	3	2	1	0
R	PTIS7	PTIS6	PTIS5	PTIS4	PTIS3	PTIS2	PTIS1	PTIS0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	u	u	u	u	u	u	u	u

▨ = Reserved or Unimplemented

u = Unaffected by reset

Figure 2-51. Port S Input Register (PTIS)

Read: Anytime. Write: Never, writes to this register have no effect.

This register always reads back the status of the associated pins.

2.3.10.3 Port S Data Direction Register (DDRS)

Module Base + 0x000A

	7	6	5	4	3	2	1	0
R W	DDRS7	DDRS6	DDRS5	DDRS4	DDRS3	DDRS2	DDRS1	DDRS0
Reset	0	0	0	0	0	0	0	0

Figure 2-52. Port S Data Direction Register (DDRS)

Read: Anytime. Write: Anytime.

This register configures port pins PS[7:0] as either input or output.

When the SPI is enabled, the PS[7:4] pins become the SPI bidirectional pins. The associated Data Direction Register bits have no effect.

When the SCI1 transmitter is enabled, the PS[3] pin becomes the TXD1 output pin and the associated Data Direction Register bit has no effect. When the SCI1 receiver is enabled, the PS[2] pin becomes the RXD1 input pin and the associated Data Direction Register bit has no effect.

When the SCI0 transmitter is enabled, the PS[1] pin becomes the TXD0 output pin and the associated Data Direction Register bit has no effect. When the SCI0 receiver is enabled, the PS[0] pin becomes the RXD0 input pin and the associated Data Direction Register bit has no effect.

If the SPI, SCI1 and SCI0 functions are disabled, the corresponding Data Direction Register bit reverts to control the I/O direction of the associated pin.

Table 2-38. DDRS Field Descriptions

Field	Description
7:0 DDRS[7:0]	**Data Direction Port S** 0 Associated pin is configured as input. 1 Associated pin is configured as output.

2.3.10.4 Port S Reduced Drive Register (RDRS)

Module Base + 0x000B

	7	6	5	4	3	2	1	0
R W	RDRS7	RDRS6	RDRS5	RDRS4	RDRS3	RDRS2	RDRS1	RDRS0
Reset	0	0	0	0	0	0	0	0

Figure 2-53. Port S Reduced Drive Register (RDRS)

Read: Anytime. Write: Anytime.

This register configures the drive strength of configured output pins as either full or reduced. If a pin is configured as input, the corresponding Reduced Drive Register bit has no effect.

Table 2-39. RDRS Field Descriptions

Field	Description
7:0 RDRS[7:0]	**Reduced Drive Port S** 0 Full drive strength at output. 1 Associated pin drives at about 1/3 of the full drive strength.

2.3.10.5 Port S Pull Device Enable Register (PERS)

Module Base + 0x000C

	7	6	5	4	3	2	1	0
R W	PERS7	PERS6	PERS5	PERS4	PERS3	PERS2	PERS1	PERS0
Reset	0	0	0	0	0	0	0	0

Figure 2-54. Port S Pull Device Enable Register (PERS)

Read: Anytime. Write: Anytime.

This register configures whether a pull-up or a pull-down device is activated on configured input or wired-or (open drain) output pins. If a pin is configured as push-pull output, the corresponding Pull Device Enable Register bit has no effect.

Table 2-40. PERS Field Descriptions

Field	Description
7:0 PERS[7:0]	**Pull Device Enable Port S** 0 Pull-up or pull-down device is disabled. 1 Pull-up or pull-down device is enabled.

2.3.10.6 Port S Polarity Select Register (PPSS)

Module Base + 0x000D

	7	6	5	4	3	2	1	0
R W	PPSS7	PPSS6	PPSS5	PPSS4	PPSS3	PPSS2	PPSS1	PPSS0
Reset	0	0	0	0	0	0	0	0

Figure 2-55. Port S Polarity Select Register (PPSS)

Read: Anytime. Write: Anytime.

The Port S Polarity Select Register selects whether a pull-down or a pull-up device is connected to the pin. The Port S Polarity Select Register is effective only when the corresponding Data Direction Register bit is set to 0 (input) and the corresponding Pull Device Enable Register bit is set to 1.

Table 2-41. PPSS Field Descriptions

Field	Description
7:0 PPSS[7:0]	**Pull Select Port S** 0 A pull-up device is connected to the associated port S pin. 1 A pull-down device is connected to the associated port S pin.

2.3.10.7 Port S Wired-OR Mode Register (WOMS)

Module Base + 0x000E

	7	6	5	4	3	2	1	0
R W	WOMS7	WOMS6	WOMS5	WOMS4	WOMS3	WOMS2	WOMS1	WOMS0
Reset	0	0	0	0	0	0	0	0

Figure 2-56. Port S Wired-OR Mode Register (WOMS)

Read: Anytime. Write: Anytime.

This register selects whether a port S output is configured as push-pull or wired-or. When a Wired-OR Mode Register bit is set to 1, the corresponding output pin is driven active low only (open drain) and a high level is not driven. A Wired-OR Mode Register bit has no effect if the corresponding pin is configured as an input.

Table 2-42. WOMS Field Descriptions

Field	Description
7:0 WOMS[7:0]	**Wired-OR Mode Port S** 0 Output buffers operate as push-pull outputs. 1 Output buffers operate as open-drain outputs.

2.3.10.8 Port S Slew Rate Register (SRRS)

Module Base + 0x003B

	7	6	5	4	3	2	1	0
R W	SRRS7	SRRS6	SRRS5	SRRS4	SRRS3	SRRS2	SRRS1	SRRS0
Reset	0	0	0	0	0	0	0	0

Figure 2-57. Port S Slew Rate Register (SRRS)

Read: Anytime. Write: Anytime.

This register enables the slew rate control and disables the digital input buffer for port pins PS[7:0].

Table 2-43. SRRS Field Descriptions

Field	Description
7:0 SRRS[7:0]	**Slew Rate Port S** 0 Disables slew rate control and enables digital input buffer. 1 Enables slew rate control and disables digital input buffer.

2.3.11 Port T

Port T is associated with the 8-channel enhanced capture timer (ECT), the Inter-IC (IIC0 and IIC1) modules and the liquid crystal display (LCD) driver. Each pin is assigned to these modules according to the following priority: LCD Driver > IIC1/IIC0 > ECT > general-purpose I/O.

When the IIC1 module is enabled and MODRR1 is set, PT[7:6] pins become SCL1 and SDA1 pins respectively. When the IIC0 module is enabled and MODRR0 is set, PT[5:4] pins become SCL0 and SDA0 respectively. Refer to the IIC block description chapter for information on enabling and disabling the IIC bus.

If the corresponding LCD frontplane drivers are enabled (and LCD module is enabled), the FP[27:24] outputs of the LCD module are available on port T pins PT[3:0].

If the corresponding LCD frontplane drivers are disabled (or LCD module is disabled) and the ECT is enabled, the timer channels configured for output compare are available on port T pins PT[3:0].

Refer to the LCD block description chapter for information on enabling and disabling the LCD and its frontplane drivers.Refer to the ECT block description chapter for information on enabling and disabling the ECT module.

During reset, port T pins are configured as inputs with pull down.

2.3.11.1 Port T I/O Register (PTT)

Module Base + 0x0000

	7	6	5	4	3	2	1	0
R W	PTT7	PTT6	PTT5	PTT4	PTT3	PTT2	PTT1	PTT0
ECT:	OC7	OC6	OC5	OC4	OC3	OC2	OC1	OC0
IIC1/IIC0:	SCL1	SDA1	SCL0	SDA0				
LCD:					1	1	1	1
Reset	0	0	0	0	0	0	0	0

Figure 2-58. Port T I/O Register (PTT)

Read: Anytime. Write: Anytime.

If the associated data direction bit (DDRTx) is set to 1 (output), a read returns the value of the I/O register bit.

If the associated data direction bit (DDRTx) is set to 0 (input) and the LCD frontplane driver is enabled (and LCD module is enabled), the associated I/O register bit (PTTx) reads “1”.

If the associated data direction bit (DDRTx) is set to 0 (input) and the LCD frontplane driver is disabled (or LCD module is disabled), a read returns the value of the pin.

2.3.11.2 Port T Input Register (PTIT)

Module Base + 0x0001

	7	6	5	4	3	2	1	0
R	PTIT7	PTIT6	PTIT5	PTIT4	PTIT3	PTIT2	PTIT1	PTIT0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	u	u	u	u	u	u	u	u

▨ = Reserved or Unimplemented u = Unaffected by reset

Figure 2-59. Port T Input Register (PTIT)

Read: Anytime. Write: Never, writes to this register have no effect.

If the LCD frontplane driver of an associated I/O pin is enabled (and LCD module is enabled), a read returns a 1.

If the LCD frontplane driver of the associated I/O pin is disabled (or LCD module is disabled), a read returns the status of the associated pin.

2.3.11.3 Port T Data Direction Register (DDRT)

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R W	DDRT7	DDRT6	DDRT5	DDRT4	DDRT3	DDRT2	DDRT1	DDRT0
Reset	0	0	0	0	0	0	0	0

Figure 2-60. Port T Data Direction Register (DDRT)

Read: Anytime. Write: Anytime.

This register configures port pins PT[7:0] as either input or output.

If a LCD frontplane driver is enabled (and LCD module is enabled), it outputs an analog signal to the corresponding pin and the associated Data Direction Register bit has no effect. If a LCD frontplane driver is disabled (or LCD module is disabled), the corresponding Data Direction Register bit reverts to control the I/O direction of the associated pin.

If the ECT module is enabled, each port pin configured for output compare is forced to be an output and the associated Data Direction Register bit has no effect. If the associated timer output compare is disabled, the corresponding Data Direction Register bit reverts to control the I/O direction of the associated pin.

If the ECT module is enabled, each port pin configured as an input capture has the corresponding Data Direction Register bit controlling the I/O direction of the associated pin.

Table 2-44. DDRT Field Descriptions

Field	Description
7:0 DDRT[7:0]	**Data Direction Port T** 0 Associated pin is configured as input. 1 Associated pin is configured as output.

2.3.11.4 Port T Reduced Drive Register (RDRT)

Module Base + 0x0003

	7	6	5	4	3	2	1	0
R W	RDRT7	RDRT6	RDRT5	RDRT4	RDRT3	RDRT2	RDRT1	RDRT0
Reset	0	0	0	0	0	0	0	0

Figure 2-61. Port T Reduced Drive Register (RDRT)

Read: Anytime. Write: Anytime.

This register configures the drive strength of configured output pins as either full or reduced. If a pin is configured as input, the corresponding Reduced Drive Register bit has no effect.

Table 2-45. RDRT Field Descriptions

Field	Description
7:0 RDRT[7:0]	**Reduced Drive Port T** 0 Full drive strength at output. 1 Associated pin drives at about 1/3 of the full drive strength.

2.3.11.5 Port T Pull Device Enable Register (PERT)

Module Base + 0x0004

	7	6	5	4	3	2	1	0
R W	PERT7	PERT6	PERT5	PERT4	PERT3	PERT2	PERT1	PERT0
Reset	0	0	0	0	1	1	1	1

Figure 2-62. Port T Pull Device Enable Register (PERT)

Read: Anytime. Write: Anytime.

This register configures whether a pull-up or a pull-down device is activated on configured input pins. If a pin is configured as output, the corresponding Pull Device Enable Register bit has no effect.

For port pins PT[7:4], a pull-up device can be activated on wired-or (open drain) output pins. If the pin is configured as push-pull output, the corresponding Pull Device Enable Register bit has no effect.

Table 2-46. PERT Field Descriptions

Field	Description
7:0 PERT[7:0]	**Pull Device Enable Port T** 0 Pull-up or pull-down device is disabled. 1 Pull-up or pull-down device is enabled.

2.3.11.6 Port T Polarity Select Register (PPST)

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R W	PPST7	PPST6	PPST5	PPST4	PPST3	PPST2	PPST1	PPST0
Reset	0	0	0	0	1	1	1	1

Figure 2-63. Port T Polarity Select Register (PPST)

Read: Anytime. Write: Anytime.

The Port T Polarity Select Register selects whether a pull-down or a pull-up device is connected to the pin. The Port T Polarity Select Register is effective only when the corresponding Data Direction Register bit is set to 0 (input) and the corresponding Pull Device Enable Register bit is set to 1.

If an IIC module is enabled, a pull-up device can be activated on either the SCL or SDA pins. Pull-down devices can not be activated on IIC pins.

Table 2-47. PPST Field Descriptions

Field	Description
7:0 PPST[7:0]	**Pull Select Port T** 0 A pull-up device is connected to the associated port T pin. 1 A pull-down device is connected to the associated port T pin.

2.3.11.7 Port T Wired-OR Mode Register (WOMT)

Module Base + 0x003B

	7	6	5	4	3	2	1	0
R	WOMT7	WOMT6	WOMT5	WOMT4	0	MODRR2	MODRR1	MODRR0
W					▨			
Reset	0	0	0	0	0	0	0	0

▨ = Reserved or Unimplemented

Figure 2-64. Port T Wired-OR Mode Register (WOMT)

Read: Anytime. Write: Anytime.

This register selects whether a port T output is configured as push-pull or wired-or. When a Wired-OR Mode Register bit is set to 1, the corresponding output pin is driven active low only (open drain) and a high level is not driven. A Wired-OR Mode Register bit has no effect if the corresponding pin is configured as an input.

If IIC is enabled, the pins are configured as wired-or and the corresponding Wired-OR Mode Register bits have no effect.

This register also configures the re-routing of IIC0, IIC1 and SCI1 on alternative ports.

Table 2-48. WOMT Field Descriptions

Field	Description
7:4 WOMT[7:4]	**Wired-OR Mode Port T** 0 Output buffers operate as push-pull outputs. 1 Output buffers operate as open-drain outputs.
2 MODRR2	**SCI1 Routing Bit** — See Table 2-49..
1 MODRR1	**IIC1 Routing Bit** — See Table 2-50..
0 MODRR0	**IIC0 Routing Bit** — See Table 2-51..

Table 2-49. SCI1 Routing

MODRR[2]	TXD1	RXD1
0	PP0	PP2

Table 2-49. SCI1 Routing

MODRR[2]	TXD1	RXD1
1	PS3	PS2

Table 2-50. IIC1 Routing

MODRR[1]	SDA1	SCL1
0	PP6	PP7
1	PT6	PT7

Table 2-51. IIC0 Routing

MODRR[0]	SDA0	SCL0
0	PP4	PP5
1	PT4	PT5

2.3.11.8 Port T Slew Rate Register (SRRT)

Module Base + 0x003B

	7	6	5	4	3	2	1	0
R W	SRRT7	SRRT6	SRRT5	SRRT4	SRRT3	SRRT2	SRRT1	SRRT0
Reset	0	0	0	0	0	0	0	0

Figure 2-65. Port T Slew Rate Register (SRRT)

Read: Anytime. Write: Anytime.

This register enables the slew rate control and disables the digital input buffer for port pins PT[7:0].

Table 2-52. SRRT Field Descriptions

Field	Description
7:0 SRRT[7:0]	**Slew Rate Port T** 0 Disables slew rate control and enables digital input buffer. 1 Enables slew rate control and disables digital input buffer.

2.3.12 Port U

Port U is associated with the stepper stall detect (SSD1 and SSD0) and motor controller (MC1 and MC0) modules. Each pin is assigned to these modules according to the following priority: SSD1/SSD0 > MC1/MC0 > general-purpose I/O.

If SSD1 module is enabled, the PU[7:4] pins are controlled by the SSD1 module. If SSD1 module is disabled, the PU[7:4] pins are controlled by the motor control PWM channels 3 and 2 (MC1).

If SSD0 module is enabled, the PU[3:0] pins are controlled by the SSD0 module. If SSD0 module is disabled, the PU[3:0] pins are controlled by the motor control PWM channels 1 and 0 (MC0).

Refer to the SSD and MC block description chapters for information on enabling and disabling the SSD module and the motor control PWM channels respectively.

During reset, port U pins are configured as high-impedance inputs.

2.3.12.1 Port U I/O Register (PTU)

Module Base + 0x0038

	7	6	5	4	3	2	1	0
R W	PTU7	PTU6	PTU5	PTU4	PTU3	PTU2	PTU1	PTU0
MC:	M1C1P	M1C1M	M1COP	M1COM	M0C1P	M0C1M	M0C0P	M0C0M
SSD1/ SSD0:	M1SINP	M1SINM	M1COSP	M1COSM	M0SINP	M0SINM	M1COSP	M0COSM
Reset	0	0	0	0	0	0	0	0

Figure 2-66. Port U I/O Register (PTU)

Read: Anytime. Write: Anytime.

If the associated data direction bit (DDRUx) is set to 1 (output), a read returns the value of the I/O register bit.

If the associated data direction bit (DDRUx) is set to 0 (input) and the slew rate is enabled, the associated I/O register bit (PTUx) reads “1”.

If the associated data direction bit (DDRUx) is set to 0 (input) and the slew rate is disabled, a read returns the value of the pin.

2.3.12.2 Port U Input Register (PTIU)

Module Base + 0x0039

	7	6	5	4	3	2	1	0
R	PTIU7	PTIU6	PTIU5	PTIU4	PTIU3	PTIU2	PTIU1	PTIU0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	u	u	u	u	u	u	u	u

▨ = Reserved or Unimplemented u = Unaffected by reset

Figure 2-67. Port U Input Register (PTIU)

Read: Anytime. Write: Never, writes to this register have no effect.

If the associated slew rate control is enabled (digital input buffer is disabled), a read returns a “1”. If the associated slew rate control is disabled (digital input buffer is enabled), a read returns the status of the associated pin.

2.3.12.3 Port U Data Direction Register (DDRU)

Module Base + 0x003A

	7	6	5	4	3	2	1	0
R W	DDRU7	DDRU6	DDRU5	DDRU4	DDRU3	DDRU2	DDRU1	DDRU0
Reset	0	0	0	0	0	0	0	0

Figure 2-68. Port U Data Direction Register (DDRU)

Read: Anytime. Write: Anytime.

This register configures port pins PU[7:0] as either input or output.

When enabled, the SSD or MC modules force the I/O state to be an output for each associated pin and the associated Data Direction Register bit has no effect. If the SSD and MC modules are disabled, the corresponding Data Direction Register bits revert to control the I/O direction of the associated pins.

Table 2-53. DDRU Field Descriptions

Field	Description
7:0 DDRU[7:0]	**Data Direction Port U** 0 Associated pin is configured as input. 1 Associated pin is configured as output.

2.3.12.4 Port U Slew Rate Register (SRRU)

Module Base + 0x003B

	7	6	5	4	3	2	1	0
R W	SRRU7	SRRU6	SRRU5	SRRU4	SRRU3	SRRU2	SRRU1	SRRU0
Reset	0	0	0	0	0	0	0	0

Figure 2-69. Port U Slew Rate Register (SRRU)

Read: Anytime. Write: Anytime.

This register enables the slew rate control and disables the digital input buffer for port pins PU[7:0].

Table 2-54. SRRU Field Descriptions

Field	Description
7:0 SRRU[7:0]	**Slew Rate Port U** 0 Disables slew rate control and enables digital input buffer. 1 Enables slew rate control and disables digital input buffer.

2.3.12.5 Port U Pull Device Enable Register (PERU)

Module Base + 0x003C

	7	6	5	4	3	2	1	0
R W	PERU7	PERU6	PERU5	PERU4	PERU3	PERU2	PERU1	PERU0
Reset	0	0	0	0	0	0	0	0

Figure 2-70. Port U Pull Device Enable Register (PERU)

Read: Anytime. Write: Anytime.

This register configures whether a pull-up or a pull-down device is activated on configured input pins. If a pin is configured as output, the corresponding Pull Device Enable Register bit has no effect.

Table 2-55. PERU Field Descriptions

Field	Description
7:0 PERU[7:0]	**Pull Device Enable Port U** 0 Pull-up or pull-down device is disabled. 1 Pull-up or pull-down device is enabled.

2.3.12.6 Port U Polarity Select Register (PPSU)

Module Base + 0x003D

	7	6	5	4	3	2	1	0
R W	PPSU7	PPSU6	PPSU5	PPSU4	PPSU3	PPSU2	PPSU1	PPSU0
Reset	0	0	0	0	0	0	0	0

Figure 2-71. Port U Polarity Select Register (PPSU)

Read: Anytime. Write: Anytime.

The Port U Polarity Select Register selects whether a pull-down or a pull-up device is connected to the pin. The Port U Polarity Select Register is effective only when the corresponding Data Direction Register bit is set to 0 (input) and the corresponding Pull Device Enable Register bit is set to 1.

Table 2-56. PPSU Field Descriptions

Field	Description
7:0 PPSU[7:0]	**Pull Select Port U** 0 A pull-up device is connected to the associated port U pin. 1 A pull-down device is connected to the associated port U pin.

2.3.13 Port V

Port V is associated with the stepper stall detect (SSD3 and SSD2) and motor controller (MC3 and MC2) modules. Each pin is assigned to these modules according to the following priority: SSD3/SSD2 > MC3/MC2 > general-purpose I/O.

If SSD3 module is enabled, the PV[7:4] pins are controlled by the SSD3 module. If SSD3 module is disabled, the PV[7:4] pins are controlled by the motor control PWM channels 7 and 6 (MC3).

If SSD2 module is enabled, the PV[3:0] pins are controlled by the SSD2 module. If SSD2 module is disabled, the PV[3:0] pins are controlled by the motor control PWM channels 5 and 4 (MC2).

Refer to the SSD and MC block description chapters for information on enabling and disabling the SSD module and the motor control PWM channels respectively.

During reset, port V pins are configured as high-impedance inputs.

2.3.13.1 Port V I/O Register (PTV)

Module Base + 0x0040

	7	6	5	4	3	2	1	0
R W	PTV7	PTV6	PTV5	PTV4	PTV3	PTV2	PTV1	PTV0
MC:	M3C1P	M3C1M	M3C0P	M3C0M	M2C1P	M2C1M	M2C0P	M2C0M
SSD3/ SSD2	M3SINP	M3SINM	M3COSP	M3COSM	M2SINP	M2SINM	M2COSP	M2COSM
Reset	0	0	0	0	0	0	0	0

Figure 2-72. Port V I/O Register (PTV)

Read: Anytime. Write: anytime.

If the associated data direction bit (DDRVx) is set to 1 (output), a read returns the value of the I/O register bit.

If the associated data direction bit (DDRVx) is set to 0 (input) and the slew rate is enabled, the associated I/O register bit (PTVx) reads “1”.

If the associated data direction bit (DDRVx) is set to 0 (input) and the slew rate is disabled, a read returns the value of the pin.

2.3.13.2 Port V Input Register (PTIV)

Module Base + 0x0041

	7	6	5	4	3	2	1	0
R	PTIV7	PTIV6	PTIV5	PTIV4	PTIV3	PTIV2	PTIV1	PTIV0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	u	u	u	u	u	u	u	u

▨ = Reserved or Unimplemented

u = Unaffected by reset

Figure 2-73. Port V Input Register (PTIV)

Read: Anytime. Write: Never, writes to this register have no effect.

If the associated slew rate control is enabled (digital input buffer is disabled), a read returns a “1”. If the associated slew rate control is disabled (digital input buffer is enabled), a read returns the status of the associated pin.

2.3.13.3 Port V Data Direction Register (DDRV)

Module Base + 0x0042

	7	6	5	4	3	2	1	0
R W	DDRV7	DDRV6	DDRV5	DDRV4	DDRV3	DDRV2	DDRV1	DDRV0
Reset	0	0	0	0	0	0	0	0

Figure 2-74. Port V Data Direction Register (DDRV)

Read: Anytime. Write: Anytime.

This register configures port pins PV[7:0] as either input or output.

When enabled, the SSD or MC modules force the I/O state to be an output for each associated pin and the associated Data Direction Register bit has no effect. If the SSD and MC modules are disabled, the corresponding Data Direction Register bits revert to control the I/O direction of the associated pins.

Table 2-57. DDRV Field Descriptions

Field	Description
7:0 DDRV[7:0]	**Data Direction Port V** 0 Associated pin is configured as input. 1 Associated pin is configured as output.

2.3.13.4 Port V Slew Rate Register (SRRV)

Module Base + 0x0043

	7	6	5	4	3	2	1	0
R W	SRRV7	SRRV6	SRRV5	SRRV4	SRRV3	SRRV2	SRRV1	SRRV0
Reset	0	0	0	0	0	0	0	0

Figure 2-75. Port V Slew Rate Register (SRRV)

Read: anytime. Write: Anytime.

This register enables the slew rate control and disables the digital input buffer for port pins PV[7:0].

Table 2-58. SRRV Field Descriptions

Field	Description
7:0 SRRV[7:0]	**Slew Rate Port V** 0 Disables slew rate control and enables digital input buffer. 1 Enables slew rate control and disables digital input buffer.

2.3.13.5 Port V Pull Device Enable Register (PERV)

Module Base + 0x0044

	7	6	5	4	3	2	1	0
R W	PERV7	PERV6	PERV5	PERV4	PERV3	PERV2	PERV1	PERV0
Reset	0	0	0	0	0	0	0	0

Figure 2-76. Port V Pull Device Enable Register (PERV)

Read: Anytime. Write: Anytime.

This register configures whether a pull-up or a pull-down device is activated on configured input pins. If a pin is configured as output, the corresponding Pull Device Enable Register bit has no effect.

Table 2-59. PERV Field Descriptions

Field	Description
7:0 PERV[7:0]	**Pull Device Enable Port V** 0 Pull-up or pull-down device is disabled. 1 Pull-up or pull-down device is enabled.

2.3.13.6 Port V Polarity Select Register (PPSV)

Module Base + 0x0045

	7	6	5	4	3	2	1	0
R W	PPSV7	PPSV6	PPSV5	PPSV4	PPSV3	PPSV2	PPSV1	PPSV0
Reset	0	0	0	0	0	0	0	0

Figure 2-77. Port V Polarity Select Register (PPSV)

Read: Anytime. Write: Anytime.

The Port V Polarity Select Register selects whether a pull-down or a pull-up device is connected to the pin. The Port V Polarity Select Register is effective only when the corresponding Data Direction Register bit is set to 0 (input) and the corresponding Pull Device Enable Register bit is set to 1.

Table 2-60. PPSV Field Descriptions

Field	Description
7:0 PPSV[7:0]	**Pull Select Port V** 0 A pull-up device is connected to the associated port V pin. 1 A pull-down device is connected to the associated port V pin.

2.3.14 Port W

Port W is associated with the stepper stall detect (SSD5 and SSD4) and motor controller (MC5 and MC4) modules. Each pin is assigned to these modules according to the following priority: SSD5/SSD4 > MC5/MC4 > general-purpose I/O.

If SSD5 module is enabled, the PW[7:4] pins are controlled by the SSD5 module. If SSD5 module is disabled, the PW[7:4] pins are controlled by the motor control PWM channels 11 and 10 (MC5).

If SSD4 module is enabled, the PW[3:0] pins are controlled by the SSD4 module. If SSD4 module is disabled, the PW[3:0] pins are controlled by the motor control PWM channels 9 and 8 (MC4).

Refer to the SSD and MC block description chapters for information on enabling and disabling the SSD module and the motor control PWM channels respectively.

During reset, port W pins are configured as high-impedance inputs.

2.3.14.1 Port W I/O Register (PTW)

Module Base + 0x0040

	7	6	5	4	3	2	1	0
R W	PTW7	PTW6	PTW5	PTW4	PTW3	PTW2	PTW1	PTW0
MC:	M5C1P	M3C1M	M5C0P	M5C0M	M4C1P	M4C1M	M4C0P	M4C0M
SSD5/ SSD4	M5SINP	M3SINM	M5COSP	M5COSM	M4SINP	M4SINM	M4COSP	M4COSM
Reset	0	0	0	0	0	0	0	0

Figure 2-78. Port W I/O Register (PTW)

Read: Anytime. Write: anytime.

If the associated data direction bit (DDRWx) is set to 1 (output), a read returns the value of the I/O register bit.

If the associated data direction bit (DDRWx) is set to 0 (input) and the slew rate is enabled, the associated I/O register bit (PTWx) reads “1”.

If the associated data direction bit (DDRWx) is set to 0 (input) and the slew rate is disabled, a read returns the value of the pin.

2.3.14.2 Port W Input Register (PTIW)

Module Base + 0x0041

	7	6	5	4	3	2	1	0
R	PTIW7	PTIW6	PTIW5	PTIW4	PTIW3	PTIW2	PTIW1	PTIW0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	u	u	u	u	u	u	u	u

▨ = Reserved or Unimplemented

u = Unaffected by reset

Figure 2-79. Port W Input Register (PTIW)

Read: Anytime. Write: Never, writes to this register have no effect.

If the associated slew rate control is enabled (digital input buffer is disabled), a read returns a “1”. If the associated slew rate control is disabled (digital input buffer is enabled), a read returns the status of the associated pin.

2.3.14.3 Port W Data Direction Register (DDRW)

Module Base + 0x0042

	7	6	5	4	3	2	1	0
R W	DDRW7	DDRW6	DDRW5	DDRW4	DDRW3	DDRW2	DDRW1	DDRW0
Reset	0	0	0	0	0	0	0	0

Figure 2-80. Port W Data Direction Register (DDRW)

Read: Anytime. Write: Anytime.

This register configures port pins PW[7:0] as either input or output.

When enabled, the SSD or MC modules force the I/O state to be an output for each associated pin and the associated Data Direction Register bit has no effect. If the SSD and MC modules are disabled, the corresponding Data Direction Register bits revert to control the I/O direction of the associated pins.

Table 2-61. DDRW Field Descriptions

Field	Description
7:0 DDRW[7:0]	**Data Direction Port W** 0 Associated pin is configured as input. 1 Associated pin is configured as output.

2.3.14.4 Port W Slew Rate Register (SRRW)

Module Base + 0x0043

	7	6	5	4	3	2	1	0
R W	SRRW7	SRRW6	SRRW5	SRRW4	SRRW3	SRRW2	SRRW1	SRRW0
Reset	0	0	0	0	0	0	0	0

Figure 2-81. Port W Slew Rate Register (SRRW)

Read: anytime. Write: Anytime.

This register enables the slew rate control and disables the digital input buffer for port pins PW[7:0].

Table 2-62. SRRW Field Descriptions

Field	Description
7:0 SRRW[7:0]	**Slew Rate Port W** 0 Disables slew rate control and enables digital input buffer. 1 Enables slew rate control and disables digital input buffer.

2.3.14.5 Port W Pull Device Enable Register (PERW)

Module Base + 0x0044

	7	6	5	4	3	2	1	0
R W	PERW7	PERW6	PERW5	PERW4	PERW3	PERW2	PERW1	PERW0
Reset	0	0	0	0	0	0	0	0

Figure 2-82. Port W Pull Device Enable Register (PERW)

Read: Anytime. Write: Anytime.

This register configures whether a pull-up or a pull-down device is activated on configured input pins. If a pin is configured as output, the corresponding Pull Device Enable Register bit has no effect.

Table 2-63. PERW Field Descriptions

Field	Description
7:0 PERW[7:0]	**Pull Device Enable Port W** 0 Pull-up or pull-down device is disabled. 1 Pull-up or pull-down device is enabled.

2.3.14.6 Port W Polarity Select Register (PPSW)

Module Base + 0x0045

	7	6	5	4	3	2	1	0
R W	PPSW7	PPSW6	PPSW5	PPSW4	PPSW3	PPSW2	PPSW1	PPSW0
Reset	0	0	0	0	0	0	0	0

Figure 2-83. Port W Polarity Select Register (PPSW)

Read: Anytime. Write: Anytime.

The Port W Polarity Select Register selects whether a pull-down or a pull-up device is connected to the pin. The Port W Polarity Select Register is effective only when the corresponding Data Direction Register bit is set to 0 (input) and the corresponding Pull Device Enable Register bit is set to 1.

Table 2-64. PPSW Field Descriptions

Field	Description
7:0 PPSW[7:0]	**Pull Select Port W** 0 A pull-up device is connected to the associated port W pin. 1 A pull-down device is connected to the associated port W pin.

2.4 Functional Description

Each pin except PE0, PE1, and BKGD can act as general-purpose I/O. In addition the pin can act as an output from a peripheral module or an input to a peripheral module.

A set of configuration registers is common to all ports. All registers can be written at any time, however a specific configuration might not become active.

> Example: Selecting a pull-up resistor. This resistor does not become active while the port is used as a push-pull output.

Table 2-65. Register Availability per Port[1]

Port	Data	Data Direction	Input	Reduced Drive	Pull Enable	Polarity Select	Wired-OR Mode	Slew Rate	Interrupt Enable	Interrupt Flag
A	yes	yes	—	yes	yes	—	—	yes	—	—
B	yes	yes	—			—	—		—	—
C	yes	yes	—			—	—		—	—
D	yes	yes	—			—	—		—	—
E	yes	yes	—			—	—		—	—
K	yes	yes	—			—	—		—	—
AD	yes	yes	yes	yes	yes	yes	—	—	yes	yes
L	yes	yes	yes	yes	yes	yes	—	yes	—	—
M	yes	yes	yes	yes	yes	yes	yes	yes	—	—
P	yes	yes	yes	yes	yes	yes	yes	yes	—	—
S	yes	yes	yes	yes	yes	yes	yes	yes	—	—
T	yes	yes	yes	yes	yes	yes	yes	yes	—	—
U	yes	yes	yes	—	yes	yes	—	yes	—	—
V	yes	yes	yes	—	yes	yes	—	yes	—	—
W	yes	yes	yes	—	yes	yes	—	yes	—	—

[1] Each cell represents one register with individual configuration bits

2.4.1 I/O Register

The I/O Register holds the value driven out to the pin if the port is used as a general-purpose I/O. Writing to the I/O Register only has an effect on the pin if the port is used as general-purpose output.

When reading the I/O Register, the value of each pin is returned if the corresponding Data Direction Register bit is set to 0 (pin configured as input). If the data direction register bits is set to 1, the content of the I/O Register bit is returned. This is independent of any other configuration (Figure 2-84).

Due to internal synchronization circuits, it can take up to 2 bus cycles until the correct value is read on the I/O Register when changing the data direction register.

2.4.2 Input Register

The Input Register is a read-only register and generally returns the value of the pin (Figure 2-84).It can be used to detect overload or short circuit conditions.

Due to internal synchronization circuits, it can take up to 2 bus cycles until the correct value is read on the Input Register when changing the Data Direction Register.

2.4.3 Data Direction Register

The Data Direction Register defines whether the pin is used as an input or an output.

. A Data Direction Register bit set to 0 configures the pin as an input. A Data Direction Register bit set to 1 configures the pin as an output. If a peripheral module controls the pin the contents of the data direction register is ignored (Figure 2-84).

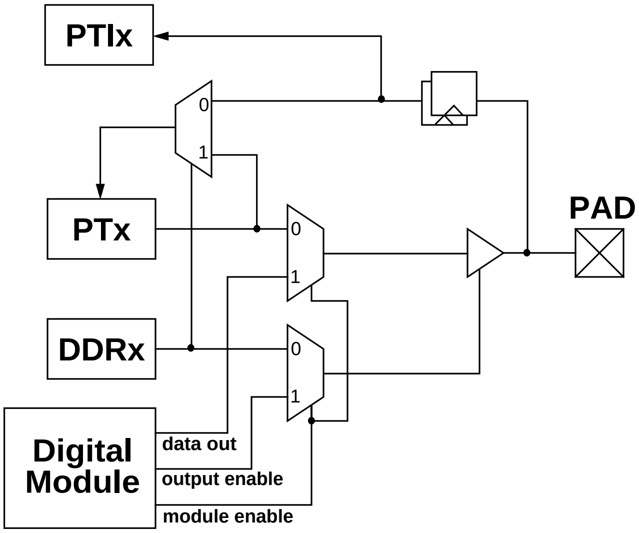


Figure 2-84. Illustration of I/O Pin Functionality

Figure 2-85 shows the state of digital inputs and outputs when an analog module drives the port. When the analog module is enabled all associated digital output ports are disabled and all associated digital input ports read “1”t.

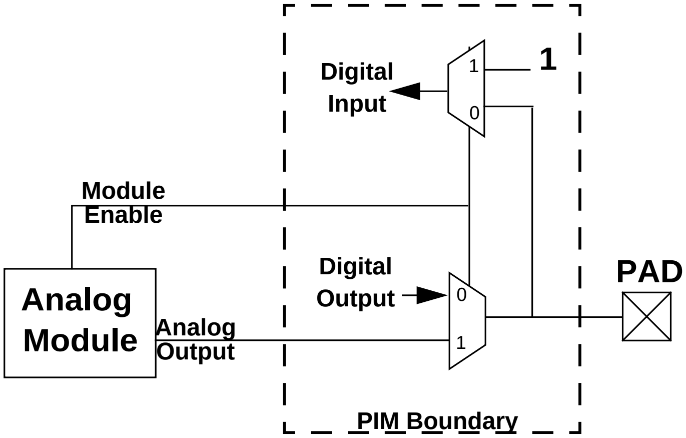


Figure 2-85. Digital Ports and Analog Module

2.4.4 Reduced Drive Register

If the port is used as an output the Reduced Drive Register allows the configuration of the drive strength.

2.4.5 Pull Device Enable Register

The Pull Device Enable Register turns on a pull-up or pull-down device. The pull device becomes active only if the pin is used as an input or as a wired-or output.

2.4.6 Polarity Select Register

The Polarity Select Register selects either a pull-up or pull-down device if enabled. The pull device becomes active only if the pin is used as an input or as a wired-or output.

2.4.7 Pin Configuration Summary

The following table summarizes the effect of various configuration in the Data Direction (DDR), Input/Output (I/O), reduced drive (RDR), Pull Enable (PE), Pull Select (PS) and Interrupt Enable (IE) register bits. The PS configuration bit is used for two purposes:

1. Configure the sensitive interrupt edge (rising or falling), if interrupt is enabled.
2. Select either a pull-up or pull-down device if PE is set to “1”.

Table 2-66. Pin Configuration Summary

DDR	IO	RDR	PE	PS	IE[1]	Function[2]	Pull Device	Interrupt
0	X	X	0	X	0	Input	Disabled	Disabled
0	X	X	1	0	0	Input	Pull Up	Disabled
0	X	X	1	1	0	Input	Pull Down	Disabled
0	X	X	0	0	1	Input	Disabled	Falling Edge
0	X	X	0	1	1	Input	Disabled	Rising Edge
0	X	X	1	0	1	Input	Pull Up	Falling Edge
0	X	X	1	1	1	Input	Pull Down	Rising Edge
1	0	0	X	X	0	Output to 0, Full Drive	Disabled	Disabled
1	1	0	X	X	0	Output to 1, Full Drive	Disabled	Disabled
1	0	1	X	X	0	Output to 0, Reduced Drive	Disabled	Disabled
1	1	1	X	X	0	Output to 1, Reduced Drive	Disabled	Disabled
1	0	0	X	0	1	Output to 0, Full Drive	Disabled	Falling Edge
1	1	0	X	1	1	Output to 1, Full Drive	Disabled	Rising Edge
1	0	1	X	0	1	Output to 0, Reduced Drive	Disabled	Falling Edge
1	1	1	X	1	1	Output to 1, Reduced Drive	Disabled	Rising Edge

[1] Applicable only on Port AD.

[2] Digital outputs are disabled and digital input logic is forced to “1” when an analog module associated with the port is enabled.

2.5 Resets

The reset values of all registers are given in the register description in Section 2.3, “Memory Map and Register Definition”.

All ports start up as general-purpose inputs on reset.

2.5.1 Reset Initialization

All registers including the data registers get set/reset asynchronously. Table 2-67 summarizes the port properties after reset initialization.

Table 2-67. Port Reset State Summary

Port	Reset States					
	Data Direction	**Pull Mode**	**Reduced Drive**	**Slew Rate**	**Wired-OR Mode**	**Interrupt**
A	Input	Pull Down	Disabled	Disabled	N/A	N/A
B	Input	Pull Down	Disabled	Disabled	N/A	N/A
C	Input	Hi-z	Disabled	Disabled	N/A	N/A
D	Input	Hi-z	Disabled	Disabled	N/A	N/A
E	Input	Pull Down[1]	Disabled	Disabled	N/A	N/A
K	Input	Pull Down	Disabled	Disabled	N/A	N/A
AD	Input	Hi-z	Disabled	N/A	N/A	Disabled
L	Input	Pull Down	Disabled	Disabled	N/A	N/A
M	Input	Hi-z	Disabled	Disabled	Disabled	N/A
P	Input	Hi-z	Disabled	Disabled	Disabled	N/A
S	Input	Hi-z	Disabled	Disabled	Disabled	N/A
T[7:4]	Input	Hi-z	Disabled	Disabled	Disabled	N/A
T[3:0]	Input	Pull Down	Disabled	Disabled	Disabled	N/A
U	Input	Hi-z	Disabled	Disabled	N/A	N/A
V	Input	Hi-z	Disabled	Disabled	N/A	N/A
W	Input	Hi-z	Disabled	Disabled	N/A	N/A

[1] PE[1:0] pins have pull-ups instead of pull-downs.

2.6 Interrupts

2.6.1 General

Port AD generates an edge sensitive interrupt if enabled. It offers eight I/O pins with edge triggered interrupt capability in wired-or fashion. The interrupt enable as well as the sensitivity to rising or falling edges can be individually configured on per pin basis. All eight bits/pins share the same interrupt vector. Interrupts can be used with the pins configured as inputs (with the corresponding ATDDIEN1 bit set to 1)or outputs.

An interrupt is generated when a bit in the port interrupt flag register and its corresponding port interrupt enable bit are both set. This external interrupt feature is capable to wake up the CPU when it is in stop or wait mode.

A digital filter on each pin prevents pulses (Figure 2-87) shorter than a specified time from generating an interrupt. The minimum time varies over process conditions, temperature and voltage (Figure 2-86 and Table 2-68).

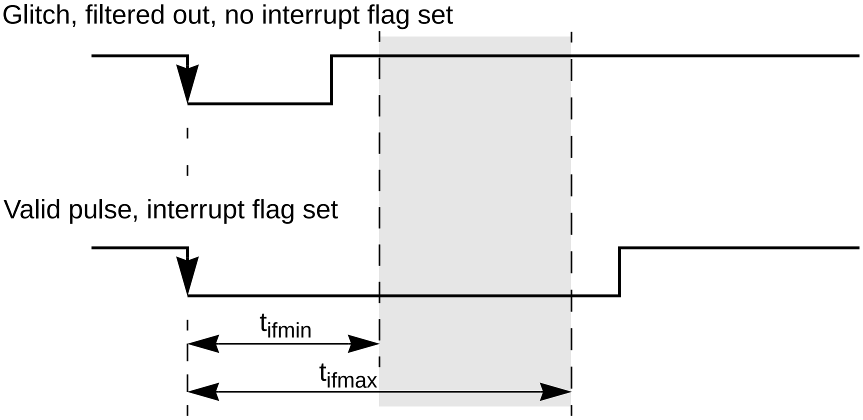


Figure 2-86. Interrupt Glitch Filter on Port AD (PPS = 0)

Table 2-68. Pulse Detection Criteria

Pulse	Mode			
	$\overline{\text{STOP}}$		STOP[1]	
		Unit		Unit
Ignored	t_{pulse} <= 3	Bus Clock	t_{pulse} <= 3.2	s
Uncertain	3 < t_{pulse} < 4	Bus Clock	3.2 < t_{pulse} < 10	s
Valid	t_{pulse} >= 4	Bus Clock	t_{pulse} >= 10	s

[1] These values include the spread of the oscillator frequency over temperature, voltage and process.

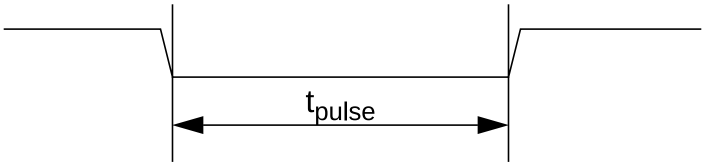


Figure 2-87. Pulse Illustration

A valid edge on an input is detected if 4 consecutive samples of a passive level are followed by 4 consecutive samples of an active level directly or indirectly

The filters are continuously clocked by the bus clock in RUN and WAIT mode. In STOP mode the clock is generated by a single RC oscillator in the port integration module. To maximize current saving the RC oscillator runs only if the following condition is true on any pin:Sample count <= 4 and port interrupt enabled (PIE=1) and port interrupt flag not set (PIF=0).

2.6.2 Interrupt Sources

Table 2-69. Port Integration Module Interrupt Sources

Interrupt Source	Interrupt Flag	Local Enable	Global (CCR) Mask
Port AD	PIFAD[7:0]	PIEAD[7:0]	I Bit

NOTE

Vector addresses and their relative interrupt priority are determined at the MCU level.

2.6.3 Operation in Stop Mode

All clocks are stopped in STOP mode. The port integration module has asynchronous paths on port AD to generate wake-up interrupts from stop mode. For other sources of external interrupts refer to the respective block description chapters.

Chapter 3
512 Kbyte Flash Module (S12XFTX512K4V3)

3.1 Introduction

This document describes the FTX512K4 module that includes a 512 Kbyte Flash (nonvolatile) memory. The Flash memory may be read as either bytes, aligned words or misaligned words. Read access time is one bus cycle for bytes and aligned words, and two bus cycles for misaligned words.

The Flash memory is ideal for program and data storage for single-supply applications allowing for field reprogramming without requiring external voltage sources for program or erase. Program and erase functions are controlled by a command driven interface. The Flash module supports both block erase and sector erase. An erased bit reads 1 and a programmed bit reads 0. The high voltage required to program and erase the Flash memory is generated internally. It is not possible to read from a Flash block while it is being erased or programmed.

CAUTION

> A Flash word must be in the erased state before being programmed. Cumulative programming of bits within a Flash word is not allowed.

3.1.1 Glossary

Command Write Sequence — A three-step MCU instruction sequence to execute built-in algorithms (including program and erase) on the Flash memory.

Multiple-Input Signature Register (MISR) — A Multiple-Input Signature Register is an output response analyzer implemented using a linear feedback shift-register (LFSR). A 16-bit MISR is used to compress data and generate a signature that is particular to the data read from a Flash block.

3.1.2 Features

- 512 Kbytes of Flash memory comprised of four 128 Kbyte blocks with each block divided into 128 sectors of 1024 bytes
- Automated program and erase algorithm
- Interrupts on Flash command completion, command buffer empty
- Fast sector erase and word program operation
- 2-stage command pipeline for faster multi-word program times
- Sector erase abort feature for critical interrupt response
- Flexible protection scheme to prevent accidental program or erase
- Single power supply for all Flash operations including program and erase

- Security feature to prevent unauthorized access to the Flash memory
- Code integrity check using built-in data compression

3.1.3 Modes of Operation

Program, erase, erase verify, and data compress operations (please refer to Section 3.4.1, “Flash Command Operations” for details).

3.1.4 Block Diagram

A block diagram of the Flash module is shown in Figure 3-1.

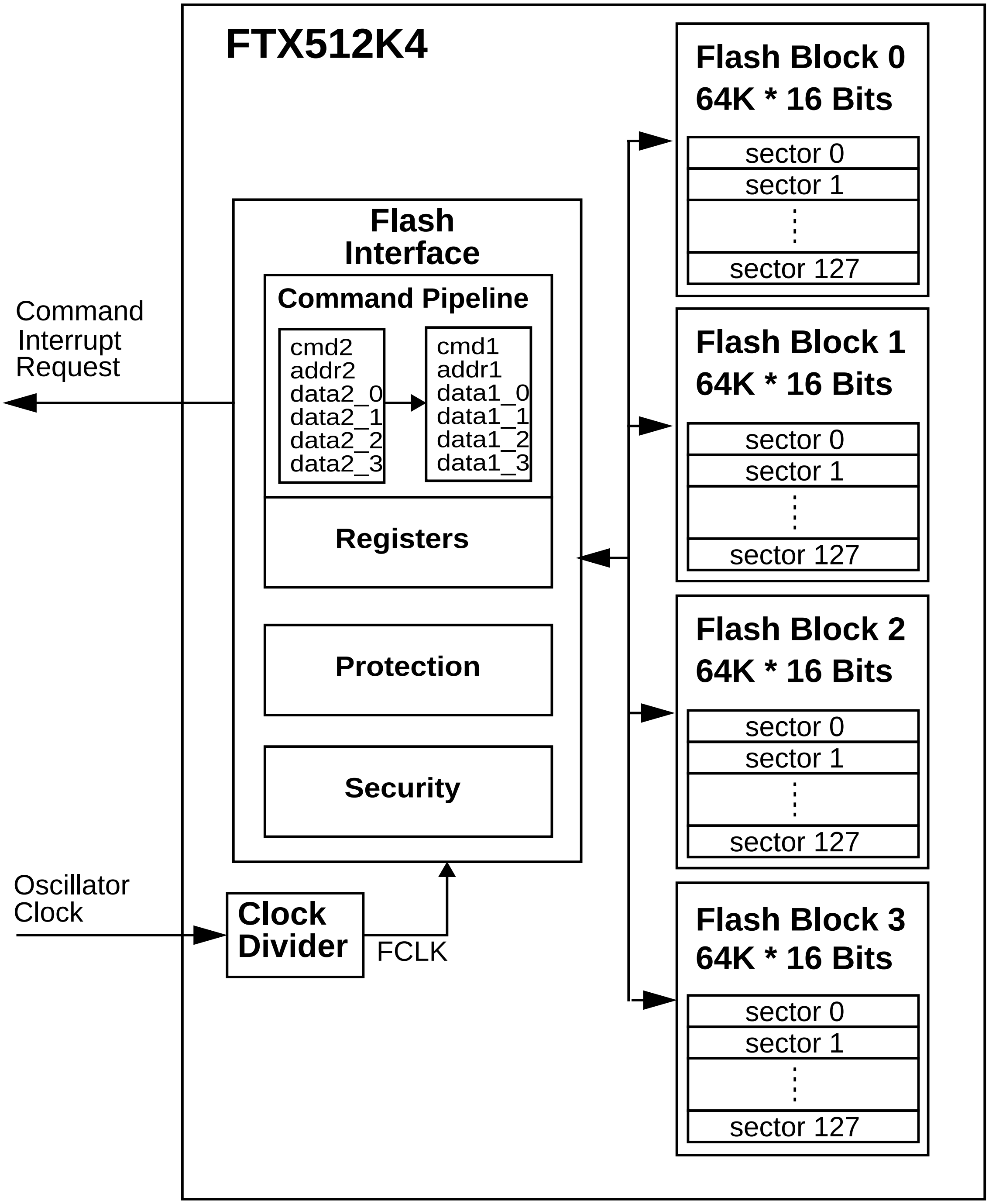


Figure 3-1. FTX512K4 Block Diagram

3.2 External Signal Description

The Flash module contains no signals that connect off-chip.

3.3 Memory Map and Register Definition

This section describes the memory map and registers for the Flash module.

3.3.1 Module Memory Map

The Flash memory map is shown in Figure 3-2. The HCS12X architecture places the Flash memory addresses between global addresses 0x78_0000 and 0x7F_FFFF. The FPROT register, described in Section 3.3.2.5, “Flash Protection Register (FPROT)”, can be set to protect regions in the Flash memory from accidental program or erase. Three separate memory regions, one growing upward from global address 0x7F_8000 in the Flash memory (called the lower region), one growing downward from global address 0x7F_FFFF in the Flash memory (called the higher region), and the remaining addresses in the Flash memory, can be activated for protection. The Flash memory addresses covered by these protectable regions are shown in the Flash memory map. The higher address region is mainly targeted to hold the boot loader code since it covers the vector space. The lower address region can be used for EEPROM emulation in an MCU without an EEPROM module since it can be left unprotected while the remaining addresses are protected from program or erase. Default protection settings as well as security information that allows the MCU to restrict access to the Flash module are stored in the Flash configuration field as described in Table 3-1.

Table 3-1. Flash Configuration Field

Global Address	Size (Bytes)	Description
0x7F_FF00 – 0x7F_FF07	8	Backdoor Comparison Key Refer to Section 3.6.1, “Unsecuring the MCU using Backdoor Key Access”
0x7F_FF08 – 0x7F_FF0C	5	Reserved
0x7F_FF0D	1	Flash Protection byte Refer to Section 3.3.2.5, “Flash Protection Register (FPROT)”
0x7F_FF0E	1	Flash Nonvolatile byte Refer to Section 3.3.2.8, “Flash Control Register (FCTL)”
0x7F_FF0F	1	Flash Security byte Refer to Section 3.3.2.2, “Flash Security Register (FSEC)”

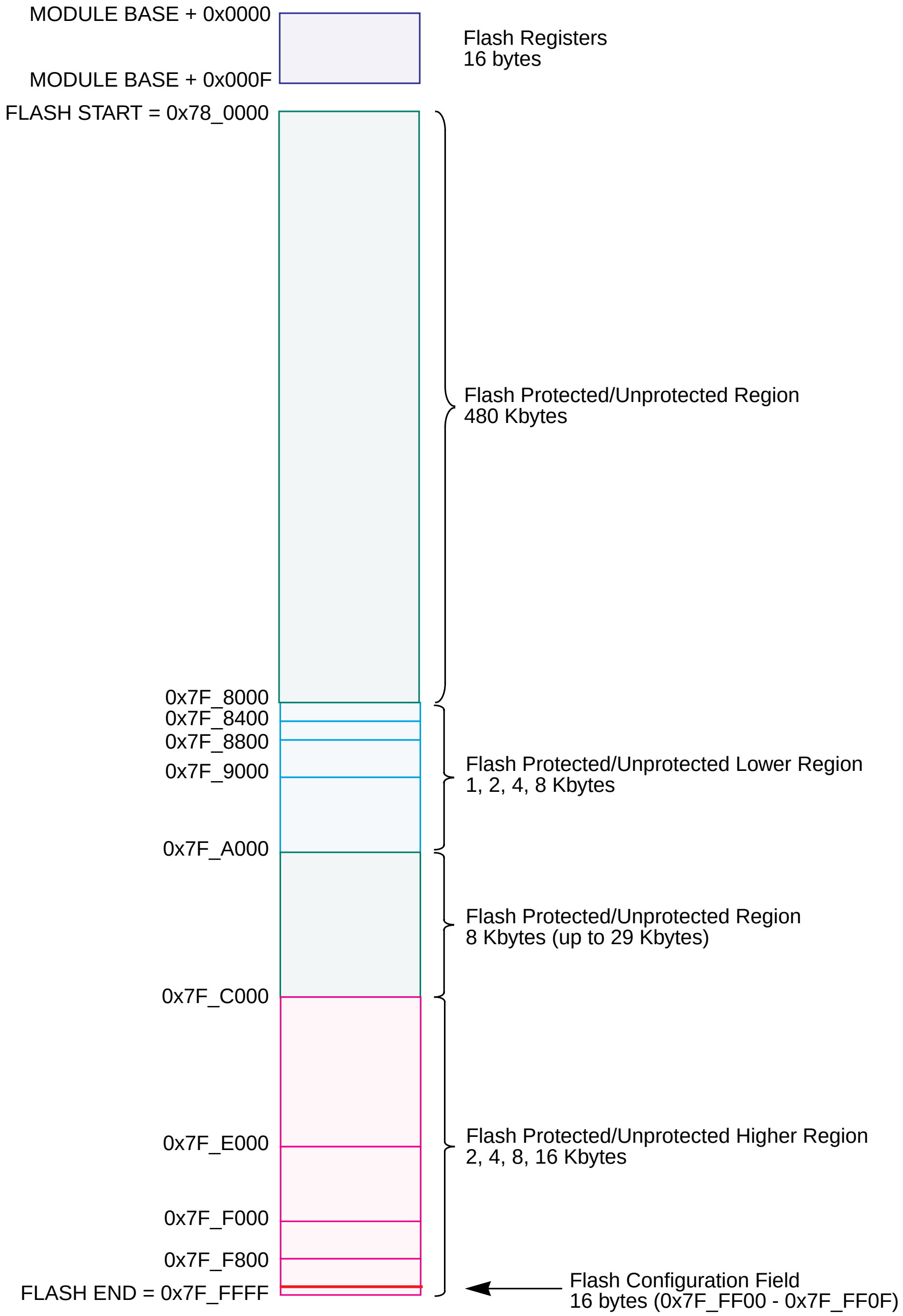


Figure 3-2. Flash Memory Map

3.3.2 Register Descriptions

The Flash module contains a set of 16 control and status registers located between module base + 0x0000 and 0x000F. A summary of the Flash module registers is given in Figure 3-3. Detailed descriptions of each register bit are provided.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 FCLKDIV	R	FDIVLD	PRDIV8	FDIV5	FDIV4	FDIV3	FDIV2	FDIV1	FDIV0
	W								
0x0001 FSEC	R	KEYEN		RNV5	RNV4	RNV3	RNV2	SEC	
	W								
0x0002 FTSTMOD	R	0	MRDS		0	0	0	0	0
	W								
0x0003 FCNFG	R	CBEIE	CCIE	KEYACC	0	0	0	0	0
	W								
0x0004 FPROT	R	FPOPEN	RNV6	FPHDIS	FPHS		FPLDIS	FPLS	
	W								
0x0005 FSTAT	R	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	0	0
	W								
0x0006 FCMD	R	0	CMDB						
	W								
0x0007 FCTL	R	0	NV6	NV5	NV4	NV3	NV2	NV1	NV0
	W								
0x0008 FADDRHI	R	FADDRHI							
	W								
0x0009 FADDRLO	R	FADDRLO							
	W								
0x000A FDATAHI	R	FDATAHI							
	W								
0x000B FDATALO	R	FDATALO							
	W								

Figure 3-3. FTX512K4 Register Summary

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x000C RESERVED1	R	0	0	0	0	0	0	0	0
	W								
0x000D RESERVED2	R	0	0	0	0	0	0	0	0
	W								
0x000E RESERVED3	R	0	0	0	0	0	0	0	0
	W								
0x000F RESERVED4	R	0	0	0	0	0	0	0	0
	W								

Figure 3-3. FTX512K4 Register Summary (continued)

3.3.2.1 Flash Clock Divider Register (FCLKDIV)

The FCLKDIV register is used to control timed events in program and erase algorithms.

Module Base + 0x0000

	7	6	5	4	3	2	1	0
R	FDIVLD	PRDIV8	FDIV5	FDIV4	FDIV3	FDIV2	FDIV1	FDIV0
W	▨							
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-4. Flash Clock Divider Register (FCLKDIV)

All bits in the FCLKDIV register are readable, bits 6-0 are write once and bit 7 is not writable.

Table 3-2. FCLKDIV Field Descriptions

Field	Description
7 FDIVLD	**Clock Divider Loaded.** 0 Register has not been written. 1 Register has been written to since the last reset.
6 PRDIV8	**Enable Prescalar by 8.** 0 The oscillator clock is directly fed into the clock divider. 1 The oscillator clock is divided by 8 before feeding into the clock divider.
5:0 FDIV[5:0]	**Clock Divider Bits** — The combination of PRDIV8 and FDIV[5:0] must divide the oscillator clock down to a frequency of 150 kHz–200 kHz. The maximum divide ratio is 512. Please refer to Section 3.4.1.1, “Writing the FCLKDIV Register” for more information.

3.3.2.2 Flash Security Register (FSEC)

The FSEC register holds all bits associated with the security of the MCU and Flash module.

Module Base + 0x0001

	7	6	5	4	3	2	1	0
R	KEYEN		RNV5	RNV4	RNV3	RNV2	SEC	
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	F	F	F	F	F	F	F	F

▨ = Unimplemented or Reserved

Figure 3-5. Flash Security Register (FSEC)

All bits in the FSEC register are readable but are not writable.

The FSEC register is loaded from the Flash Configuration Field at address 0x7F_FF0F during the reset sequence, indicated by F in Figure 3-5.

Table 3-3. FSEC Field Descriptions

Field	Description
7:6 KEYEN[1:0]	**Backdoor Key Security Enable Bits** — The KEYEN[1:0] bits define the enabling of backdoor key access to the Flash module as shown in Table 3-4.
5:2 RNV[5:2]	**Reserved Nonvolatile Bits** — The RNV[5:2] bits should remain in the erased state for future enhancements.
1:0 SEC[1:0]	**Flash Security Bits** — The SEC[1:0] bits define the security state of the MCU as shown in Table 3-5. If the Flash module is unsecured using backdoor key access, the SEC[1:0] bits are forced to 1:0.

Table 3-4. Flash KEYEN States

KEYEN[1:0]	Status of Backdoor Key Access
00	DISABLED
01[1]	DISABLED
10	ENABLED
11	DISABLED

[1] Preferred KEYEN state to disable Backdoor Key Access.

Table 3-5. Flash Security States

SEC[1:0]	Status of Security
00	SECURED
01[1]	SECURED
10	UNSECURED
11	SECURED

[1] Preferred SEC state to set MCU to secured state.

The security function in the Flash module is described in Section 3.6, “Flash Module Security”.

3.3.2.3 Flash Test Mode Register (FTSTMOD)

The FTSTMOD register is used to control Flash test features.

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R	0	MRDS		0	0	0	0	0
W	▨			▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-6. Flash Test Mode Register (FTSTMOD —Normal Mode)

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R	0	MRDS		WRALL	0	0	0	0
W	▨				▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-7. Flash Test Mode Register (FTSTMOD — Special Mode)

MRDS bits are readable and writable while all remaining bits read 0 and are not writable in normal mode. The WRALL bit is writable only in special mode to simplify mass erase and erase verify operations. When writing to the FTSTMOD register in special mode, all unimplemented/reserved bits must be written to 0.

Table 3-6. FTSTMOD Field Descriptions

Field	Description
6:5 MRDS[1:0]	**Margin Read Setting** — The MRDS[1:0] bits are used to set the sense-amp margin level for reads of the Flash array as shown in Table 3-7.
4 WRALL	**Write to all Register Banks** — If the WRALL bit is set, all banked FDATA registers sharing the same register address will be written simultaneously during a register write. 0 Write only to the FDATA register bank selected using BKSEL. 1 Write to all FDATA register banks.

Table 3-7. FTSTMOD Margin Read Settings

MRDS[1:0]	Margin Read Setting
00	Normal
01	Program Margin[1]
10	Erase Margin[2]
11	Normal

[1] Flash array reads will be sensitive to program margin.

[2] Flash array reads will be sensitive to erase margin.

3.3.2.4 Flash Configuration Register (FCNFG)

The FCNFG register enables the Flash interrupts and gates the security backdoor writes.

Module Base + 0x0003

	7	6	5	4	3	2	1	0
R	CBEIE	CCIE	KEYACC	0	0	0	0	0
W				▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-8. Flash Configuration Register (FCNFG — Normal Mode)

Module Base + 0x0003

	7	6	5	4	3	2	1	0
R	CBEIE	CCIE	KEYACC	0	0	0	BKSEL	
W				▨	▨	▨		
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-9. Flash Configuration Register (FCNFG — Special Mode)

CBEIE, CCIE and KEYACC bits are readable and writable while all remaining bits read 0 and are not writable in normal mode. KEYACC is only writable if KEYEN (see Section 3.3.2.2, “Flash Security Register (FSEC)” is set to the enabled state. BKSEL is readable and writable in special mode to simplify mass erase and erase verify operations. When writing to the FCNFG register in special mode, all unimplemented/ reserved bits must be written to 0.

Table 3-8. FCNFG Field Descriptions

Field	Description
7 CBEIE	**Command Buffer Empty Interrupt Enable** — The CBEIE bit enables an interrupt in case of an empty command buffer in the Flash module. 0 Command buffer empty interrupt disabled. 1 An interrupt will be requested whenever the CBEIF flag (see Section 3.3.2.6, “Flash Status Register (FSTAT)”) is set.
6 CCIE	**Command Complete Interrupt Enable** — The CCIE bit enables an interrupt in case all commands have been completed in the Flash module. 0 Command complete interrupt disabled. 1 An interrupt will be requested whenever the CCIF flag (see Section 3.3.2.6, “Flash Status Register (FSTAT)”) is set.
5 KEYACC	**Enable Security Key Writing** 0 Flash writes are interpreted as the start of a command write sequence. 1 Writes to Flash array are interpreted as keys to open the backdoor. Reads of the Flash array return invalid data.
1:0 BKSEL[1:0]	**Block Select** — The BKSEL[1:0] bits indicates which register bank is active according to Table 3-9.

Table 3-9. Flash Register Bank Selects

BKSEL[1:0]	Selected Block
00	Flash Block 0
01	Flash Block 1
10	Flash Block 2
11	Flash Block 3

3.3.2.5 Flash Protection Register (FPROT)

The FPROT register defines which Flash sectors are protected against program or erase operations.

Module Base + 0x0004

	7	6	5	4	3	2	1	0
R	FPOPEN	RNV6	FPHDIS	FPHS		FPLDIS	FPLS	
W		▨						
Reset	F	F	F	F	F	F	F	F

▨ = Unimplemented or Reserved

Figure 3-10. Flash Protection Register (FPROT)

All bits in the FPROT register are readable and writable with restrictions (see Section 3.3.2.5.1, “Flash Protection Restrictions”) except for RNV[6] which is only readable.

During the reset sequence, the FPROT register is loaded from the Flash Configuration Field at global address 0x7F_FF0D. To change the Flash protection that will be loaded during the reset sequence, the upper sector of the Flash memory must be unprotected, then the Flash Protect/Security byte located as described in Table 3-1 must be reprogrammed.

Trying to alter data in any protected area in the Flash memory will result in a protection violation error and the PVIOL flag will be set in the FSTAT register. The mass erase of a Flash block is not possible if any of the Flash sectors contained in the Flash block are protected.

Table 3-10. FPROT Field Descriptions

Field	Description
7 FPOPEN	**Flash Protection Open** — The FPOPEN bit determines the protection function for program or erase as shown in Table 3-11. 0 The FPHDIS and FPLDIS bits define unprotected address ranges as specified by the corresponding FPHS[1:0] and FPLS[1:0] bits. For an MCU without an EEPROM module, the FPOPEN clear state allows the main part of the Flash block to be protected while a small address range can remain unprotected for EEPROM emulation. 1 The FPHDIS and FPLDIS bits enable protection for the address range specified by the corresponding FPHS[1:0] and FPLS[1:0] bits.
6 RNV6	**Reserved Nonvolatile Bit** — The RNV[6] bit should remain in the erased state for future enhancements.

Table 3-10. FPROT Field Descriptions (continued)

Field	Description
5 FPHDIS	**Flash Protection Higher Address Range Disable** — The FPHDIS bit determines whether there is a protected/unprotected area in a specific region of the Flash memory ending with global address 0x7F_FFFF. 0 Protection/Unprotection enabled. 1 Protection/Unprotection disabled.
4:3 FPHS[1:0]	**Flash Protection Higher Address Size** — The FPHS[1:0] bits determine the size of the protected/unprotected area as shown inTable 3-12. The FPHS[1:0] bits can only be written to while the FPHDIS bit is set.
2 FPLDIS	**Flash Protection Lower Address Range Disable** — The FPLDIS bit determines whether there is a protected/unprotected area in a specific region of the Flash memory beginning with global address 0x7F_8000. 0 Protection/Unprotection enabled. 1 Protection/Unprotection disabled.
1:0 FPLS[1:0]	**Flash Protection Lower Address Size** — The FPLS[1:0] bits determine the size of the protected/unprotected area as shown in Table 3-13. The FPLS[1:0] bits can only be written to while the FPLDIS bit is set.

Table 3-11. Flash Protection Function

FPOPEN	FPHDIS	FPLDIS	Function[1]
1	1	1	No Protection
1	1	0	Protected Low Range
1	0	1	Protected High Range
1	0	0	Protected High and Low Ranges
0	1	1	Full Flash memory Protected
0	1	0	Unprotected Low Range
0	0	1	Unprotected High Range
0	0	0	Unprotected High and Low Ranges

[1] For range sizes, refer to Table 3-12 and Table 3-13.

Table 3-12. Flash Protection Higher Address Range

FPHS[1:0]	Global Address Range	Protected Size
00	0x7F_F800–0x7F_FFFF	2 Kbytes
01	0x7F_F000–0x7F_FFFF	4 Kbytes
10	0x7F_E000–0x7F_FFFF	8 Kbytes
11	0x7F_C000–0x7F_FFFF	16 Kbytes

Table 3-13. Flash Protection Lower Address Range

FPLS[1:0]	Global Address Range	Protected Size
00	0x7F_8000–0x7F_83FF	1 Kbytes
01	0x7F_8000–0x7F_87FF	2 Kbytes
10	0x7F_8000–0x7F_8FFF	4 Kbytes
11	0x7F_8000–0x7F_9FFF	8 Kbytes

All possible Flash protection scenarios are shown in Figure 3-11. Although the protection scheme is loaded from the Flash array at global address 0x7F_FF0D during the reset sequence, it can be changed by the user. This protection scheme can be used by applications requiring re-programming in single chip mode while providing as much protection as possible if re-programming is not required.

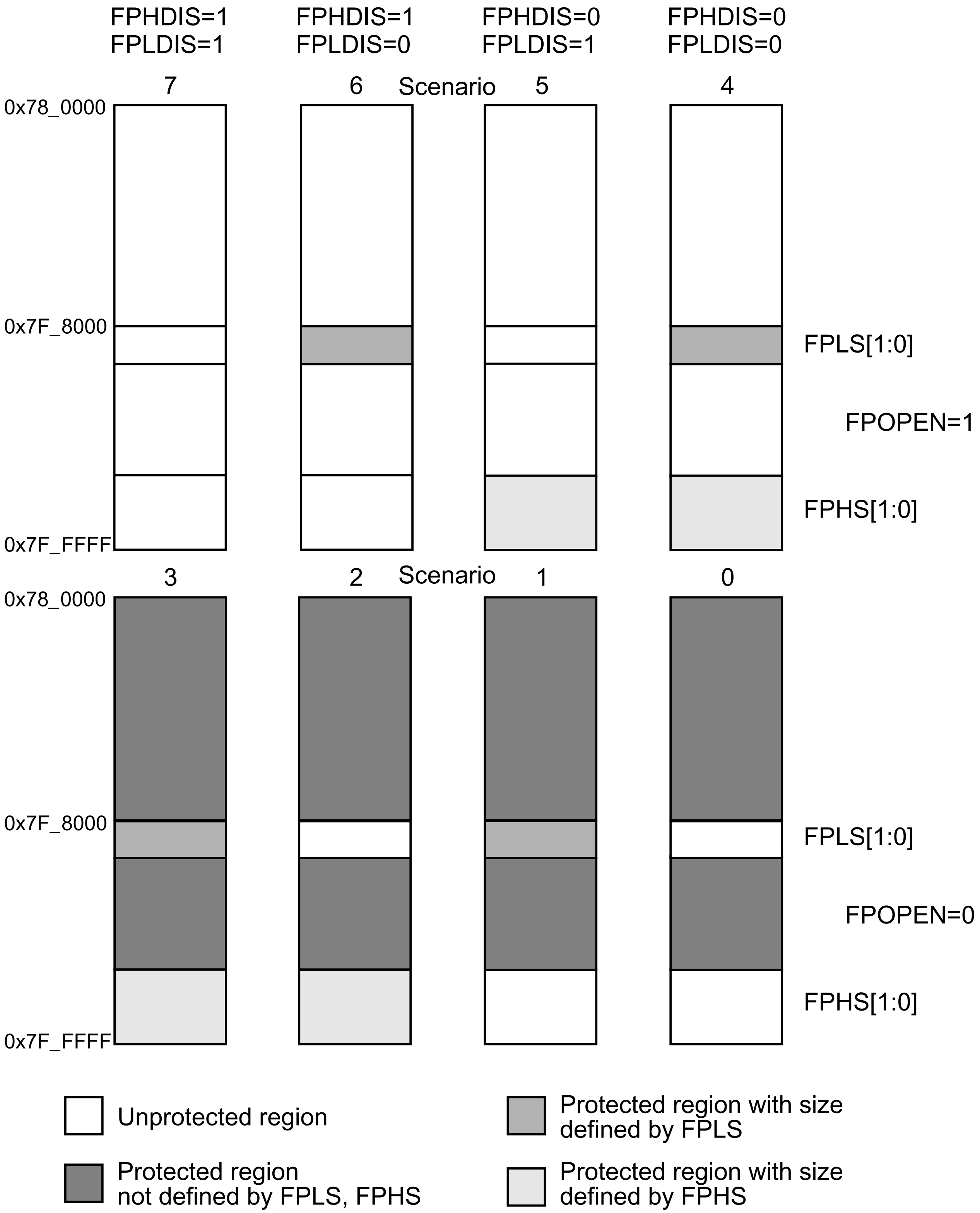


Figure 3-11. Flash Protection Scenarios

3.3.2.5.1 Flash Protection Restrictions

The general guideline is that Flash protection can only be added and not removed. Table 3-14 specifies all valid transitions between Flash protection scenarios. Any attempt to write an invalid scenario to the FPROT register will be ignored and the FPROT register will remain unchanged. The contents of the FPROT register reflect the active protection scenario. See the FPHS and FPLS descriptions for additional restrictions.

Table 3-14. Flash Protection Scenario Transitions

From Protection Scenario	To Protection Scenario[1]							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	X	X	X	X				
1		X		X				
2			X	X				
3				X				
4				X	X			
5			X	X	X	X		
6		X		X	X		X	
7	X	X	X	X	X	X	X	X

[1] Allowed transitions marked with X.

3.3.2.6 Flash Status Register (FSTAT)

The FSTAT register defines the operational status of the module.

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	0	0
W		▨			▨	▨	▨	▨
Reset	1	1	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-12. Flash Status Register (FSTAT — Normal Mode)

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	FAIL	0
W		▨			▨	▨		▨
Reset	1	1	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-13. Flash Status Register (FSTAT — Special Mode)

CBEIF, PVIOL, and ACCERR are readable and writable, CCIF and BLANK are readable and not writable, remaining bits read 0 and are not writable in normal mode. FAIL is readable and writable in special mode. FAIL must be clear in special mode when starting a command write sequence.

Table 3-15. FSTAT Field Descriptions

Field	Description
7 CBEIF	**Command Buffer Empty Interrupt Flag** — The CBEIF flag indicates that the address, data and command buffers are empty so that a new command write sequence can be started. Writing a 0 to the CBEIF flag has no effect on CBEIF. Writing a 0 to CBEIF after writing an aligned word to the Flash address space, but before CBEIF is cleared, will abort a command write sequence and cause the ACCERR flag to be set. Writing a 0 to CBEIF outside of a command write sequence will not set the ACCERR flag. The CBEIF flag is cleared by writing a 1 to CBEIF. The CBEIF flag is used together with the CBEIE bit in the FCNFG register to generate an interrupt request (see Figure 3-32). 0 Command buffers are full. 1 Command buffers are ready to accept a new command.
6 CCIF	**Command Complete Interrupt Flag** — The CCIF flag indicates that there are no more commands pending. The CCIF flag is cleared when CBEIF is cleared and sets automatically upon completion of all active and pending commands. The CCIF flag does not set when an active commands completes and a pending command is fetched from the command buffer. Writing to the CCIF flag has no effect on CCIF. The CCIF flag is used together with the CCIE bit in the FCNFG register to generate an interrupt request (see Figure 3-32). 0 Command in progress. 1 All commands are completed.
5 PVIOL	**Protection Violation Flag** —The PVIOL flag indicates an attempt was made to program or erase an address in a protected area of the Flash memory during a command write sequence. Writing a 0 to the PVIOL flag has no effect on PVIOL. The PVIOL flag is cleared by writing a 1 to PVIOL. While PVIOL is set, it is not possible to launch a command or start a command write sequence. 0 No protection violation detected. 1 Protection violation has occurred.
4 ACCERR	**Access Error Flag** — The ACCERR flag indicates an illegal access has occurred to the Flash memory caused by either a violation of the command write sequence (see Section 3.4.1.2, “Command Write Sequence”), issuing an illegal Flash command (see Table 3-17), launching the sector erase abort command terminating a sector erase operation early (see Section 3.4.2.6, “Sector Erase Abort Command”) or the execution of a CPU STOP instruction while a command is executing (CCIF = 0). Writing a 0 to the ACCERR flag has no effect on ACCERR. The ACCERR flag is cleared by writing a 1 to ACCERR.While ACCERR is set, it is not possible to launch a command or start a command write sequence. If ACCERR is set by an erase verify operation or a data compress operation, any buffered command will not launch. 0 No access error detected. 1 Access error has occurred.
2 BLANK	**Flag Indicating the Erase Verify Operation Status** — When the CCIF flag is set after completion of an erase verify command, the BLANK flag indicates the result of the erase verify operation. The BLANK flag is cleared by the Flash module when CBEIF is cleared as part of a new valid command write sequence. Writing to the BLANK flag has no effect on BLANK. 0 Flash block verified as not erased. 1 Flash block verified as erased.
1 FAIL	**Flag Indicating a Failed Flash Operation** — The FAIL flag will set if the erase verify operation fails (selected Flash block verified as not erased). Writing a 0 to the FAIL flag has no effect on FAIL. The FAIL flag is cleared by writing a 1 to FAIL. 0 Flash operation completed without error. 1 Flash operation failed.

3.3.2.7 Flash Command Register (FCMD)

The FCMD register is the Flash command register.

Module Base + 0x0006

	7	6	5	4	3	2	1	0
R	0	CMDB						
W	▨							
Reset	1	1	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-14. Flash Command Register (FCMD)

All CMDB bits are readable and writable during a command write sequence while bit 7 reads 0 and is not writable.

Table 3-16. FCMD Field Descriptions

Field	Description
6:0 CMDB[6:0]	**Flash Command** — Valid Flash commands are shown in Table 3-17. Writing any command other than those listed in Table 3-17 sets the ACCERR flag in the FSTAT register.

Table 3-17. Valid Flash Command List

CMDB[6:0]	NVM Command
0x05	Erase Verify
0x06	Data Compress
0x20	Word Program
0x40	Sector Erase
0x41	Mass Erase
0x47	Sector Erase Abort

3.3.2.8 Flash Control Register (FCTL)

The FCTL register is the Flash control register.

Module Base + 0x0007

	7	6	5	4	3	2	1	0
R	0	NV6	NV5	NV4	NV3	NV2	NV1	NV0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	F	F	F	F	F	F	F

▨ = Unimplemented or Reserved

Figure 3-15. Flash Control Register (FCTL)

All bits in the FCTL register are readable but are not writable.

The FCTL NV bits are loaded from the Flash nonvolatile byte located at global address 0x7F_FF0E during the reset sequence, indicated by F in Figure 3-15.

Table 3-18. FCTL Field Descriptions

Field	Description
6:0 NV[6:0]	**Nonvolatile Bits** — The NV[6:0] bits are available as nonvolatile bits. Refer to the Device User Guide for proper use of the NV bits.

3.3.2.9 Flash Address Registers (FADDR)

The FADDRHI and FADDRLO registers are the Flash address registers.

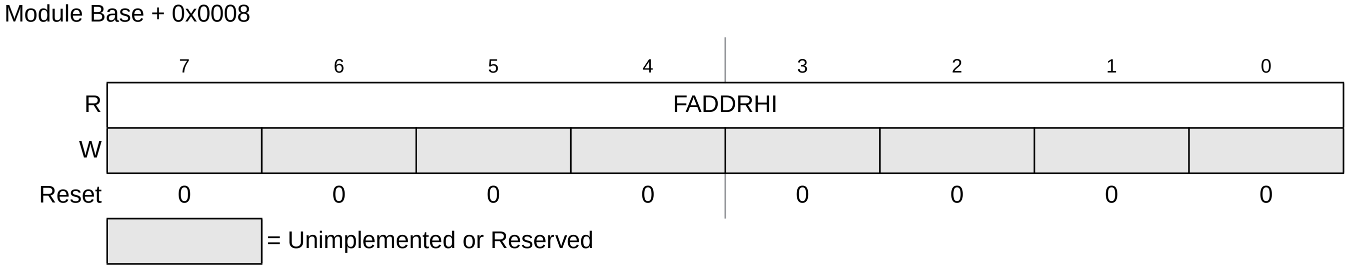


Figure 3-16. Flash Address High Register (FADDRHI)

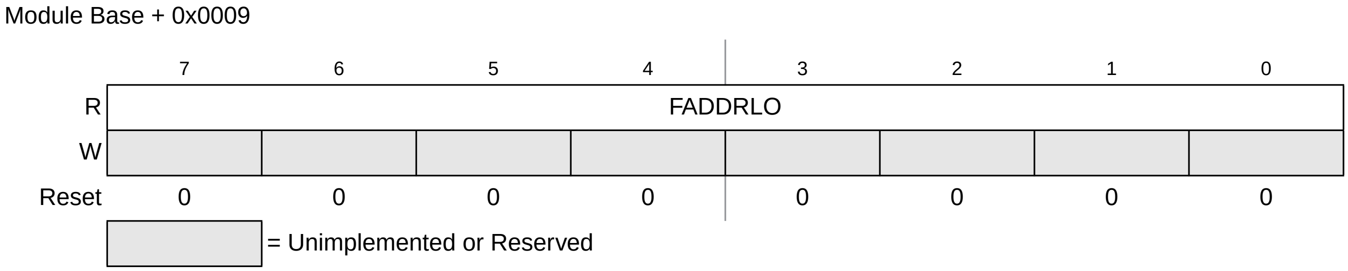


Figure 3-17. Flash Address Low Register (FADDRLO)

All FADDRHI and FADDRLO bits are readable but are not writable. After an array write as part of a command write sequence, the FADDR registers will contain the mapped MCU address written.

3.3.2.10 Flash Data Registers (FDATA)

The FDATAHI and FDATALO registers are the Flash data registers.

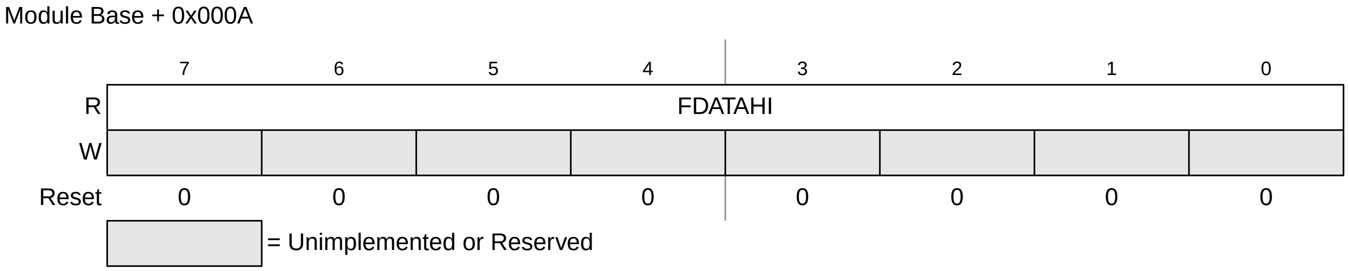


Figure 3-18. Flash Data High Register (FDATAHI)

Module Base + 0x000B

	7	6	5	4	3	2	1	0
R	FDATALO							
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-19. Flash Data Low Register (FDATALO)

All FDATAHI and FDATALO bits are readable but are not writable. At the completion of a data compress operation, the resulting 16-bit signature is stored in the FDATA registers. The data compression signature is readable in the FDATA registers until a new command write sequence is started.

3.3.2.11 RESERVED1

This register is reserved for factory testing and is not accessible.

Module Base + 0x000C

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-20. RESERVED1

All bits read 0 and are not writable.

3.3.2.12 RESERVED2

This register is reserved for factory testing and is not accessible.

Module Base + 0x000D

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-21. RESERVED2

All bits read 0 and are not writable.

3.3.2.13 RESERVED3

This register is reserved for factory testing and is not accessible.

Module Base + 0x000E

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-22. RESERVED3

All bits read 0 and are not writable.

3.3.2.14 RESERVED4

This register is reserved for factory testing and is not accessible.

Module Base + 0x000F

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-23. RESERVED4

All bits read 0 and are not writable.

3.4 Functional Description

3.4.1 Flash Command Operations

Write operations are used to execute program, erase, erase verify, erase abort, and data compress algorithms described in this section. The program and erase algorithms are controlled by a state machine whose timebase, FCLK, is derived from the oscillator clock via a programmable divider. The command register, as well as the associated address and data registers, operate as a buffer and a register (2-stage FIFO) so that a second command along with the necessary data and address can be stored to the buffer while the first command is still in progress. This pipelined operation allows a time optimization when programming more than one word on a specific row in the Flash block as the high voltage generation can be kept active in between two programming commands. The pipelined operation also allows a simplification of command launching. Buffer empty as well as command completion are signalled by flags in the Flash status register with corresponding interrupts generated, if enabled.

The next sections describe:

1. How to write the FCLKDIV register
2. Command write sequences to program, erase, erase verify, erase abort, and data compress operations on the Flash memory

3. Valid Flash commands
4. Effects resulting from illegal Flash command write sequences or aborting Flash operations

3.4.1.1 Writing the FCLKDIV Register

Prior to issuing any Flash command after a reset, the user is required to write the FCLKDIV register to divide the oscillator clock down to within the 150 kHz to 200 kHz range. Since the program and erase timings are also a function of the bus clock, the FCLKDIV determination must take this information into account.

If we define:

- FCLK as the clock of the Flash timing control block
- Tbus as the period of the bus clock
- INT(x) as taking the integer part of x (e.g. INT(4.323) = 4)

then FCLKDIV register bits PRDIV8 and FDIV[5:0] are to be set as described in Figure 3-24.

For example, if the oscillator clock frequency is 950kHz and the bus clock frequency is 10MHz, FCLKDIV bits FDIV[5:0] should be set to 0x04 (000100) and bit PRDIV8 set to 0. The resulting FCLK frequency is then 190kHz. As a result, the Flash program and erase algorithm timings are increased over the optimum target by:

$$(200 - 190) / 200 \times 100 = 5\%$$

If the oscillator clock frequency is 16MHz and the bus clock frequency is 40MHz, FCLKDIV bits FDIV[5:0] should be set to 0x0A (001010) and bit PRDIV8 set to 1. The resulting FCLK frequency is then 182kHz. In this case, the Flash program and erase algorithm timings are increased over the optimum target by:

$$(200 - 182) / 200 \times 100 = 9\%$$

CAUTION

> Program and erase command execution time will increase proportionally with the period of FCLK. Because of the impact of clock synchronization on the accuracy of the functional timings, programming or erasing the Flash memory cannot be performed if the bus clock runs at less than 1 MHz. Programming or erasing the Flash memory with FCLK < 150 kHz should be avoided. Setting FCLKDIV to a value such that FCLK < 150 kHz can destroy the Flash memory due to overstress. Setting FCLKDIV to a value such that (1/FCLK+Tbus) < 5µs can result in incomplete programming or erasure of the Flash memory cells.

If the FCLKDIV register is written, the FDIVLD bit is set automatically. If the FDIVLD bit is 0, the FCLKDIV register has not been written since the last reset. If the FCLKDIV register has not been written to, the Flash command loaded during a command write sequence will not execute and the ACCERR flag in the FSTAT register will set.

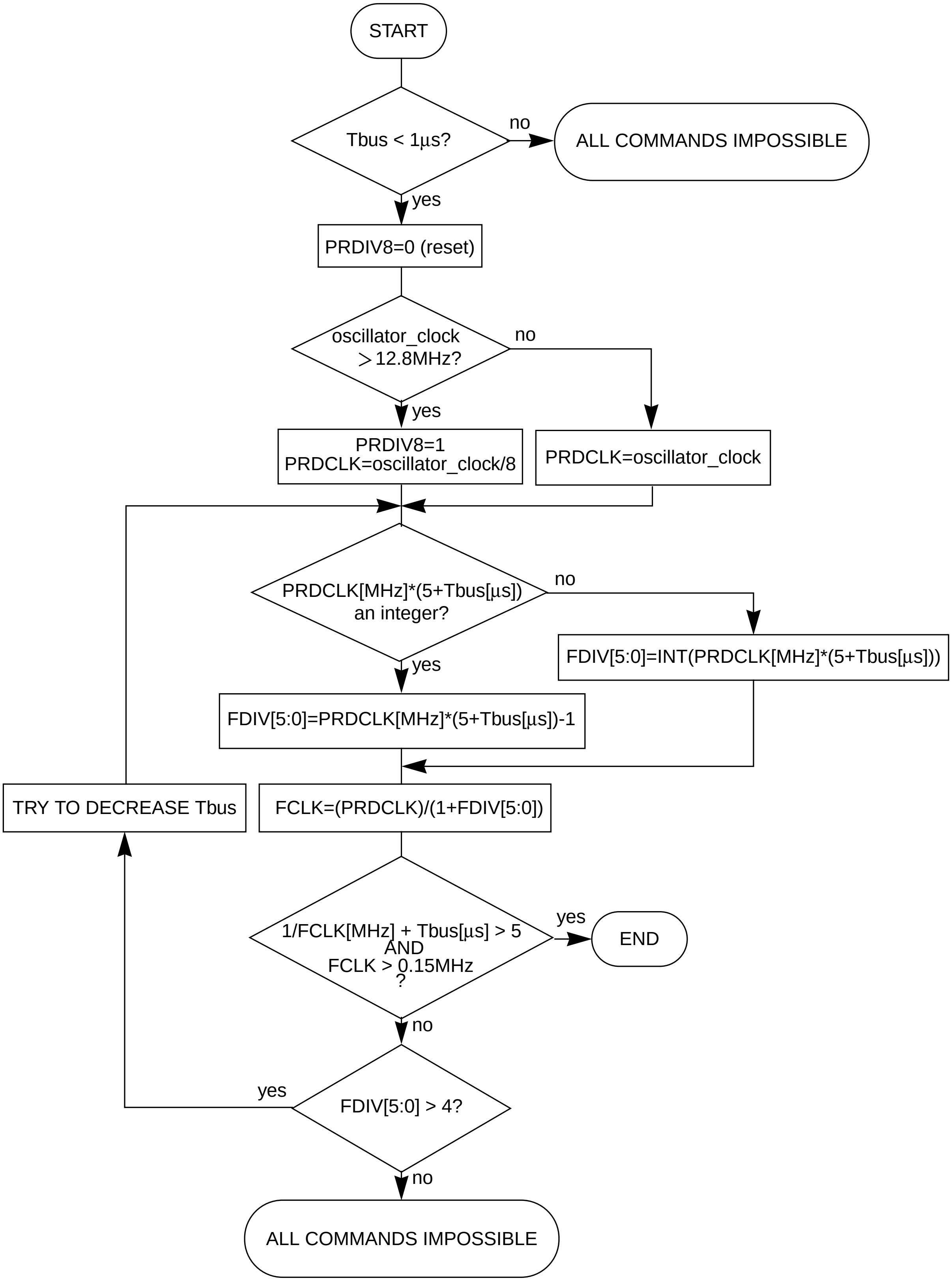


Figure 3-24. Determination Procedure for PRDIV8 and FDIV Bits

3.4.1.2 Command Write Sequence

The Flash command controller is used to supervise the command write sequence to execute program, erase, erase verify, erase abort, and data compress algorithms.

Before starting a command write sequence, the ACCERR and PVIOL flags in the FSTAT register must be clear (see Section 3.3.2.6, "Flash Status Register (FSTAT)") and the CBEIF flag should be tested to determine the state of the address, data and command buffers. If the CBEIF flag is set, indicating the buffers are empty, a new command write sequence can be started. If the CBEIF flag is clear, indicating the buffers are not available, a new command write sequence will overwrite the contents of the address, data and command buffers.

A command write sequence consists of three steps which must be strictly adhered to with writes to the Flash module not permitted between the steps. However, Flash register and array reads are allowed during a command write sequence. The basic command write sequence is as follows:

1. Write to a valid address in the Flash memory. Addresses in multiple Flash blocks can be written to as long as the location is at the same relative address in each available Flash block. Multiple addresses must be written in Flash block order starting with the lower Flash block.
2. Write a valid command to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the command.

The address written in step 1 will be stored in the FADDR registers and the data will be stored in the FDATA registers. If the CBEIF flag in the FSTAT register is clear when the first Flash array write occurs, the contents of the address and data buffers will be overwritten and the CBEIF flag will be set. When the CBEIF flag is cleared, the CCIF flag is cleared on the same bus cycle by the Flash command controller indicating that the command was successfully launched. For all command write sequences except data compress and sector erase abort, the CBEIF flag will set four bus cycles after the CCIF flag is cleared indicating that the address, data, and command buffers are ready for a new command write sequence to begin. For data compress and sector erase abort operations, the CBEIF flag will remain clear until the operation completes. Except for the sector erase abort command, a buffered command will wait for the active operation to be completed before being launched. The sector erase abort command is launched when the CBEIF flag is cleared as part of a sector erase abort command write sequence. Once a command is launched, the completion of the command operation is indicated by the setting of the CCIF flag in the FSTAT register. The CCIF flag will set upon completion of all active and buffered commands.

3.4.2 Flash Commands

Table 3-19 summarizes the valid Flash commands along with the effects of the commands on the Flash block.

Table 3-19. Flash Command Description

FCMDB	NVM Command	Function on Flash Memory
0x05	Erase Verify	Verify all memory bytes in the Flash block are erased. If the Flash block is erased, the BLANK flag in the FSTAT register will set upon command completion.

Table 3-19. Flash Command Description

FCMDB	NVM Command	Function on Flash Memory
0x06	Data Compress	Compress data from a selected portion of the Flash block. The resulting signature is stored in the FDATA register.
0x20	Program	Program a word (two bytes) in the Flash block.
0x40	Sector Erase	Erase all memory bytes in a sector of the Flash block.
0x41	Mass Erase	Erase all memory bytes in the Flash block. A mass erase of the full Flash block is only possible when FPLDIS, FPHDIS and FPOPEN bits in the FPROT register are set prior to launching the command.
0x47	Sector Erase Abort	Abort the sector erase operation. The sector erase operation will terminate according to a set procedure. The Flash sector should not be considered erased if the ACCERR flag is set upon command completion.

CAUTION

A Flash word must be in the erased state before being programmed. Cumulative programming of bits within a Flash word is not allowed.

3.4.2.1 Erase Verify Command

The erase verify operation will verify that a Flash block is erased.

An example flow to execute the erase verify operation is shown in Figure 3-25. The erase verify command write sequence is as follows:

1. Write an aligned word to a valid address in the Flash array memory to start the command write sequence for the erase verify command. The address and data written will be ignored. Multiple Flash blocks can be simultaneously erase verified by writing to the same relative address in each Flash block.
2. Write the erase verify command, 0x05, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the erase verify command.

After launching the erase verify command, the CCIF flag in the FSTAT register will set after the operation has completed unless a new command write sequence has been buffered. The number of bus cycles required to execute the erase verify operation is equal to the number of addresses in a Flash block plus 14 bus cycles as measured from the time the CBEIF flag is cleared until the CCIF flag is set. Upon completion of the erase verify operation, the BLANK flag in the FSTAT register will be set if all addresses in the selected Flash blocks are verified to be erased. If any address in a selected Flash block is not erased, the erase verify operation will terminate and the BLANK flag in the FSTAT register will remain clear. The MRDS bits in the FTSTMOD register will determine the sense-amp margin setting during the erase verify operation.

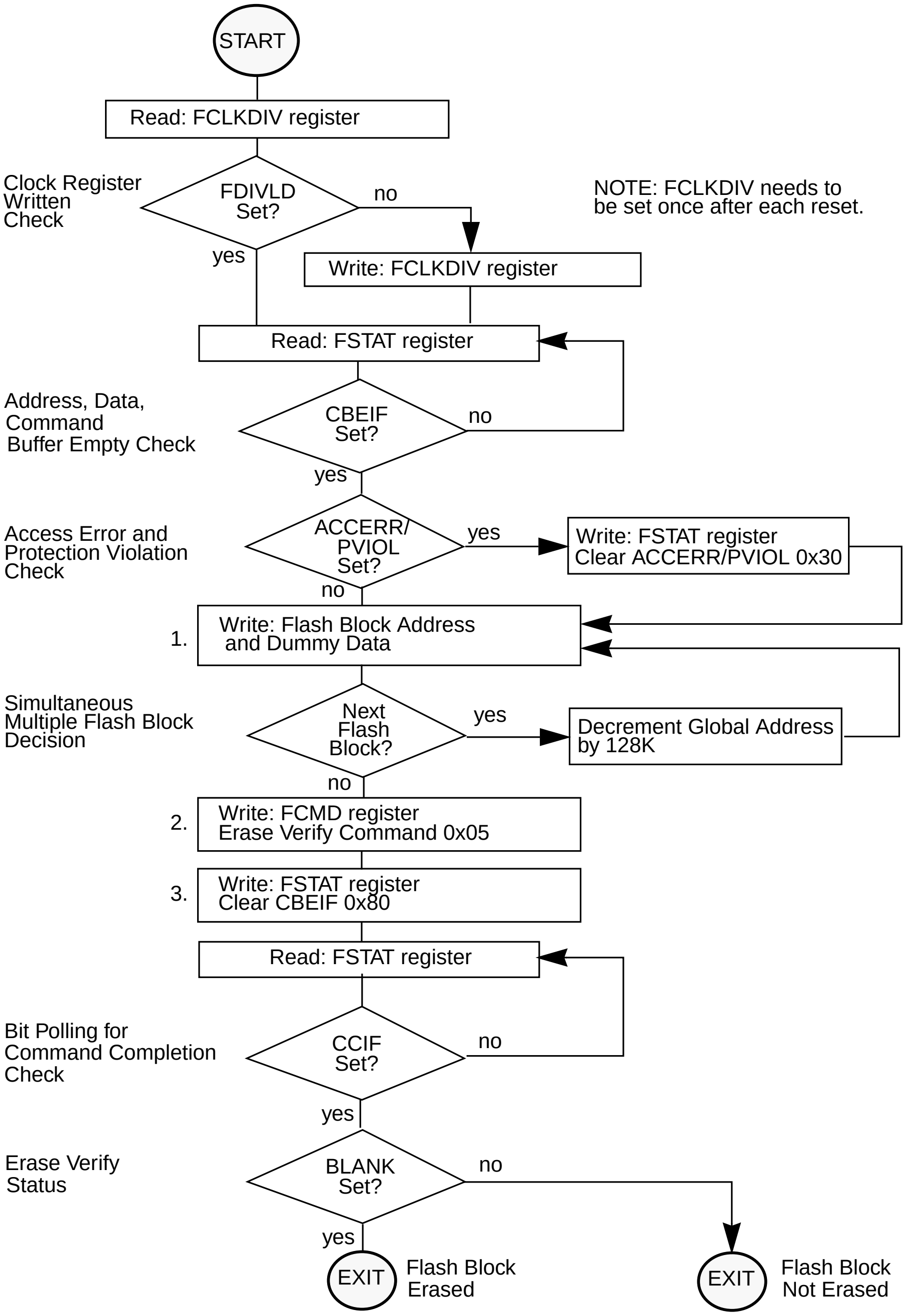


Figure 3-25. Example Erase Verify Command Flow

3.4.2.2 Data Compress Command

The data compress operation will check Flash code integrity by compressing data from a selected portion of the Flash memory into a signature analyzer.

An example flow to execute the data compress operation is shown in Figure 3-26. The data compress command write sequence is as follows:

1. Write an aligned word to a valid address in the Flash array memory to start the command write sequence for the data compress command. The address written determines the starting address for the data compress operation and the data written determines the number of consecutive words to compress with an upper limit of 16,384. Multiple Flash blocks can be simultaneously compressed by writing to the same relative address in each Flash block. If more than one Flash block is written to in this step, the first data written will determine the number of consecutive words to compress in each selected Flash block.
2. Write the data compress command, 0x06, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the data compress command.

After launching the data compress command, the CCIF flag in the FSTAT register will set after the data compress operation has completed. The number of bus cycles required to execute the data compress operation is equal to two times the number of consecutive words to compress plus the number of Flash blocks simultaneously compressed plus 18 bus cycles as measured from the time the CBEIF flag is cleared until the CCIF flag is set. Once the CCIF flag is set, the signature generated by the data compress operation is available in the FDATA registers. The signature in the FDATA registers can be compared to the expected signature to determine the integrity of the selected data stored in the selected Flash memory. If the last address of a Flash block is reached during the data compress operation, data compression will continue with the starting address of the same Flash block. The MRDS bits in the FTSTMOD register will determine the sense-amp margin setting during the data compress operation.

NOTE

Since the FDATA registers (or data buffer) are written to as part of the data compress operation, a command write sequence is not allowed to be buffered behind a data compress command write sequence. The CBEIF flag will not set after launching the data compress command to indicate that a command should not be buffered behind it. If an attempt is made to start a new command write sequence with a data compress operation active, the ACCERR flag in the FSTAT register will be set. A new command write sequence should only be started after reading the signature stored in the FDATA registers.

In order to take corrective action, it is recommended that the data compress command be executed on a Flash sector or subset of a Flash sector. If the data compress operation on a Flash sector returns an invalid signature, the Flash sector should be erased using the sector erase command and then reprogrammed using the program command.

The data compress command can be used to verify that a sector or sequential set of sectors are erased.

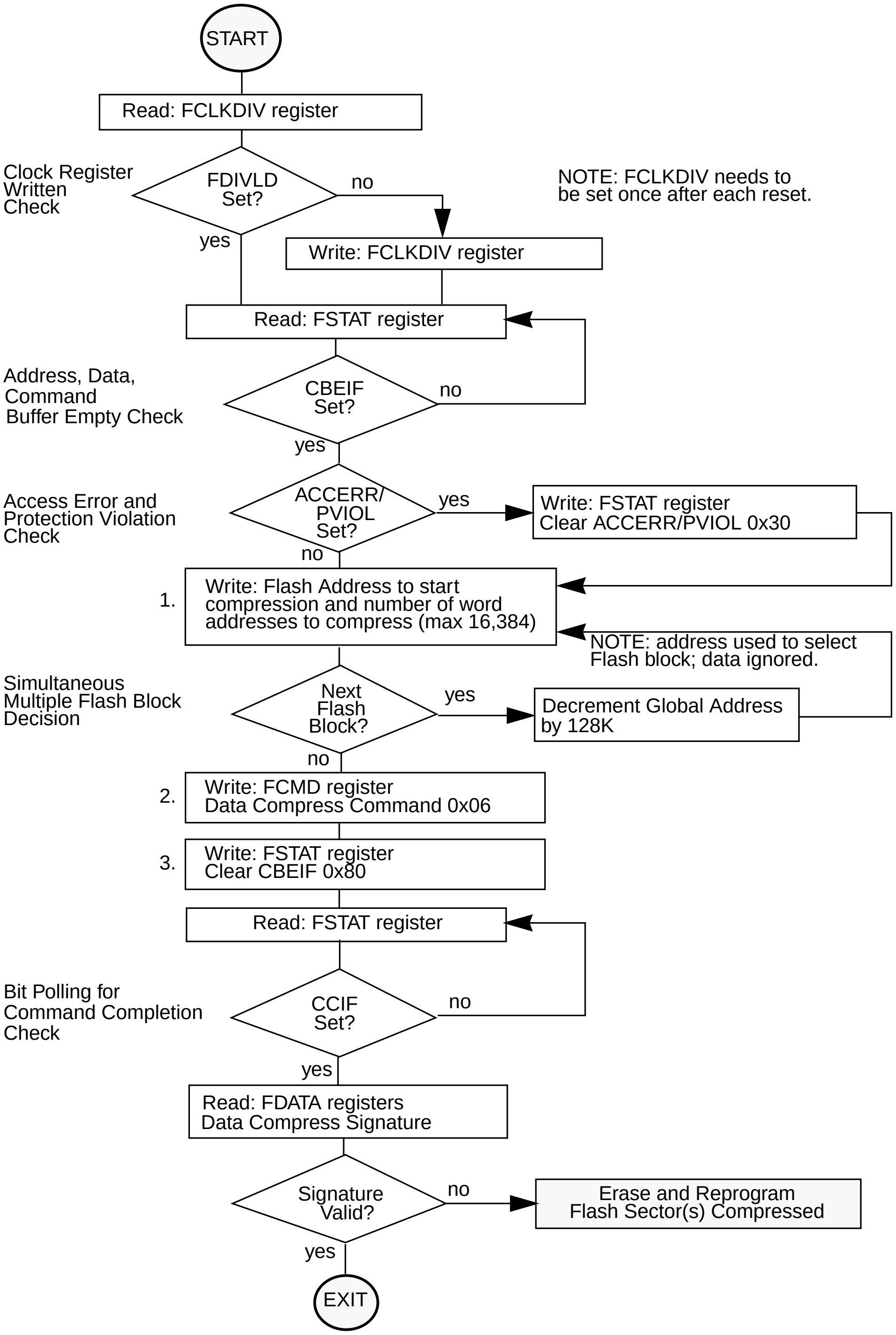


Figure 3-26. Example Data Compress Command Flow

3.4.2.2.1 Data Compress Operation

The Flash module contains a 16-bit multiple-input signature register (MISR) for each Flash block to generate a 16-bit signature based on selected Flash array data. If multiple Flash blocks are selected for simultaneous compression, then the signature from each Flash block is further compressed to generate a single 16-bit signature. The final 16-bit signature, found in the FDATA registers after the data compress operation has completed, is based on the following logic equation which is executed on every data compression cycle during the operation:

MISR[15:0] = {MISR[14:0], ^MISR[15,4,2,1]} ^ DATA[15:0] ***Eqn. 3-1***

where MISR is the content of the internal signature register associated with each Flash block and DATA is the data to be compressed as shown in Figure 3-27.

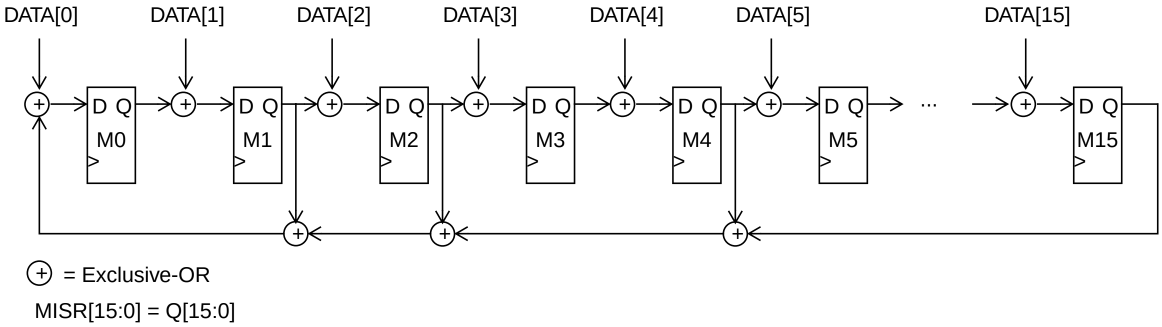


Figure 3-27. 16-Bit MISR Diagram

During the data compress operation, the following steps are executed:

1. MISR for each Flash block is reset to 0xFFFF.
2. Initialized DATA equal to 0xFFFF is compressed into the MISR for each selected Flash block which results in the MISR containing 0x0001.
3. DATA equal to the selected Flash array data range is read and compressed into the MISR for each selected Flash block with addresses incrementing.
4. DATA equal to the selected Flash array data range is read and compressed into the MISR for each selected Flash block with addresses decrementing.
5. If Flash block 0 is selected for compression, DATA equal to the contents of the MISR for Flash block 0 is compressed into the MISR for Flash block 0. If data in Flash block 0 was not selected for compression, the MISR for Flash block 0 contains 0xFFFF.
6. If Flash block 1 is selected for compression, DATA equal to the contents of the MISR for Flash block 1 is compressed into the MISR for Flash block 0.
7. If Flash block 2 is selected for compression, DATA equal to the contents of the MISR for Flash block 2 is compressed into the MISR for Flash block 0.
8. If Flash block 3 is selected for compression, DATA equal to the contents of the MISR for Flash block 3 is compressed into the MISR for Flash block 0.
9. If Flash block 4 is selected for compression, DATA equal to the contents of the MISR for Flash block 4 is compressed into the MISR for Flash block 0.

10. If Flash block 5 is selected for compression, DATA equal to the contents of the MISR for Flash block 5 is compressed into the MISR for Flash block 0.
11. If Flash block 6 is selected for compression, DATA equal to the contents of the MISR for Flash block 6 is compressed into the MISR for Flash block 0.
12. If Flash block 7 is selected for compression, DATA equal to the contents of the MISR for Flash block 7 is compressed into the MISR for Flash block 0.
13. The contents of the MISR for Flash block 0 are written to the FDATA registers.

3.4.2.3 Program Command

The program operation will program a previously erased word in the Flash memory using an embedded algorithm.

An example flow to execute the program operation is shown in Figure 3-28. The program command write sequence is as follows:

1. Write an aligned word to a valid address in the Flash array memory to start the command write sequence for the program command. The data written will be programmed to the address written. Multiple Flash blocks can be simultaneously programmed by writing to the same relative address in each Flash block.
2. Write the program command, 0x20, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the program command.

If a word to be programmed is in a protected area of the Flash block, the PVIOL flag in the FSTAT register will set and the program command will not launch. Once the program command has successfully launched, the CCIF flag in the FSTAT register will set after the program operation has completed unless a new command write sequence has been buffered. By executing a new program command write sequence on sequential words after the CBEIF flag in the FSTAT register has been set, up to 55% faster programming time per word can be effectively achieved than by waiting for the CCIF flag to set after each program operation.

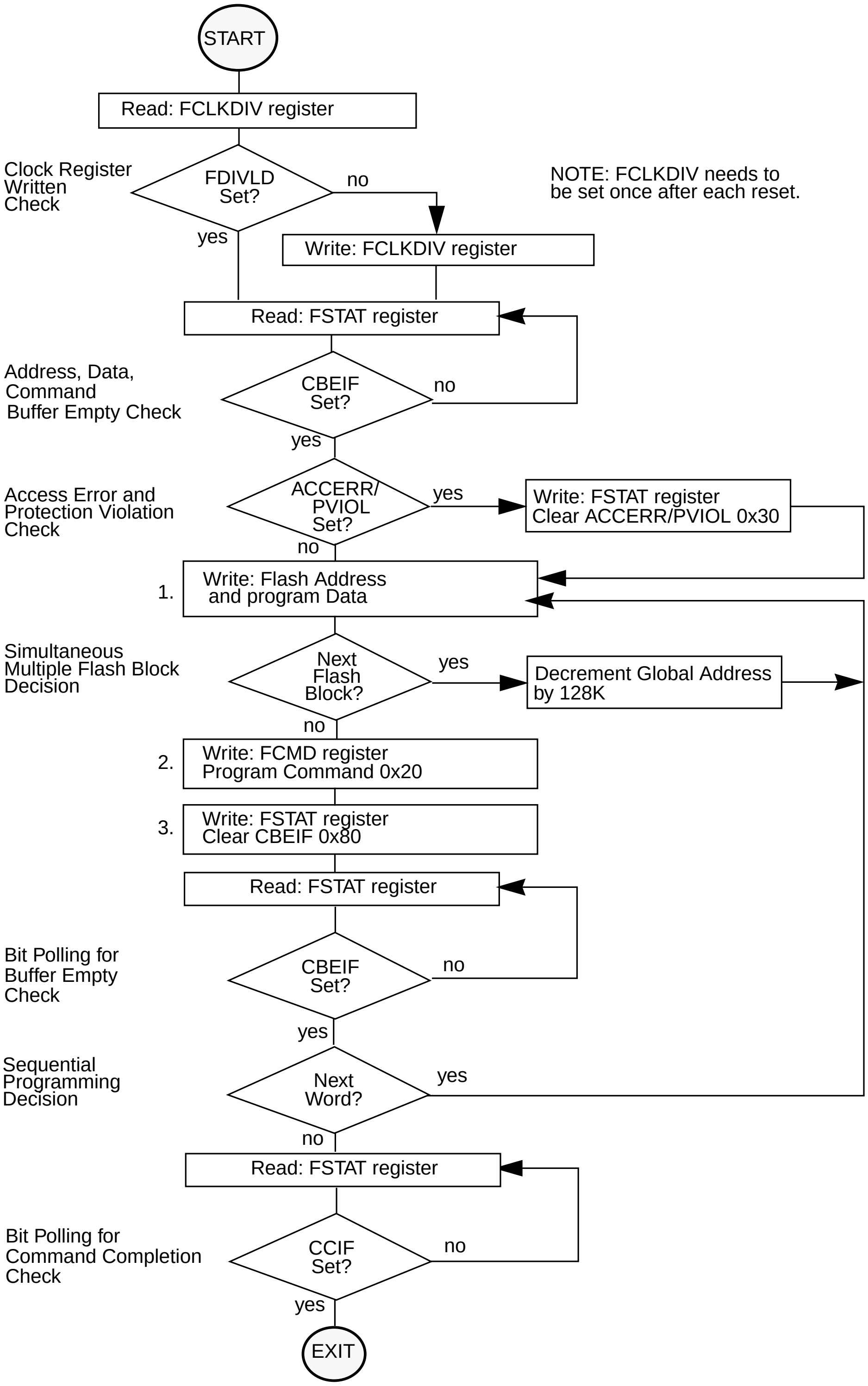


Figure 3-28. Example Program Command Flow

3.4.2.4 Sector Erase Command

The sector erase operation will erase all addresses in a 1 Kbyte sector of Flash memory using an embedded algorithm.

An example flow to execute the sector erase operation is shown in Figure 3-29. The sector erase command write sequence is as follows:

1. Write an aligned word to a valid address in the Flash array memory to start the command write sequence for the sector erase command. The Flash address written determines the sector to be erased while global address bits [9:0] and the data written are ignored. Multiple Flash sectors can be simultaneously erased by writing to the same relative address in each Flash block.
2. Write the sector erase command, 0x40, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the sector erase command.

If a Flash sector to be erased is in a protected area of the Flash block, the PVIOL flag in the FSTAT register will set and the sector erase command will not launch. Once the sector erase command has successfully launched, the CCIF flag in the FSTAT register will set after the sector erase operation has completed unless a new command write sequence has been buffered.

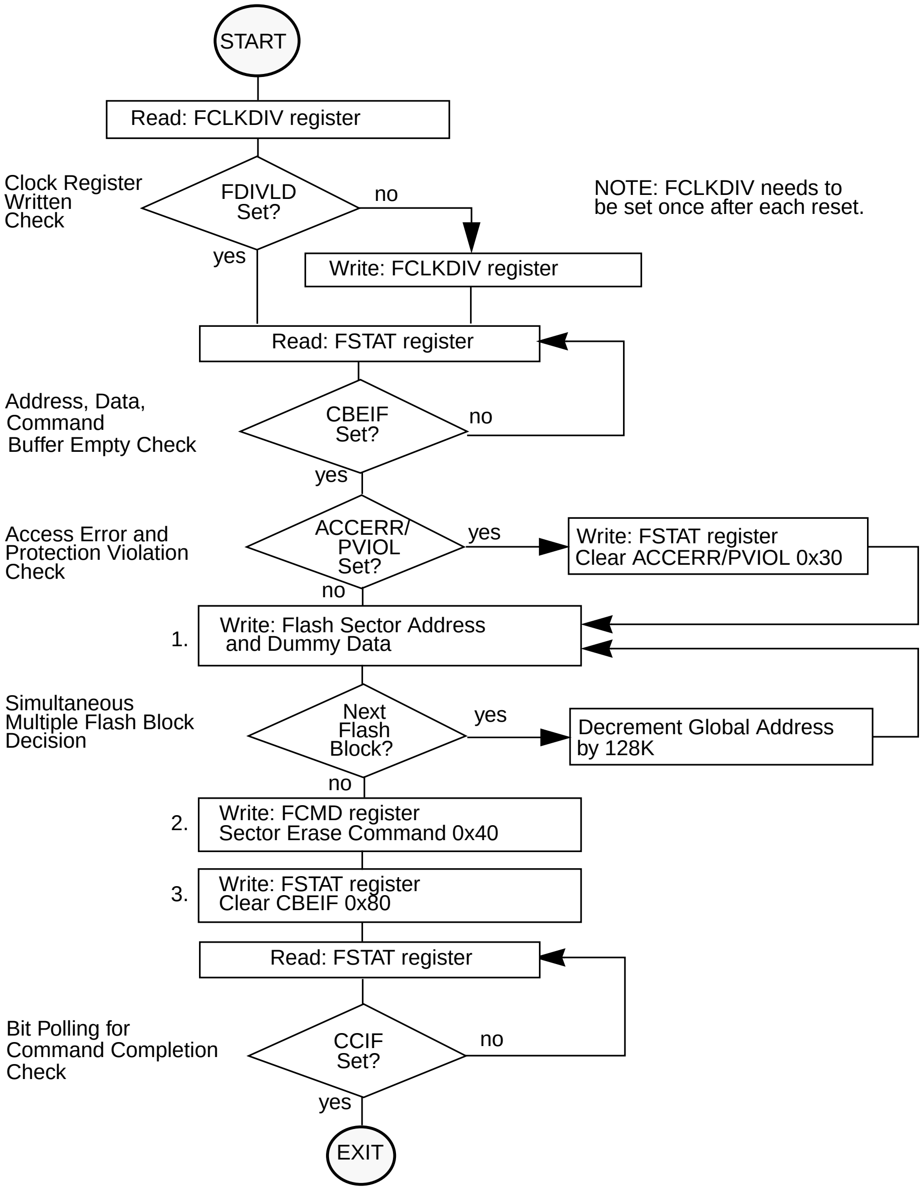


Figure 3-29. Example Sector Erase Command Flow

3.4.2.5 Mass Erase Command

The mass erase operation will erase all addresses in a Flash block using an embedded algorithm.

An example flow to execute the mass erase operation is shown in Figure 3-30. The mass erase command write sequence is as follows:

1. Write an aligned word to any valid address in a Flash block to start the command write sequence for the mass erase command. The data written will be ignored while the address written determines which Flash block is erased. Multiple Flash blocks can be simultaneously mass erased by writing to the same relative address in each Flash block.
2. Write the mass erase command, 0x41, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the mass erase command.

If a Flash block to be erased contains any protected area, the PVIOL flag in the FSTAT register will set and the mass erase command will not launch. Once the mass erase command has successfully launched, the CCIF flag in the FSTAT register will set after the mass erase operation has completed unless a new command write sequence has been buffered.

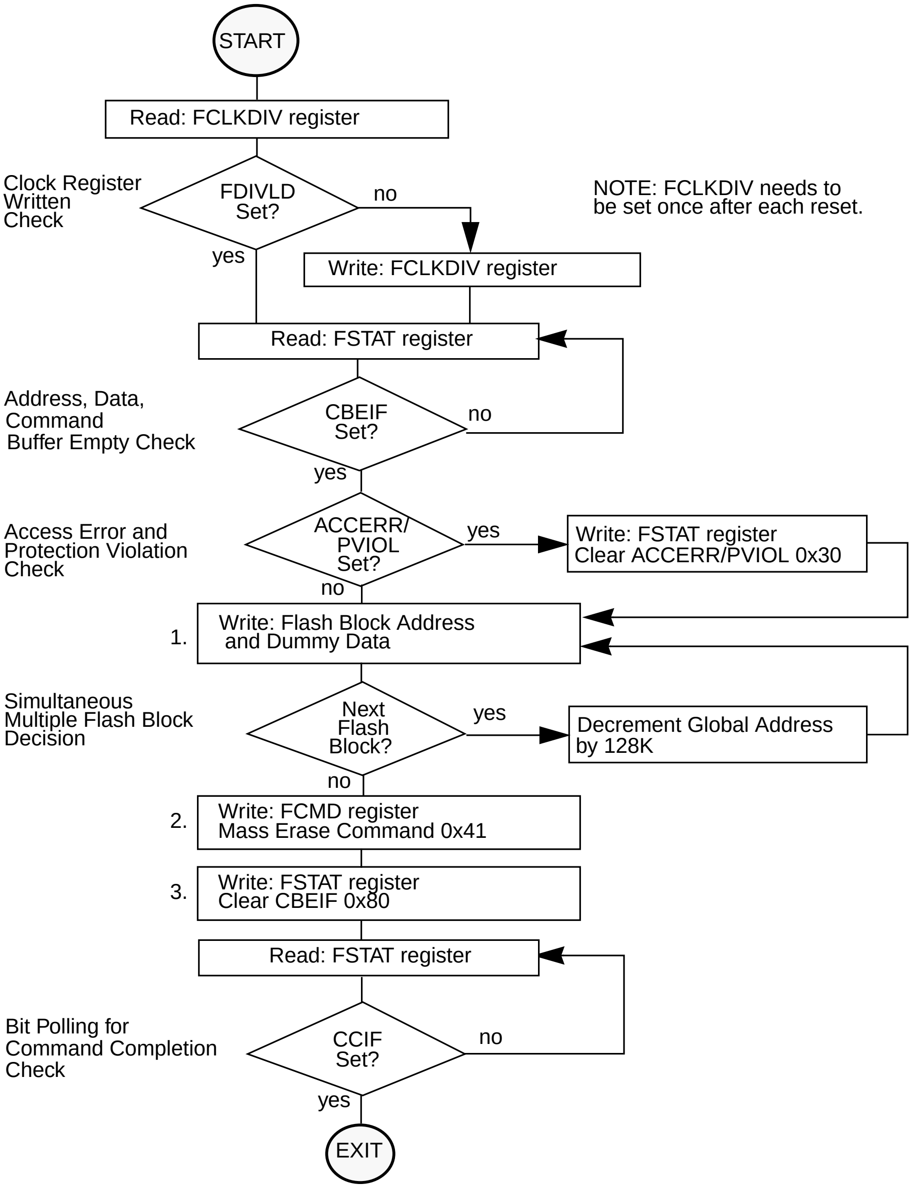


Figure 3-30. Example Mass Erase Command Flow

3.4.2.6 Sector Erase Abort Command

The sector erase abort operation will terminate the active sector erase operation so that other sectors in a Flash block are available for read and program operations without waiting for the sector erase operation to complete.

An example flow to execute the sector erase abort operation is shown in Figure 3-31. The sector erase abort command write sequence is as follows:

1. Write an aligned word to any valid address in the Flash array memory to start the command write sequence for the sector erase abort command. The address and data written are ignored.
2. Write the sector erase abort command, 0x47, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the sector erase abort command.

If the sector erase abort command is launched resulting in the early termination of an active sector erase operation, the ACCERR flag will set once the operation completes as indicated by the CCIF flag being set. The ACCERR flag sets to inform the user that the Flash sector may not be fully erased and a new sector erase command must be launched before programming any location in that specific sector. If the sector erase abort command is launched but the active sector erase operation completes normally, the ACCERR flag will not set upon completion of the operation as indicated by the CCIF flag being set. Therefore, if the ACCERR flag is not set after the sector erase abort command has completed, a Flash sector being erased when the abort command was launched will be fully erased. The maximum number of cycles required to abort a sector erase operation is equal to four FCLK periods (see Section 3.4.1.1, "Writing the FCLKDIV Register") plus five bus cycles as measured from the time the CBEIF flag is cleared until the CCIF flag is set. If sectors in multiple Flash blocks are being simultaneously erased, the sector erase abort operation will be applied to all active Flash blocks without writing to each Flash block in the sector erase abort command write sequence.

NOTE

Since the ACCERR bit in the FSTAT register may be set at the completion of the sector erase abort operation, a command write sequence is not allowed to be buffered behind a sector erase abort command write sequence. The CBEIF flag will not set after launching the sector erase abort command to indicate that a command should not be buffered behind it. If an attempt is made to start a new command write sequence with a sector erase abort operation active, the ACCERR flag in the FSTAT register will be set. A new command write sequence may be started after clearing the ACCERR flag, if set.

NOTE

The sector erase abort command should be used sparingly since a sector erase operation that is aborted counts as a complete program/erase cycle.

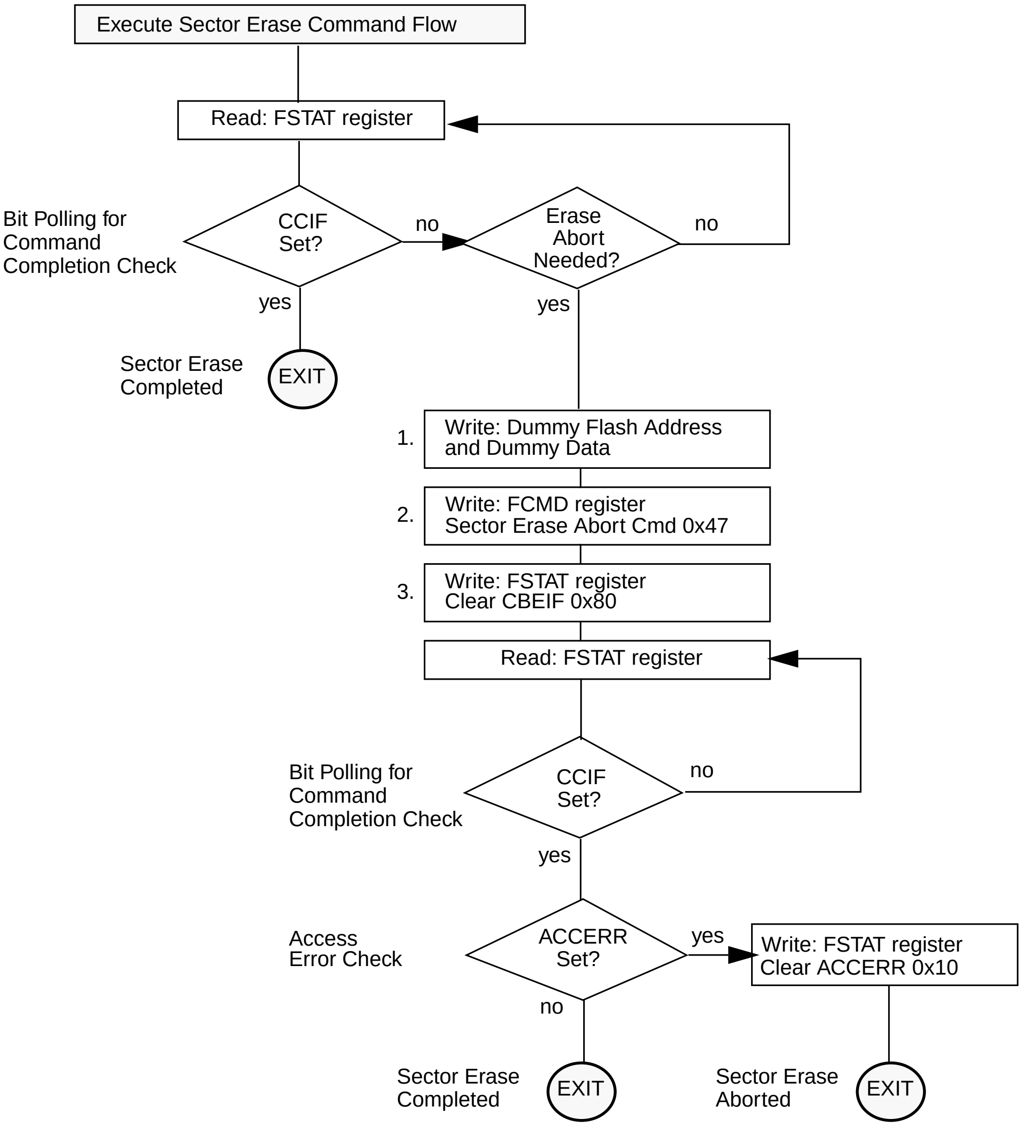


Figure 3-31. Example Sector Erase Abort Command Flow

3.4.3 Illegal Flash Operations

The ACCERR flag will be set during the command write sequence if any of the following illegal steps are performed, causing the command write sequence to immediately abort:

1. Writing to a Flash address before initializing the FCLKDIV register.
2. Writing a byte or misaligned word to a valid Flash address.
3. Starting a command write sequence while a data compress operation is active.
4. Starting a command write sequence while a sector erase abort operation is active.
5. Writing a Flash address in step 1 of a command write sequence that is not the same relative address as the first one written in the same command write sequence.
6. Writing to any Flash register other than FCMD after writing to a Flash address.
7. Writing a second command to the FCMD register in the same command write sequence.
8. Writing an invalid command to the FCMD register.
9. When security is enabled, writing a command other than mass erase to the FCMD register when the write originates from a non-secure memory location or from the Background Debug Mode.
10. Writing to a Flash address after writing to the FCMD register.
11. Writing to any Flash register other than FSTAT (to clear CBEIF) after writing to the FCMD register.
12. Writing a 0 to the CBEIF flag in the FSTAT register to abort a command write sequence.

The ACCERR flag will not be set if any Flash register is read during a valid command write sequence.

The ACCERR flag will also be set if any of the following events occur:

1. Launching the sector erase abort command while a sector erase operation is active which results in the early termination of the sector erase operation (see Section 3.4.2.6, “Sector Erase Abort Command”).
2. The MCU enters stop mode and a program or erase operation is in progress. The operation is aborted immediately and any pending command is purged (see Section 3.5.2, “Stop Mode”).

If the Flash memory is read during execution of an algorithm (CCIF = 0), the read operation will return invalid data and the ACCERR flag will not be set.

If the ACCERR flag is set in the FSTAT register, the user must clear the ACCERR flag before starting another command write sequence (see Section 3.3.2.6, “Flash Status Register (FSTAT)”).

The PVIOL flag will be set after the command is written to the FCMD register during a command write sequence if any of the following illegal operations are attempted, causing the command write sequence to immediately abort:

1. Writing the program command if an address written in the command write sequence was in a protected area of the Flash memory
2. Writing the sector erase command if an address written in the command write sequence was in a protected area of the Flash memory
3. Writing the mass erase command to a Flash block while any Flash protection is enabled in the block

If the PVIOL flag is set in the FSTAT register, the user must clear the PVIOL flag before starting another command write sequence (see Section 3.3.2.6, “Flash Status Register (FSTAT)”).

3.5 Operating Modes

3.5.1 Wait Mode

If a command is active (CCIF = 0) when the MCU enters wait mode, the active command and any buffered command will be completed.

The Flash module can recover the MCU from wait mode if the CBEIF and CCIF interrupts are enabled (see Section 3.8, “Interrupts”).

3.5.2 Stop Mode

If a command is active (CCIF = 0) when the MCU enters stop mode, the operation will be aborted and, if the operation is program or erase, the Flash array data being programmed or erased may be corrupted and the CCIF and ACCERR flags will be set. If active, the high voltage circuitry to the Flash memory will immediately be switched off when entering stop mode. Upon exit from stop mode, the CBEIF flag is set and any buffered command will not be launched. The ACCERR flag must be cleared before starting a command write sequence (see Section 3.4.1.2, “Command Write Sequence”).

NOTE

As active commands are immediately aborted when the MCU enters stop mode, it is strongly recommended that the user does not use the STOP instruction during program or erase operations.

3.5.3 Background Debug Mode

In background debug mode (BDM), the FPROT register is writable. If the MCU is unsecured, then all Flash commands listed in Table 3-19 can be executed. If the MCU is secured and is in special single chip mode, only mass erase can be executed.

3.6 Flash Module Security

The Flash module provides the necessary security information to the MCU. After each reset, the Flash module determines the security state of the MCU as defined in Section 3.3.2.2, “Flash Security Register (FSEC)”.

The contents of the Flash security byte at 0x7F_FF0F in the Flash Configuration Field must be changed directly by programming 0x7F_FF0F when the MCU is unsecured and the higher address sector is unprotected. If the Flash security byte is left in a secured state, any reset will cause the MCU to initialize to a secure operating mode.

3.6.1 Unsecuring the MCU using Backdoor Key Access

The MCU may be unsecured by using the backdoor key access feature which requires knowledge of the contents of the backdoor keys (four 16-bit words programmed at addresses 0x7F_FF00–0x7F_FF07). If the KEYEN[1:0] bits are in the enabled state (see Section 3.3.2.2, "Flash Security Register (FSEC)") and the KEYACC bit is set, a write to a backdoor key address in the Flash memory triggers a comparison between the written data and the backdoor key data stored in the Flash memory. If all four words of data are written to the correct addresses in the correct order and the data matches the backdoor keys stored in the Flash memory, the MCU will be unsecured. The data must be written to the backdoor keys sequentially starting with 0x7F_FF00–1 and ending with 0x7F_FF06–7. 0x0000 and 0xFFFF are not permitted as backdoor keys. While the KEYACC bit is set, reads of the Flash memory will return invalid data.

The user code stored in the Flash memory must have a method of receiving the backdoor keys from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If the KEYEN[1:0] bits are in the enabled state (see Section 3.3.2.2, "Flash Security Register (FSEC)"), the MCU can be unsecured by the backdoor key access sequence described below:

1. Set the KEYACC bit in the Flash Configuration Register (FCNFG).
2. Write the correct four 16-bit words to Flash addresses 0xFF00–0xFF07 sequentially starting with 0x7F_FF00.
3. Clear the KEYACC bit. Depending on the user code used to write the backdoor keys, a wait cycle (NOP) may be required before clearing the KEYACC bit.
4. If all four 16-bit words match the backdoor keys stored in Flash addresses 0x7F_FF00–0x7F_FF07, the MCU is unsecured and the SEC[1:0] bits in the FSEC register are forced to the unsecure state of 1:0.

The backdoor key access sequence is monitored by an internal security state machine. An illegal operation during the backdoor key access sequence will cause the security state machine to lock, leaving the MCU in the secured state. A reset of the MCU will cause the security state machine to exit the lock state and allow a new backdoor key access sequence to be attempted. The following operations during the backdoor key access sequence will lock the security state machine:

1. If any of the four 16-bit words does not match the backdoor keys programmed in the Flash array.
2. If the four 16-bit words are written in the wrong sequence.
3. If more than four 16-bit words are written.
4. If any of the four 16-bit words written are 0x0000 or 0xFFFF.
5. If the KEYACC bit does not remain set while the four 16-bit words are written.
6. If any two of the four 16-bit words are written on successive MCU clock cycles.

After the backdoor keys have been correctly matched, the MCU will be unsecured. Once the MCU is unsecured, the Flash security byte can be programmed to the unsecure state, if desired.

In the unsecure state, the user has full control of the contents of the backdoor keys by programming addresses 0x7F_FF00–0x7F_FF07 in the Flash Configuration Field.

The security as defined in the Flash security byte (0x7F_FF0F) is not changed by using the backdoor key access sequence to unsecure. The backdoor keys stored in addresses 0x7F_FF00–0x7F_FF07 are unaffected by the backdoor key access sequence. After the next reset of the MCU, the security state of the Flash module is determined by the Flash security byte (0x7F_FF0F). The backdoor key access sequence has no effect on the program and erase protections defined in the Flash protection register.

It is not possible to unsecure the MCU in special single chip mode by using the backdoor key access sequence in background debug mode (BDM).

3.6.2 Unsecuring the MCU in Special Single Chip Mode using BDM

The MCU can be unsecured in special single chip mode by erasing the Flash module by the following method:

- Reset the MCU into special single chip mode, delay while the erase test is performed by the BDM secure ROM, send BDM commands to disable protection in the Flash module, and execute a mass erase command write sequence to erase the Flash memory.

After the CCIF flag sets to indicate that the mass operation has completed, reset the MCU into special single chip mode. The BDM secure ROM will verify that the Flash memory is erased and will assert the UNSEC bit in the BDM status register. This BDM action will cause the MCU to override the Flash security state and the MCU will be unsecured. All BDM commands will be enabled and the Flash security byte may be programmed to the unsecure state by the following method:

- Send BDM commands to execute a word program sequence to program the Flash security byte to the unsecured state and reset the MCU.

3.7 Resets

3.7.1 Flash Reset Sequence

On each reset, the Flash module executes a reset sequence to hold CPU activity while loading the following registers from the Flash memory according to Table 3-1:

- FPROT — Flash Protection Register (see Section 3.3.2.5).
- FCTL - Flash Control Register (see Section 3.3.2.8).
- FSEC — Flash Security Register (see Section 3.3.2.2).

3.7.2 Reset While Flash Command Active

If a reset occurs while any Flash command is in progress, that command will be immediately aborted. The state of the word being programmed or the sector/block being erased is not guaranteed.

3.8 Interrupts

The Flash module can generate an interrupt when all Flash command operations have completed, when the Flash address, data and command buffers are empty.

Table 3-20. Flash Interrupt Sources

Interrupt Source	Interrupt Flag	Local Enable	Global (CCR) Mask
Flash Address, Data and Command Buffers empty	CBEIF (FSTAT register)	CBEIE (FCNFG register)	I Bit
All Flash commands completed	CCIF (FSTAT register)	CCIE (FCNFG register)	I Bit

NOTE

Vector addresses and their relative interrupt priority are determined at the MCU level.

3.8.1 Description of Flash Interrupt Operation

The logic used for generating interrupts is shown in Figure 3-32.

The Flash module uses the CBEIF and CCIF flags in combination with the CBIE and CCIE enable bits to generate the Flash command interrupt request.

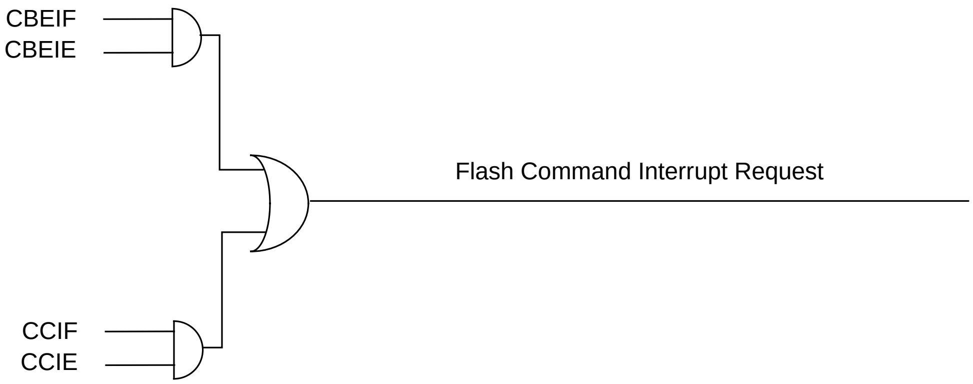


Figure 3-32. Flash Interrupt Implementation

For a detailed description of the register bits, refer to Section 3.3.2.4, “Flash Configuration Register (FCNFG)” and Section 3.3.2.6, “Flash Status Register (FSTAT)”.

Chapter 4
4 Kbyte EEPROM Module (S12XEETX4KV2)

4.1 Introduction

This document describes the EETX4K module which includes a 4 Kbyte EEPROM (nonvolatile) memory. The EEPROM memory may be read as either bytes, aligned words, or misaligned words. Read access time is one bus cycle for bytes and aligned words, and two bus cycles for misaligned words.

The EEPROM memory is ideal for data storage for single-supply applications allowing for field reprogramming without requiring external voltage sources for program or erase. Program and erase functions are controlled by a command driven interface. The EEPROM module supports both block erase (all memory bytes) and sector erase (4 memory bytes). An erased bit reads 1 and a programmed bit reads 0. The high voltage required to program and erase the EEPROM memory is generated internally. It is not possible to read from the EEPROM block while it is being erased or programmed.

CAUTION

> An EEPROM word (2 bytes) must be in the erased state before being programmed. Cumulative programming of bits within a word is not allowed.

4.1.1 Glossary

Command Write Sequence — A three-step MCU instruction sequence to execute built-in algorithms (including program and erase) on the EEPROM memory.

4.1.2 Features

- 4 Kbytes of EEPROM memory divided into 1024 sectors of 4 bytes
- Automated program and erase algorithm
- Interrupts on EEPROM command completion and command buffer empty
- Fast sector erase and word program operation
- 2-stage command pipeline
- Sector erase abort feature for critical interrupt response
- Flexible protection scheme to prevent accidental program or erase
- Single power supply for all EEPROM operations including program and erase

4.1.3 Modes of Operation

Program, erase and erase verify operations (please refer to Section 4.4.1, “EEPROM Command Operations” for details).

4.1.4 Block Diagram

A block diagram of the EEPROM module is shown in Figure 4-1.

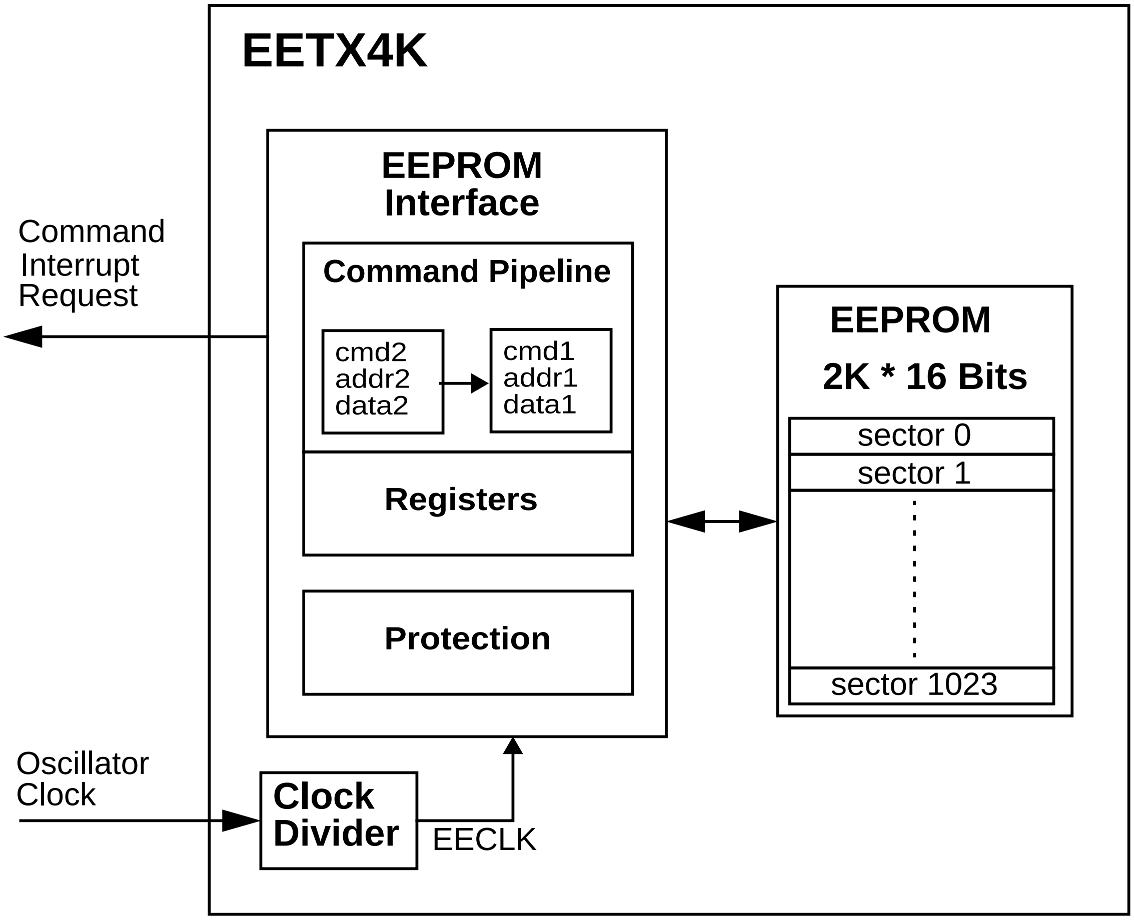


Figure 4-1. EETX4K Block Diagram

4.2 External Signal Description

The EEPROM module contains no signals that connect off-chip.

4.3 Memory Map and Register Definition

This section describes the memory map and registers for the EEPROM module.

4.3.1 Module Memory Map

The EEPROM memory map is shown in Figure 4-2. The HCS12X architecture places the EEPROM memory addresses between global addresses 0x13_F000 and 0x13_FFFF. The EPROT register, described in Section 4.3.2.5, “EEPROM Protection Register (EPROT)”, can be set to protect the upper region in the EEPROM memory from accidental program or erase. The EEPROM addresses covered by this protectable region are shown in the EEPROM memory map. The default protection setting is stored in the EEPROM configuration field as described in Table 4-1.

Table 4-1. EEPROM Configuration Field

Global Address	Size (bytes)	Description
0x13_FFFC	1	Reserved
0x13_FFFD	1	EEPROM Protection byte Refer to Section 4.3.2.5, “EEPROM Protection Register (EPROT)”
0x13_FFFE – 0x13_FFFF	2	Reserved

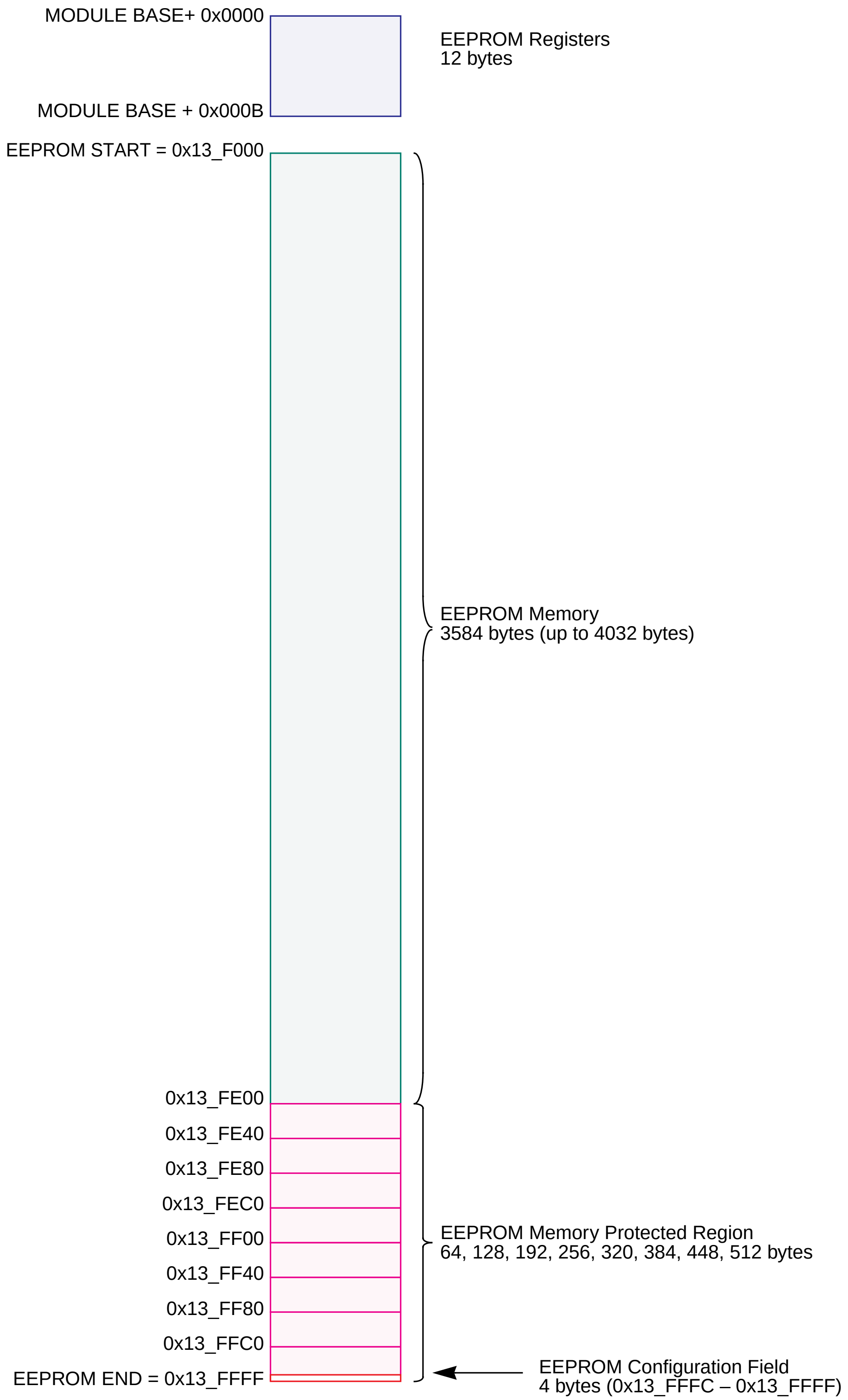


Figure 4-2. EEPROM Memory Map

4.3.2 Register Descriptions

The EEPROM module also contains a set of 12 control and status registers located between EEPROM module base + 0x0000 and 0x000B. A summary of the EEPROM module registers is given in Figure 4-3. Detailed descriptions of each register bit are provided.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 ECLKDIV	R	EDIVLD	PRDIV8	EDIV5	EDIV4	EDIV3	EDIV2	EDIV1	EDIV0
	W	▨							
0x0001 RESERVED1	R	0	0	0	0	0	0	0	0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0002 RESERVED2	R	0	0	0	0	0	0	0	0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0003 ECNFG	R	CBEIE	CCIE	0	0	0	0	0	0
	W			▨	▨	▨	▨	▨	▨
0x0004 EPROT	R	EPOPEN	RNV6	RNV5	RNV4	EPDIS	EPS2	EPS1	EPS0
	W		▨	▨	▨				
0x0005 ESTAT	R	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	0	0
	W		▨			▨	▨	▨	▨
0x0006 ECMD	R	0	CMDB						
	W	▨							
0x0007 RESERVED3	R	0	0	0	0	0	0	0	0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0008 EADDRHI	R	0	0	0	0	0	EABHI		
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0009 EADDRLO	R	EABLO							
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x000A EDATAHI	R	EDHI							
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x000B EDATALO	R	EDLO							
	W	▨	▨	▨	▨	▨	▨	▨	▨

▨ = Unimplemented or Reserved

Figure 4-3. EETX4K Register Summary

4.3.2.1 EEPROM Clock Divider Register (ECLKDIV)

The ECLKDIV register is used to control timed events in program and erase algorithms.

Module Base + 0x0000

	7	6	5	4	3	2	1	0
R	EDIVLD	PRDIV8	EDIV5	EDIV4	EDIV3	EDIV2	EDIV1	EDIV0
W	▨							
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 4-4. EEPROM Clock Divider Register (ECLKDIV)

All bits in the ECLKDIV register are readable, bits 6–0 are write once and bit 7 is not writable.

Table 4-2. ECLKDIV Field Descriptions

Field	Description
7 EDIVLD	**Clock Divider Loaded** 0 Register has not been written. 1 Register has been written to since the last reset.
6 PRDIV8	**Enable Prescalar by 8** 0 The oscillator clock is directly fed into the ECLKDIV divider. 1 Enables a Prescalar by 8, to divide the oscillator clock before feeding into the clock divider.
5:0 EDIV[5:0]	**Clock Divider Bits** — The combination of PRDIV8 and EDIV[5:0] effectively divides the EEPROM module input oscillator clock down to a frequency of 150 kHz – 200 kHz. The maximum divide ratio is 512. Please refer to Section 4.4.1.1, "Writing the ECLKDIV Register" for more information.

4.3.2.2 RESERVED1

This register is reserved for factory testing and is not accessible.

Module Base + 0x0001

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 4-5. RESERVED1

All bits read 0 and are not writable.

4.3.2.3 RESERVED2

This register is reserved for factory testing and is not accessible.

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 4-6. RESERVED2

All bits read 0 and are not writable.

4.3.2.4 EEPROM Configuration Register (ECNFG)

The ECNFG register enables the EEPROM interrupts.

Module Base + 0x0003

	7	6	5	4	3	2	1	0
R	CBEIE	CCIE	0	0	0	0	0	0
W			▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 4-7. EEPROM Configuration Register (ECNFG)

CBEIE and CCIE bits are readable and writable while all remaining bits read 0 and are not writable.

Table 4-3. ECNFG Field Descriptions

Field	Description
7 CBEIE	**Command Buffer Empty Interrupt Enable** — The CBEIE bit enables an interrupt in case of an empty command buffer in the EEPROM module. 0 Command Buffer Empty interrupt disabled. 1 An interrupt will be requested whenever the CBEIF flag (see Section 4.3.2.6, “EEPROM Status Register (ESTAT)”) is set.
6 CCIE	**Command Complete Interrupt Enable** — The CCIE bit enables an interrupt in case all commands have been completed in the EEPROM module. 0 Command Complete interrupt disabled. 1 An interrupt will be requested whenever the CCIF flag (see Section 4.3.2.6, “EEPROM Status Register (ESTAT)”) is set.

4.3.2.5 EEPROM Protection Register (EPROT)

The EPROT register defines which EEPROM sectors are protected against program or erase operations.

Module Base + 0x0004

	7	6	5	4	3	2	1	0
R	EPOPEN	RNV6	RNV5	RNV4	EPDIS	EPS2	EPS1	EPS0
W		▨	▨	▨				
Reset	F	F	F	F	F	F	F	F

▨ = Unimplemented or Reserved

Figure 4-8. EEPROM Protection Register (EPROT)

During the reset sequence, the EPROT register is loaded from the EEPROM Protection byte at address offset 0x0FFD (see Table 4-1).All bits in the EPROT register are readable and writable except for RNV[6:4] which are only readable. The EPOPEN and EPDIS bits can only be written to the protected state. The EPS bits can be written anytime until bit EPDIS is cleared. If the EPOPEN bit is cleared, the state of the EPDIS and EPS bits is irrelevant.

To change the EEPROM protection that will be loaded during the reset sequence, the EEPROM memory must be unprotected, then the EEPROM Protection byte must be reprogrammed. Trying to alter data in any protected area in the EEPROM memory will result in a protection violation error and the PVIOL flag will be set in the ESTAT register. The mass erase of an EEPROM block is possible only when protection is fully disabled by setting the EPOPEN and EPDIS bits.

Table 4-4. EPROT Field Descriptions

Field	Description
7 EPOPEN	**Opens the EEPROM for Program or Erase** 0 The entire EEPROM memory is protected from program and erase. 1 The EEPROM sectors not protected are enabled for program or erase.
6:4 RNV[6:4]	**Reserved Nonvolatile Bits** — The RNV[6:4] bits should remain in the erased state “1” for future enhancements.
3 EPDIS	**EEPROM Protection Address Range Disable** — The EPDIS bit determines whether there is a protected area in a specific region of the EEPROM memory ending with address offset 0x0FFF. 0 Protection enabled. 1 Protection disabled.
2:0 EPS[2:0]	**EEPROM Protection Address Size** — The EPS[2:0] bits determine the size of the protected area as shown inTable 4-5. The EPS bits can only be written to while the EPDIS bit is set.

Table 4-5. EEPROM Protection Address Range

EPS[2:0]	Address Offset Range	Protected Size
000	0x0FC0 – 0x0FFF	64 bytes
001	0x0F80 – 0x0FFF	128 bytes
010	0x0F40 – 0x0FFF	192 bytes
011	0x0F00 – 0x0FFF	256 bytes
100	0x0EC0 – 0x0FFF	320 bytes
101	0x0E80 – 0x0FFF	384 bytes
110	0x0E40 – 0x0FFF	448 bytes
111	0x0E00 – 0x0FFF	512 bytes

4.3.2.6 EEPROM Status Register (ESTAT)

The ESTAT register defines the operational status of the module.

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	0	0
W		▨			▨	▨	▨	▨
Reset	1	1	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 4-9. EEPROM Status Register (ESTAT — Normal Mode)

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	FAIL	0
W		▨			▨	▨		▨
Reset	1	1	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 4-10. EEPROM Status Register (ESTAT — Special Mode)

CBEIF, PVIOL, and ACCERR are readable and writable, CCIF and BLANK are readable and not writable, remaining bits read 0 and are not writable in normal mode. FAIL is readable and writable in special mode.

Table 4-6. ESTAT Field Descriptions

Field	Description
7 CBEIF	**Command Buffer Empty Interrupt Flag** — The CBEIF flag indicates that the address, data, and command buffers are empty so that a new command write sequence can be started. The CBEIF flag is cleared by writing a 1 to CBEIF. Writing a 0 to the CBEIF flag has no effect on CBEIF. Writing a 0 to CBEIF after writing an aligned word to the EEPROM address space but before CBEIF is cleared will abort a command write sequence and cause the ACCERR flag to be set. Writing a 0 to CBEIF outside of a command write sequence will not set the ACCERR flag. The CBEIF flag is used together with the CBEIE bit in the ECNFG register to generate an interrupt request (see Figure 4-24). 0 Buffers are full. 1 Buffers are ready to accept a new command.
6 CCIF	**Command Complete Interrupt Flag** — The CCIF flag indicates that there are no more commands pending. The CCIF flag is cleared when CBEIF is clear and sets automatically upon completion of all active and pending commands. The CCIF flag does not set when an active commands completes and a pending command is fetched from the command buffer. Writing to the CCIF flag has no effect on CCIF. The CCIF flag is used together with the CCIE bit in the ECNFG register to generate an interrupt request (see Figure 4-24). 0 Command in progress. 1 All commands are completed.
5 PVIOL	**Protection Violation Flag** — The PVIOL flag indicates an attempt was made to program or erase an address in a protected area of the EEPROM memory during a command write sequence. The PVIOL flag is cleared by writing a 1 to PVIOL. Writing a 0 to the PVIOL flag has no effect on PVIOL. While PVIOL is set, it is not possible to launch a command or start a command write sequence. 0 No failure. 1 A protection violation has occurred.
4 ACCERR	**Access Error Flag** — The ACCERR flag indicates an illegal access has occurred to the EEPROM memory caused by either a violation of the command write sequence (see Section 4.4.1.2, “Command Write Sequence”), issuing an illegal EEPROM command (see Table 4-8), launching the sector erase abort command terminating a sector erase operation early (see Section 4.4.2.5, “Sector Erase Abort Command”) or the execution of a CPU STOP instruction while a command is executing (CCIF = 0). The ACCERR flag is cleared by writing a 1 to ACCERR. Writing a 0 to the ACCERR flag has no effect on ACCERR. While ACCERR is set, it is not possible to launch a command or start a command write sequence. If ACCERR is set by an erase verify operation, any buffered command will not launch. 0 No access error detected. 1 Access error has occurred.
2 BLANK	**Flag Indicating the Erase Verify Operation Status** — When the CCIF flag is set after completion of an erase verify command, the BLANK flag indicates the result of the erase verify operation. The BLANK flag is cleared by the EEPROM module when CBEIF is cleared as part of a new valid command write sequence. Writing to the BLANK flag has no effect on BLANK. 0 EEPROM block verified as not erased. 1 EEPROM block verified as erased.
1 FAIL	**Flag Indicating a Failed EEPROM Operation** — The FAIL flag will set if the erase verify operation fails (EEPROM block verified as not erased). The FAIL flag is cleared by writing a 1 to FAIL. Writing a 0 to the FAIL flag has no effect on FAIL. 0 EEPROM operation completed without error. 1 EEPROM operation failed.

4.3.2.7 EEPROM Command Register (ECMD)

The ECMD register is the EEPROM command register.

Module Base + 0x0006

	7	6	5	4	3	2	1	0
R	0	CMDB						
W	▨							
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 4-11. EEPROM Command Register (ECMD)

All CMDB bits are readable and writable during a command write sequence while bit 7 reads 0 and is not writable.

Table 4-7. ECMD Field Descriptions

Field	Description
6:0 CMDB[6:0]	**EEPROM Command Bits** — Valid EEPROM commands are shown in Table 4-8. Writing any command other than those listed in Table 4-8 sets the ACCERR flag in the ESTAT register.

Table 4-8. Valid EEPROM Command List

CMDB[6:0]	Command
0x05	Erase Verify
0x20	Word Program
0x40	Sector Erase
0x41	Mass Erase
0x47	Sector Erase Abort
0x60	Sector Modify

4.3.2.8 RESERVED3

This register is reserved for factory testing and is not accessible.

Module Base + 0x0007

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 4-12. RESERVED3

All bits read 0 and are not writable.

EEPROM Address Registers (EADDR)

The EADDRHI and EADDRLO registers are the EEPROM address registers.

Module Base + 0x0008

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	EABHI		
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 4-13. EEPROM Address High Register (EADDRHI)

Module Base + 0x0009

	7	6	5	4	3	2	1	0
R	EABLO							
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 4-14. EEPROM Address Low Register (EADDRLO)

All EABHI and EABLO bits read 0 and are not writable in normal modes.

All EABHI and EABLO bits are readable and writable in special modes.

The MCU address bit AB0 is not stored in the EADDR registers since the EEPROM block is not byte addressable.

4.3.2.9 EEPROM Data Registers (EDATA)

The EDATAHI and EDATALO registers are the EEPROM data registers.

Module Base + 0x000A

	7	6	5	4	3	2	1	0
R	EDHI							
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 4-15. EEPROM Data High Register (EDATAHI)

Module Base + 0x000B

	7	6	5	4	3	2	1	0
R	EDLO							
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 4-16. EEPROM Data Low Register (EDATALO)

All EDHI and EDLO bits read 0 and are not writable in normal modes.

All EDHI and EDLO bits are readable and writable in special modes.

4.4 Functional Description

4.4.1 EEPROM Command Operations

Write operations are used to execute program, erase, erase verify, sector erase abort, and sector modify algorithms described in this section. The program, erase, and sector modify algorithms are controlled by a state machine whose timebase, EECLK, is derived from the oscillator clock via a programmable divider. The command register as well as the associated address and data registers operate as a buffer and a register (2-stage FIFO) so that a second command along with the necessary data and address can be stored to the buffer while the first command is still in progress. Buffer empty as well as command completion are signalled by flags in the EEPROM status register with interrupts generated, if enabled.

The next sections describe:

1. How to write the ECLKDIV register
2. Command write sequences to program, erase, erase verify, sector erase abort, and sector modify operations on the EEPROM memory
3. Valid EEPROM commands
4. Effects resulting from illegal EEPROM command write sequences or aborting EEPROM operations

4.4.1.1 Writing the ECLKDIV Register

Prior to issuing any EEPROM command after a reset, the user is required to write the ECLKDIV register to divide the oscillator clock down to within the 150 kHz to 200 kHz range. Since the program and erase timings are also a function of the bus clock, the ECLKDIV determination must take this information into account.

If we define:

- ECLK as the clock of the EEPROM timing control block
- Tbus as the period of the bus clock
- INT(x) as taking the integer part of x (e.g., INT(4.323)=4)

then ECLKDIV register bits PRDIV8 and EDIV[5:0] are to be set as described in Figure 4-17.

For example, if the oscillator clock frequency is 950 kHz and the bus clock frequency is 10 MHz, ECLKDIV bits EDIV[5:0] should be set to 0x04 (000100) and bit PRDIV8 set to 0. The resulting EECLK frequency is then 190 kHz. As a result, the EEPROM program and erase algorithm timings are increased over the optimum target by:

$$(200 - 190) / 200 \times 100 \ = \ 5\%$$

If the oscillator clock frequency is 16 MHz and the bus clock frequency is 40 MHz, ECLKDIV bits EDIV[5:0] should be set to 0x0A (001010) and bit PRDIV8 set to 1. The resulting EECLK frequency is then 182 kHz. In this case, the EEPROM program and erase algorithm timings are increased over the optimum target by:

$$(200 - 182) / 200 \times 100 \ = \ 9\%$$

CAUTION

> Program and erase command execution time will increase proportionally with the period of EECLK. Because of the impact of clock synchronization on the accuracy of the functional timings, programming or erasing the EEPROM memory cannot be performed if the bus clock runs at less than 1 MHz. Programming or erasing the EEPROM memory with EECLK < 150 kHz should be avoided. Setting ECLKDIV to a value such that EECLK < 150 kHz can destroy the EEPROM memory due to overstress. Setting ECLKDIV to a value such that (1/EECLK+Tbus) < 5 µs can result in incomplete programming or erasure of the EEPROM memory cells.

If the ECLKDIV register is written, the EDIVLD bit is set automatically. If the EDIVLD bit is 0, the ECLKDIV register has not been written since the last reset. If the ECLKDIV register has not been written to, the EEPROM command loaded during a command write sequence will not execute and the ACCERR flag in the ESTAT register will set.

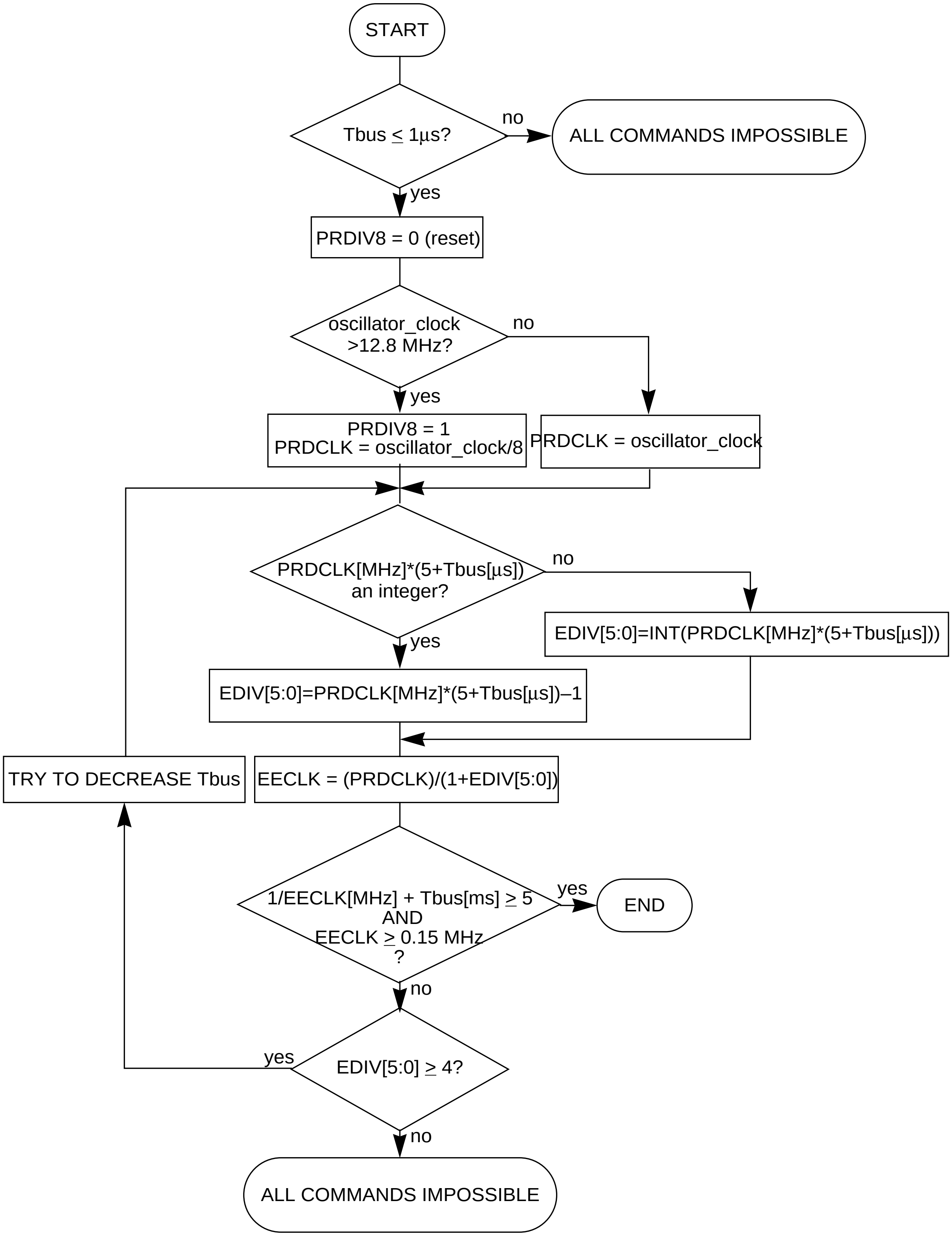


Figure 4-17. Determination Procedure for PRDIV8 and EDIV Bits

4.4.1.2 Command Write Sequence

The EEPROM command controller is used to supervise the command write sequence to execute program, erase, erase verify, sector erase abort, and sector modify algorithms.

Before starting a command write sequence, the ACCERR and PVIOL flags in the ESTAT register must be clear (see Section 4.3.2.6, “EEPROM Status Register (ESTAT)”) and the CBEIF flag should be tested to determine the state of the address, data and command buffers. If the CBEIF flag is set, indicating the buffers are empty, a new command write sequence can be started. If the CBEIF flag is clear, indicating the buffers are not available, a new command write sequence will overwrite the contents of the address, data and command buffers.

A command write sequence consists of three steps which must be strictly adhered to with writes to the EEPROM module not permitted between the steps. However, EEPROM register and array reads are allowed during a command write sequence. The basic command write sequence is as follows:

1. Write to one address in the EEPROM memory.
2. Write a valid command to the ECMD register.
3. Clear the CBEIF flag in the ESTAT register by writing a 1 to CBEIF to launch the command.

The address written in step 1 will be stored in the EADDR registers and the data will be stored in the EDATA registers. If the CBEIF flag in the ESTAT register is clear when the first EEPROM array write occurs, the contents of the address and data buffers will be overwritten and the CBEIF flag will be set. When the CBEIF flag is cleared, the CCIF flag is cleared on the same bus cycle by the EEPROM command controller indicating that the command was successfully launched. For all command write sequences except sector erase abort, the CBEIF flag will set four bus cycles after the CCIF flag is cleared indicating that the address, data, and command buffers are ready for a new command write sequence to begin. For sector erase abort operations, the CBEIF flag will remain clear until the operation completes. Except for the sector erase abort command, a buffered command will wait for the active operation to be completed before being launched. The sector erase abort command is launched when the CBEIF flag is cleared as part of a sector erase abort command write sequence. Once a command is launched, the completion of the command operation is indicated by the setting of the CCIF flag in the ESTAT register. The CCIF flag will set upon completion of all active and buffered commands.

4.4.2 EEPROM Commands

Table 4-9 summarizes the valid EEPROM commands along with the effects of the commands on the EEPROM block.

Table 4-9. EEPROM Command Description

ECMDB	Command	Function on EEPROM Memory
0x05	Erase Verify	Verify all memory bytes in the EEPROM block are erased. If the EEPROM block is erased, the BLANK flag in the ESTAT register will set upon command completion.
0x20	Program	Program a word (two bytes) in the EEPROM block.
0x40	Sector Erase	Erase all four memory bytes in a sector of the EEPROM block.

Table 4-9. EEPROM Command Description

ECMDB	Command	Function on EEPROM Memory
0x41	Mass Erase	Erase all memory bytes in the EEPROM block. A mass erase of the full EEPROM block is only possible when EPOPEN and EPDIS bits in the EPROT register are set prior to launching the command.
0x47	Sector Erase Abort	Abort the sector erase operation. The sector erase operation will terminate according to a set procedure. The EEPROM sector should not be considered erased if the ACCERR flag is set upon command completion.
0x60	Sector Modify	Erase all four memory bytes in a sector of the EEPROM block and reprogram the addressed word.

CAUTION

An EEPROM word (2 bytes) must be in the erased state before being programmed. Cumulative programming of bits within a word is not allowed.

4.4.2.1 Erase Verify Command

The erase verify operation will verify that the EEPROM memory is erased.

An example flow to execute the erase verify operation is shown in Figure 4-18. The erase verify command write sequence is as follows:

1. Write to an EEPROM address to start the command write sequence for the erase verify command. The address and data written will be ignored.
2. Write the erase verify command, 0x05, to the ECMD register.
3. Clear the CBEIF flag in the ESTAT register by writing a 1 to CBEIF to launch the erase verify command.

After launching the erase verify command, the CCIF flag in the ESTAT register will set after the operation has completed unless a new command write sequence has been buffered. The number of bus cycles required to execute the erase verify operation is equal to the number of words in the EEPROM memory plus 14 bus cycles as measured from the time the CBEIF flag is cleared until the CCIF flag is set. Upon completion of the erase verify operation, the BLANK flag in the ESTAT register will be set if all addresses in the EEPROM memory are verified to be erased. If any address in the EEPROM memory is not erased, the erase verify operation will terminate and the BLANK flag in the ESTAT register will remain clear.

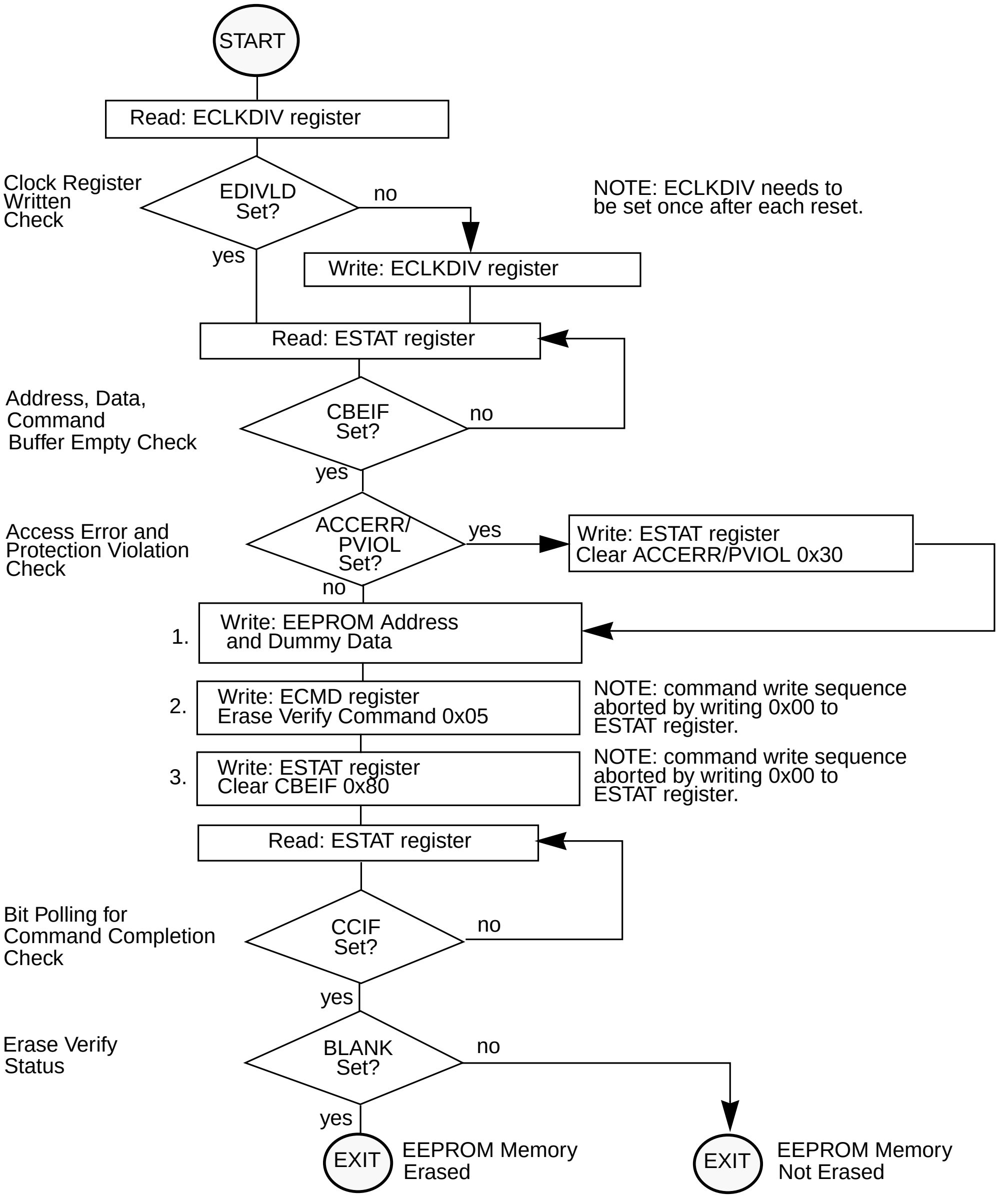


Figure 4-18. Example Erase Verify Command Flow

4.4.2.2 Program Command

The program operation will program a previously erased word in the EEPROM memory using an embedded algorithm.

An example flow to execute the program operation is shown in Figure 4-19. The program command write sequence is as follows:

1. Write to an EEPROM block address to start the command write sequence for the program command. The data written will be programmed to the address written.
2. Write the program command, 0x20, to the ECMD register.
3. Clear the CBEIF flag in the ESTAT register by writing a 1 to CBEIF to launch the program command.

If a word to be programmed is in a protected area of the EEPROM memory, the PVIOL flag in the ESTAT register will set and the program command will not launch. Once the program command has successfully launched, the CCIF flag in the ESTAT register will set after the program operation has completed unless a new command write sequence has been buffered.

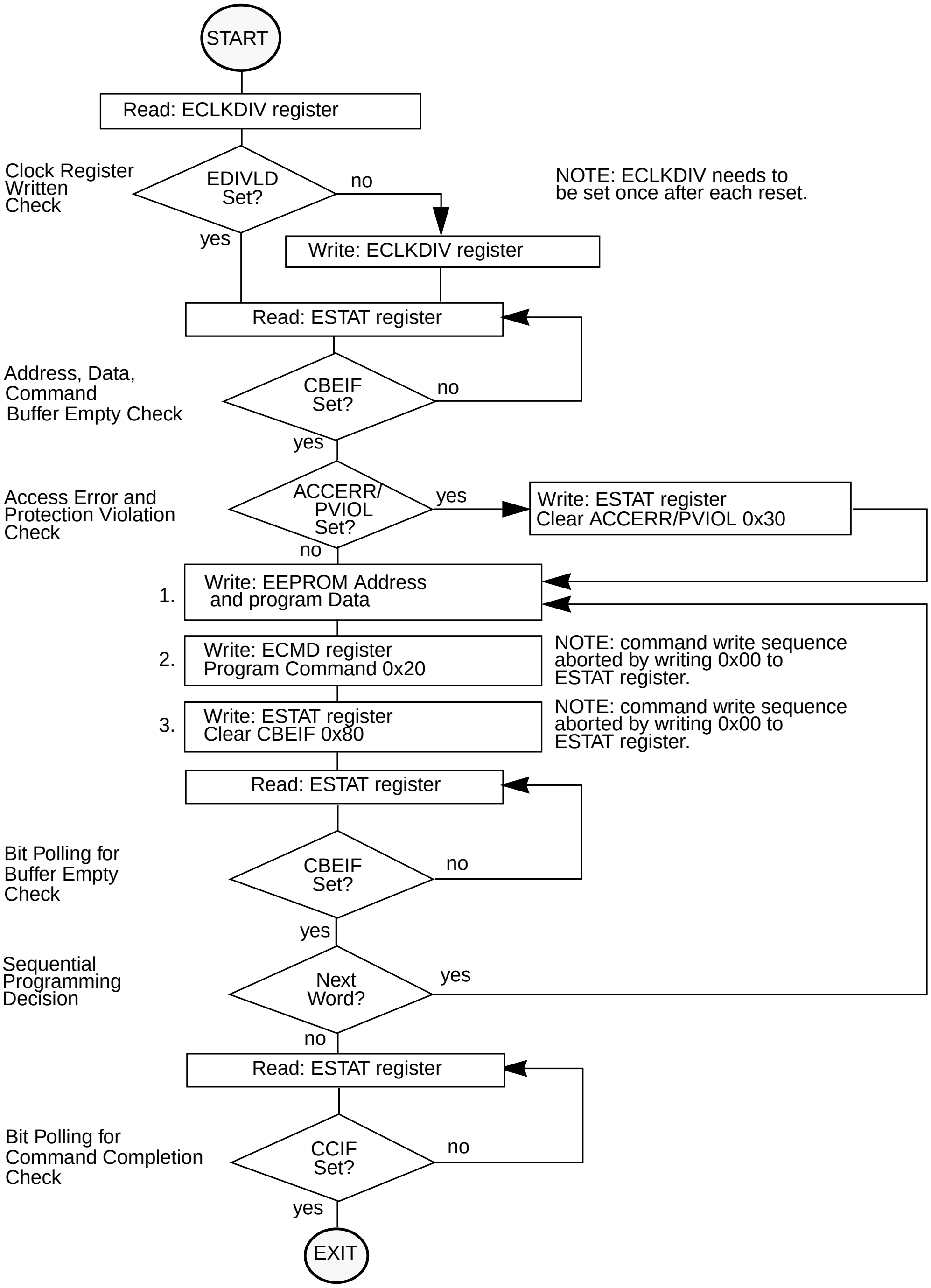


Figure 4-19. Example Program Command Flow

4.4.2.3 Sector Erase Command

The sector erase operation will erase both words in a sector of EEPROM memory using an embedded algorithm.

An example flow to execute the sector erase operation is shown in Figure 4-20. The sector erase command write sequence is as follows:

1. Write to an EEPROM memory address to start the command write sequence for the sector erase command. The EEPROM address written determines the sector to be erased while global address bits [1:0] and the data written are ignored.
2. Write the sector erase command, 0x40, to the ECMD register.
3. Clear the CBEIF flag in the ESTAT register by writing a 1 to CBEIF to launch the sector erase command.

If an EEPROM sector to be erased is in a protected area of the EEPROM memory, the PVIOL flag in the ESTAT register will set and the sector erase command will not launch. Once the sector erase command has successfully launched, the CCIF flag in the ESTAT register will set after the sector erase operation has completed unless a new command write sequence has been buffered.

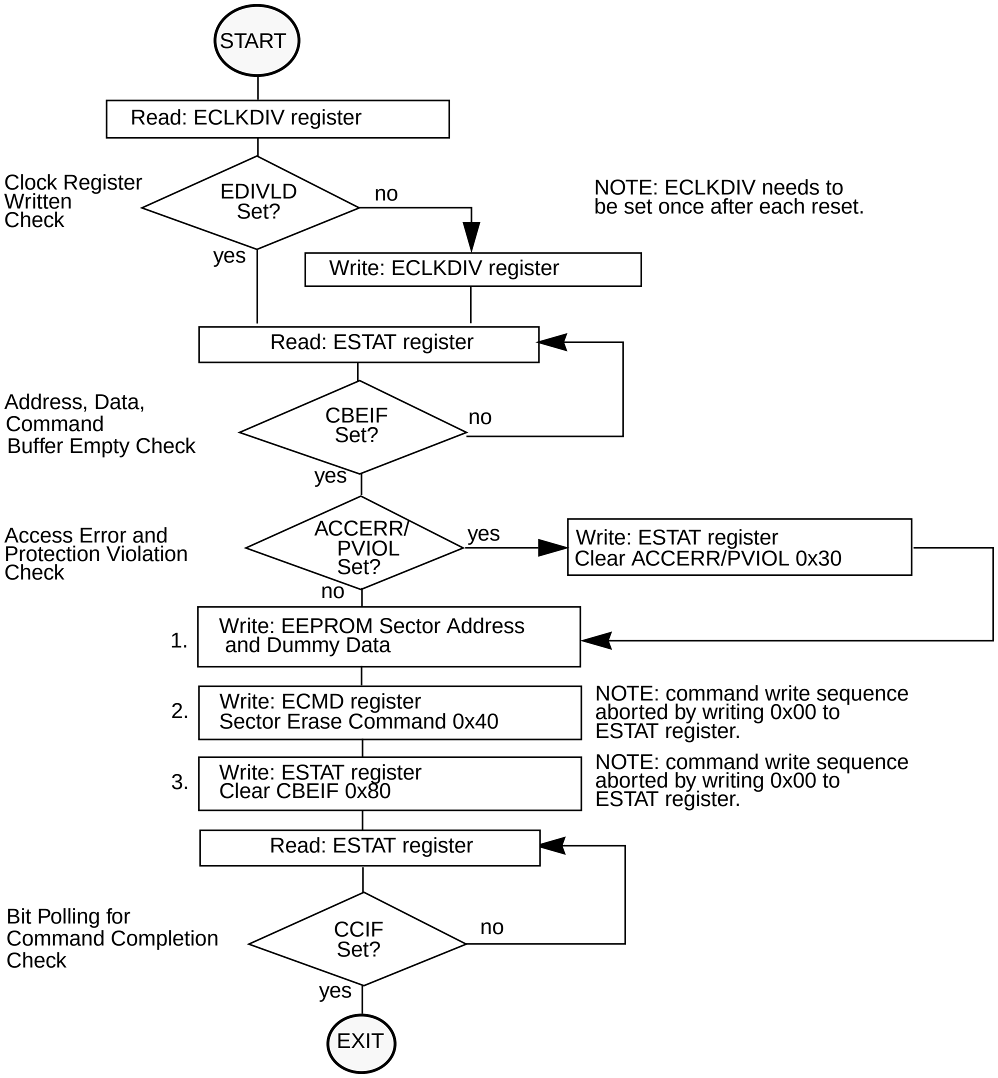


Figure 4-20. Example Sector Erase Command Flow

4.4.2.4 Mass Erase Command

The mass erase operation will erase all addresses in an EEPROM block using an embedded algorithm.

An example flow to execute the mass erase operation is shown in Figure 4-21. The mass erase command write sequence is as follows:

1. Write to an EEPROM memory address to start the command write sequence for the mass erase command. The address and data written will be ignored.
2. Write the mass erase command, 0x41, to the ECMD register.
3. Clear the CBEIF flag in the ESTAT register by writing a 1 to CBEIF to launch the mass erase command.

If the EEPROM memory to be erased contains any protected area, the PVIOL flag in the ESTAT register will set and the mass erase command will not launch. Once the mass erase command has successfully launched, the CCIF flag in the ESTAT register will set after the mass erase operation has completed unless a new command write sequence has been buffered.

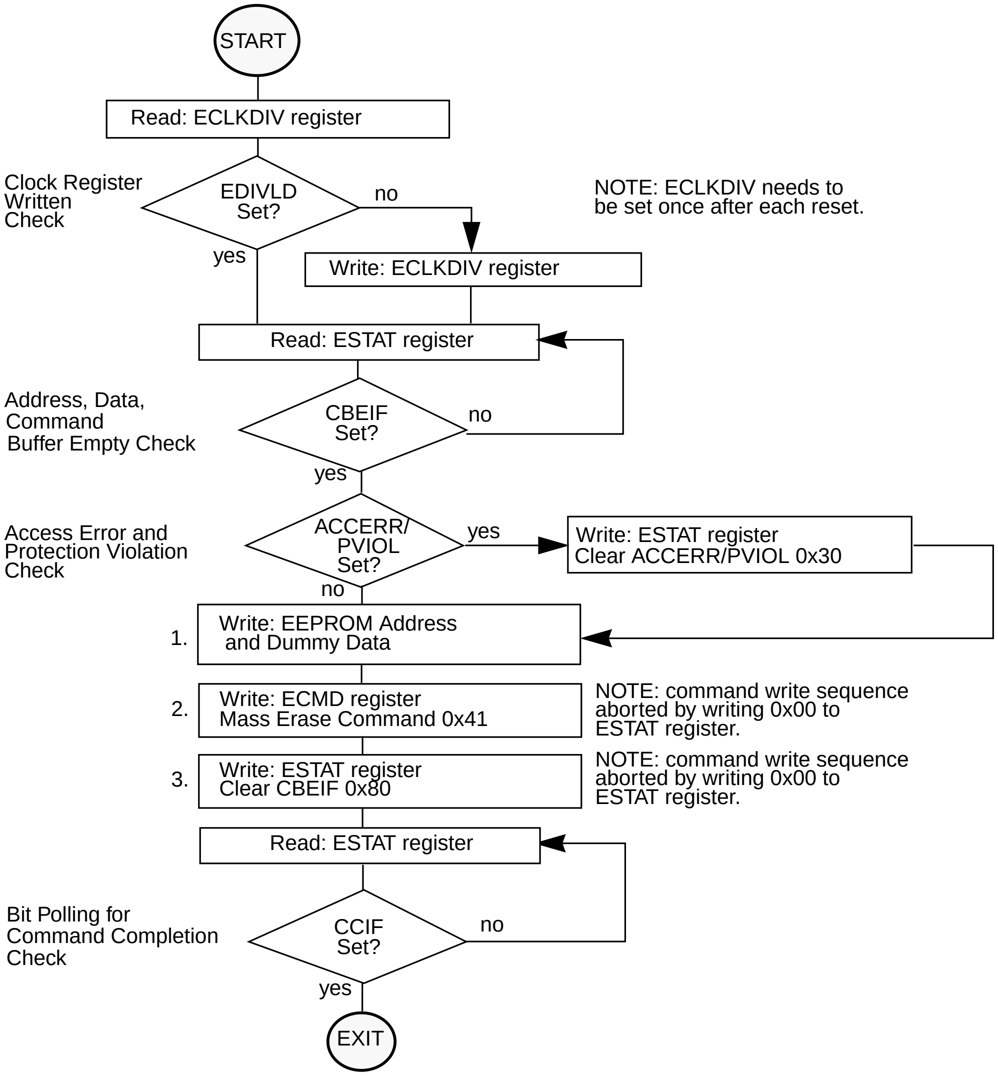


Figure 4-21. Example Mass Erase Command Flow

4.4.2.5 Sector Erase Abort Command

The sector erase abort operation will terminate the active sector erase or sector modify operation so that other sectors in an EEPROM block are available for read and program operations without waiting for the sector erase or sector modify operation to complete.

An example flow to execute the sector erase abort operation is shown in Figure 4-22. The sector erase abort command write sequence is as follows:

1. Write to any EEPROM memory address to start the command write sequence for the sector erase abort command. The address and data written are ignored.
2. Write the sector erase abort command, 0x47, to the ECMD register.
3. Clear the CBEIF flag in the ESTAT register by writing a 1 to CBEIF to launch the sector erase abort command.

If the sector erase abort command is launched resulting in the early termination of an active sector erase or sector modify operation, the ACCERR flag will set once the operation completes as indicated by the CCIF flag being set. The ACCERR flag sets to inform the user that the EEPROM sector may not be fully erased and a new sector erase or sector modify command must be launched before programming any location in that specific sector. If the sector erase abort command is launched but the active sector erase or sector modify operation completes normally, the ACCERR flag will not set upon completion of the operation as indicated by the CCIF flag being set. If the sector erase abort command is launched after the sector modify operation has completed the sector erase step, the program step will be allowed to complete. The maximum number of cycles required to abort a sector erase or sector modify operation is equal to four EECLK periods (see Section 4.4.1.1, “Writing the ECLKDIV Register”) plus five bus cycles as measured from the time the CBEIF flag is cleared until the CCIF flag is set.

NOTE

Since the ACCERR bit in the ESTAT register may be set at the completion of the sector erase abort operation, a command write sequence is not allowed to be buffered behind a sector erase abort command write sequence. The CBEIF flag will not set after launching the sector erase abort command to indicate that a command should not be buffered behind it. If an attempt is made to start a new command write sequence with a sector erase abort operation active, the ACCERR flag in the ESTAT register will be set. A new command write sequence may be started after clearing the ACCERR flag, if set.

NOTE

The sector erase abort command should be used sparingly since a sector erase operation that is aborted counts as a complete program/erase cycle.

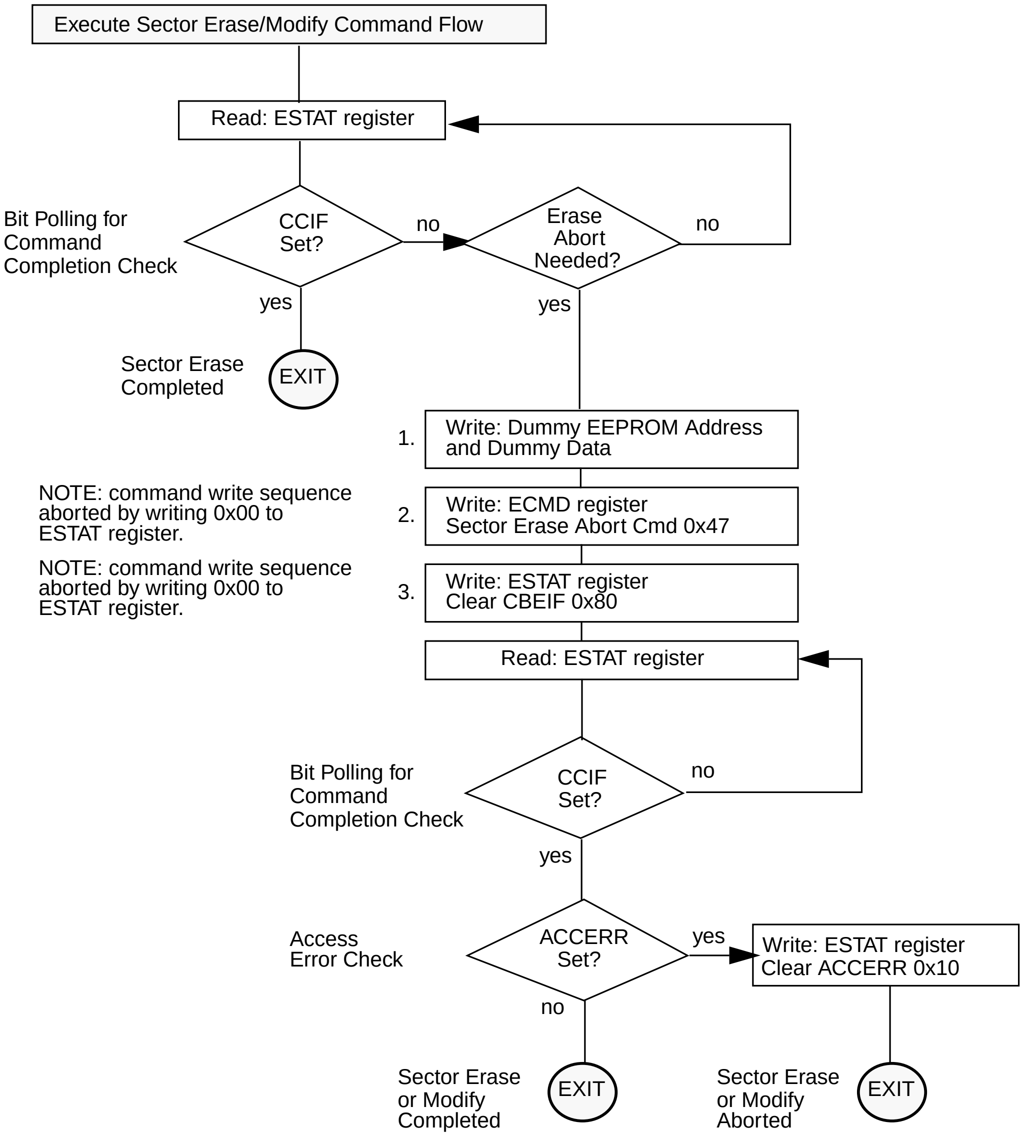


Figure 4-22. Example Sector Erase Abort Command Flow

4.4.2.6 Sector Modify Command

The sector modify operation will erase both words in a sector of EEPROM memory followed by a reprogram of the addressed word using an embedded algorithm.

An example flow to execute the sector modify operation is shown in Figure 4-23. The sector modify command write sequence is as follows:

1. Write to an EEPROM memory address to start the command write sequence for the sector modify command. The EEPROM address written determines the sector to be erased and word to be reprogrammed while byte address bit 0 is ignored.
2. Write the sector modify command, 0x60, to the ECMD register.
3. Clear the CBEIF flag in the ESTAT register by writing a 1 to CBEIF to launch the sector erase command.

If an EEPROM sector to be modified is in a protected area of the EEPROM memory, the PVIOL flag in the ESTAT register will set and the sector modify command will not launch. Once the sector modify command has successfully launched, the CCIF flag in the ESTAT register will set after the sector modify operation has completed unless a new command write sequence has been buffered.

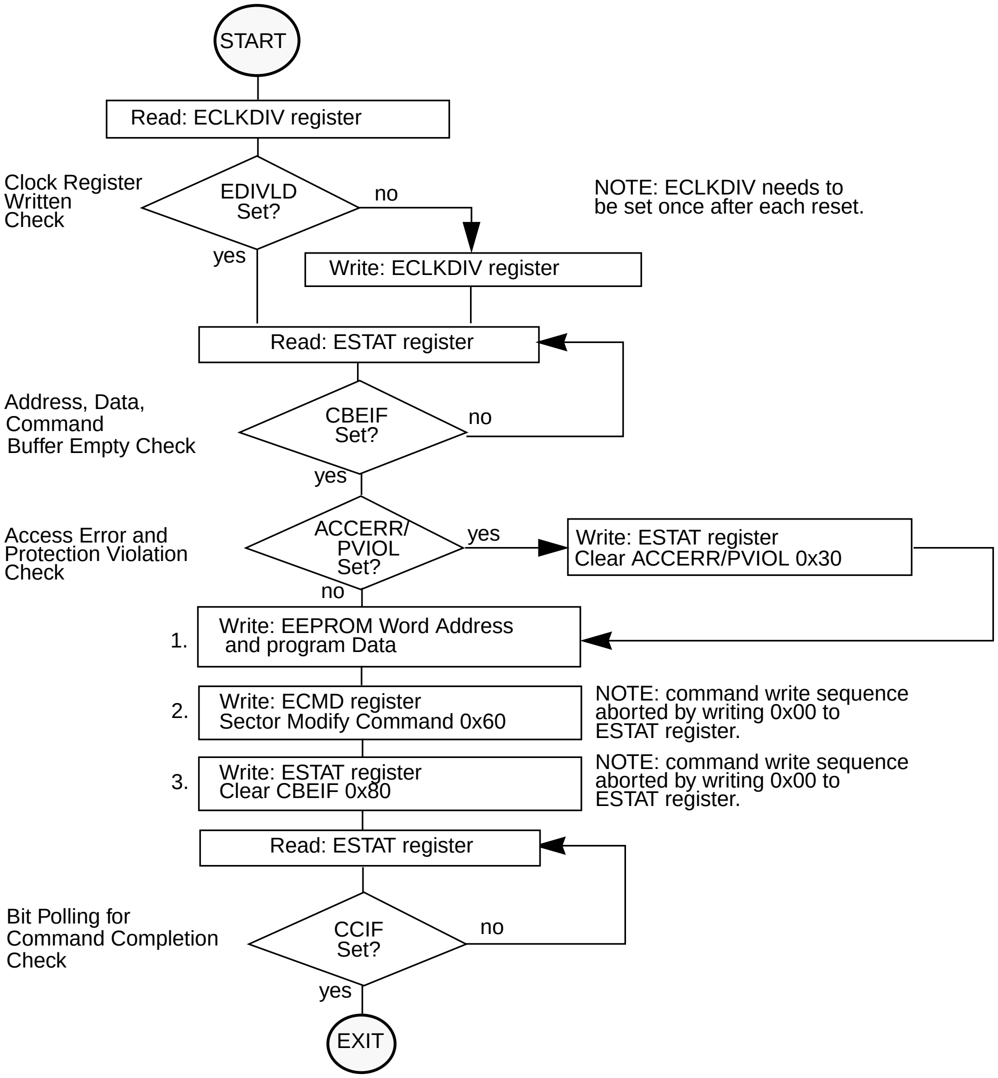


Figure 4-23. Example Sector Modify Command Flow

4.4.3 Illegal EEPROM Operations

The ACCERR flag will be set during the command write sequence if any of the following illegal steps are performed, causing the command write sequence to immediately abort:

1. Writing to an EEPROM address before initializing the ECLKDIV register.
2. Writing a byte or misaligned word to a valid EEPROM address.
3. Starting a command write sequence while a sector erase abort operation is active.
4. Writing to any EEPROM register other than ECMD after writing to an EEPROM address.
5. Writing a second command to the ECMD register in the same command write sequence.
6. Writing an invalid command to the ECMD register.
7. Writing to an EEPROM address after writing to the ECMD register.
8. Writing to any EEPROM register other than ESTAT (to clear CBEIF) after writing to the ECMD register.
9. Writing a 0 to the CBEIF flag in the ESTAT register to abort a command write sequence.

The ACCERR flag will not be set if any EEPROM register is read during a valid command write sequence.

The ACCERR flag will also be set if any of the following events occur:

1. Launching the sector erase abort command while a sector erase or sector modify operation is active which results in the early termination of the sector erase or sector modify operation (see Section 4.4.2.5, “Sector Erase Abort Command”).
2. The MCU enters stop mode and a command operation is in progress. The operation is aborted immediately and any pending command is purged (see Section 4.5.2, “Stop Mode”).

If the EEPROM memory is read during execution of an algorithm (CCIF = 0), the read operation will return invalid data and the ACCERR flag will not be set.

If the ACCERR flag is set in the ESTAT register, the user must clear the ACCERR flag before starting another command write sequence (see Section 4.3.2.6, “EEPROM Status Register (ESTAT)”).

The PVIOL flag will be set after the command is written to the ECMD register during a command write sequence if any of the following illegal operations are attempted, causing the command write sequence to immediately abort:

1. Writing the program command if the address written in the command write sequence was in a protected area of the EEPROM memory.
2. Writing the sector erase command if the address written in the command write sequence was in a protected area of the EEPROM memory.
3. Writing the mass erase command to the EEPROM memory while any EEPROM protection is enabled.
4. Writing the sector modify command if the address written in the command write sequence was in a protected area of the EEPROM memory.

If the PVIOL flag is set in the ESTAT register, the user must clear the PVIOL flag before starting another command write sequence (see Section 4.3.2.6, “EEPROM Status Register (ESTAT)”).

4.5 Operating Modes

4.5.1 Wait Mode

If a command is active (CCIF = 0) when the MCU enters the wait mode, the active command and any buffered command will be completed.

The EEPROM module can recover the MCU from wait mode if the CBEIF and CCIF interrupts are enabled (see Section 4.8, “Interrupts”).

4.5.2 Stop Mode

If a command is active (CCIF = 0) when the MCU enters the stop mode, the operation will be aborted and, if the operation is program, sector erase, mass erase, or sector modify, the EEPROM array data being programmed or erased may be corrupted and the CCIF and ACCERR flags will be set. If active, the high voltage circuitry to the EEPROM memory will immediately be switched off when entering stop mode. Upon exit from stop mode, the CBEIF flag is set and any buffered command will not be launched. The ACCERR flag must be cleared before starting a command write sequence (see Section 4.4.1.2, “Command Write Sequence”).

NOTE

As active commands are immediately aborted when the MCU enters stop mode, it is strongly recommended that the user does not use the STOP instruction during program, sector erase, mass erase, or sector modify operations.

4.5.3 Background Debug Mode

In background debug mode (BDM), the EPROT register is writable. If the MCU is unsecured, then all EEPROM commands listed in Table 4-9 can be executed. If the MCU is secured and is in special single chip mode, the only command available to execute is mass erase.

4.6 EEPROM Module Security

The EEPROM module does not provide any security information to the MCU. After each reset, the security state of the MCU is a function of information provided by the Flash module (see the specific FTX Block Guide).

4.6.1 Unsecuring the MCU in Special Single Chip Mode using BDM

Before the MCU can be unsecured in special single chip mode, the EEPROM memory must be erased using the following method :

- Reset the MCU into special single chip mode, delay while the erase test is performed by the BDM secure ROM, send BDM commands to disable protection in the EEPROM module, and execute a mass erase command write sequence to erase the EEPROM memory.

After the CCIF flag sets to indicate that the EEPROM mass operation has completed and assuming that the Flash memory has also been erased, reset the MCU into special single chip mode. The BDM secure ROM will verify that the Flash and EEPROM memory are erased and will assert the UNSEC bit in the BDM status register. This BDM action will cause the MCU to override the Flash security state and the MCU will be unsecured. Once the MCU is unsecured, BDM commands will be enabled and the Flash security byte may be programmed to the unsecure state.

4.7 Resets

4.7.1 EEPROM Reset Sequence

On each reset, the EEPROM module executes a reset sequence to hold CPU activity while loading the EPROT register from the EEPROM memory according to Table 4-1.

4.7.2 Reset While EEPROM Command Active

If a reset occurs while any EEPROM command is in progress, that command will be immediately aborted. The state of a word being programmed or the sector / block being erased is not guaranteed.

4.8 Interrupts

The EEPROM module can generate an interrupt when all EEPROM command operations have completed, when the EEPROM address, data, and command buffers are empty.

Table 4-10. EEPROM Interrupt Sources

Interrupt Source	Interrupt Flag	Local Enable	Global (CCR) Mask
EEPROM address, data, and command buffers empty	CBEIF (ESTAT register)	CBEIE (ECNFG register)	I Bit
All EEPROM commands completed	CCIF (ESTAT register)	CCIE (ECNFG register)	I Bit

NOTE

Vector addresses and their relative interrupt priority are determined at the MCU level.

4.8.1 Description of EEPROM Interrupt Operation

The logic used for generating interrupts is shown in Figure 4-24.

The EEPROM module uses the CBEIF and CCIF flags in combination with the CBIE and CCIE enable bits to generate the EEPROM command interrupt request.

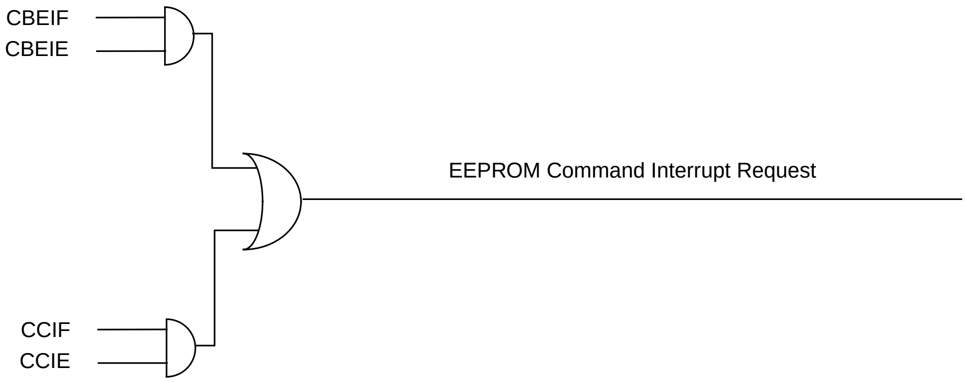


Figure 4-24. EEPROM Interrupt Implementation

For a detailed description of the register bits, refer to Section 4.3.2.4, “EEPROM Configuration Register (ECNFG)” and Section 4.3.2.6, “EEPROM Status Register (ESTAT)” .

Chapter 5
XGATE (S12XGATEV2)

Revision History

Version Number	Date	Author	Description of Changes
02.22	14 Dec 2005		Updated code example
02.23	19 Mar 2007		Internal updates
V03.24	13 Feb 2009		- Minor corrections (5.3.1.4/5-220)

5.1 Introduction

The XGATE module is a peripheral co-processor that allows autonomous data transfers between the MCU's peripherals and the internal memories. It has a built in RISC core that is able to pre-process the transferred data and perform complex communication protocols.

The XGATE module is intended to increase the MCU's data throughput by lowering the S12X_CPU's interrupt load.

Figure 5-1 gives an overview on the XGATE architecture.

This document describes the functionality of the XGATE module, including:

- XGATE registers (Section 5.3, "Memory Map and Register Definition")
- XGATE RISC core (**Section 5.4.1, "XGATE RISC Core**")
- Hardware semaphores (Section 5.4.4, "Semaphores")
- Interrupt handling (Section 5.5, "Interrupts")
- Debug features (Section 5.6, "Debug Mode")
- Security (Section 5.7, "Security")
- Instruction set (Section 5.8, "Instruction Set")

5.1.1 Glossary of Terms

XGATE Request

A service request from a peripheral module which is directed to the XGATE by the S12X_INT module (see Figure 5-1).

XGATE Channel

The resources in the XGATE module (i.e. Channel ID number, Priority level, Service Request Vector, Interrupt Flag) which are associated with a particular XGATE Request.

XGATE Channel ID

A 7-bit identifier associated with an XGATE channel. In S12X designs valid Channel IDs range from $78 to $09.

XGATE Channel Interrupt

An S12X_CPU interrupt that is triggered by a code sequence running on the XGATE module.

XGATE Software Channel

Special XGATE channel that is not associated with any peripheral service request. A Software Channel is triggered by its Software Trigger Bit which is implemented in the XGATE module.

XGATE Semaphore

A set of hardware flip-flops that can be exclusively set by either the S12X_CPU or the XGATE. (see 5.4.4/5-231)

XGATE Thread

A code sequence which is executed by the XGATE's RISC core after receiving an XGATE request.

XGATE Debug Mode

A special mode in which the XGATE's RISC core is halted for debug purposes. This mode enables the XGATE's debug features (see 5.6/5-233).

XGATE Software Error

The XGATE is able to detect a number of error conditions caused by erratic software (see 5.4.5/5-232). These error conditions will cause the XGATE to seize program execution and flag an Interrupt to the S12X_CPU.

Word

A 16 bit entity.

Byte

An 8 bit entity.

5.1.2 Features

The XGATE module includes these features:

- Data movement between various targets (i.e Flash, RAM, and peripheral modules)
- Data manipulation through built in RISC core
- Provides up to 112 XGATE channels

- — 104 hardware triggered channels
  - — 8 software triggered channels
- Hardware semaphores which are shared between the S12X_CPU and the XGATE module
- Able to trigger S12X_CPU interrupts upon completion of an XGATE transfer
- Software error detection to catch erratic application code

5.1.3 Modes of Operation

There are four run modes on S12X devices.

- Run mode, wait mode, stop mode

  The XGATE is able to operate in all of these three system modes. Clock activity will be automatically stopped when the XGATE module is idle.
- Freeze mode (BDM active)

  In freeze mode all clocks of the XGATE module may be stopped, depending on the module configuration (see Section 5.3.1.1, "XGATE Control Register (XGMCTL)").

5.1.4 Block Diagram

Figure Figure 5-1 shows a block diagram of the XGATE.

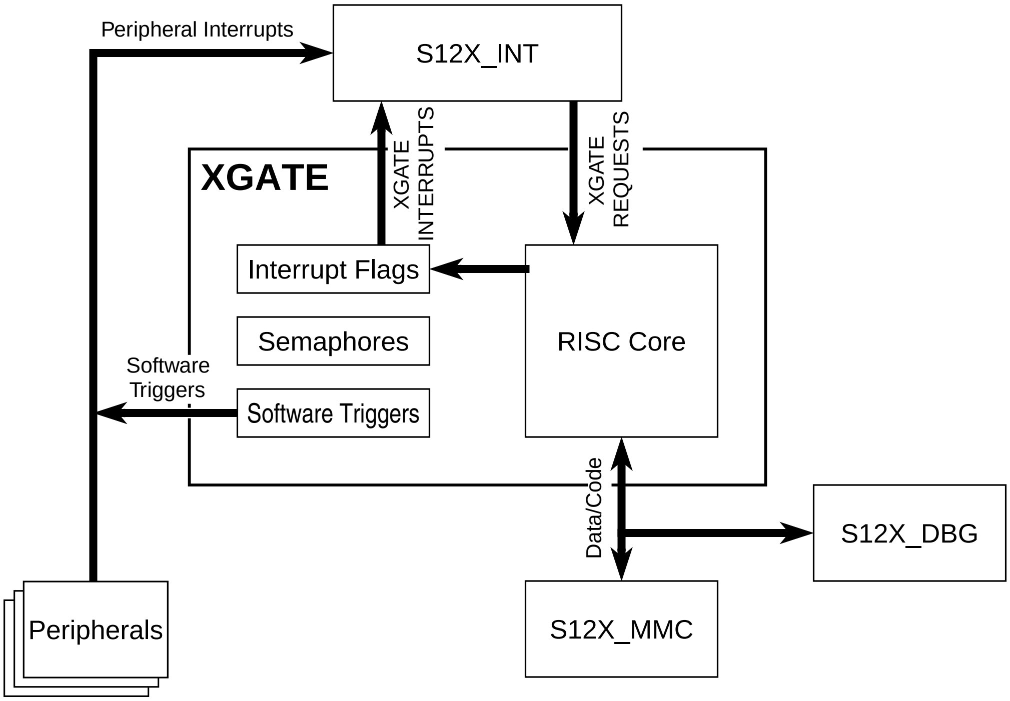


Figure 5-1. XGATE Block Diagram

5.2 External Signal Description

The XGATE module has no external pins.

5.3 Memory Map and Register Definition

This section provides a detailed description of address space and registers used by the XGATE module.

The memory map for the XGATE module is given below in Figure 5-2.The address listed for each register is the sum of a base address and an address offset. The base address is defined at the SoC level and the address offset is defined at the module level. Reserved registers read zero. Write accesses to the reserved registers have no effect.

5.3.1 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 XGMCTL	R	0	0	0	0	0	0	0	0	XGE	XGFRZ	XGDBG	XGSS	XG FACT	0	XG SWEIF	XGIE
	W	XGEM	XG FRZM	XG DBGM	XGSSM	XG FACTM	▨	XG SWEIFM	XGIEM						▨		
0x0002 XGMCHID	R									0	XGCHID[6:0]						
	W									▨	▨	▨	▨	▨	▨	▨	▨
0x0003 Reserved	R									▨	▨	▨	▨	▨	▨	▨	▨
	W																
0x0004 Reserved	R									▨	▨	▨	▨	▨	▨	▨	▨
	W																
0x0005 Reserved	R									▨	▨	▨	▨	▨	▨	▨	▨
	W																
0x0006 XGVBR	R	XGVBR[15:1]															0
	W																▨

▨ = Unimplemented or Reserved

Figure 5-2. XGATE Register Summary (Sheet 1 of 3)

		127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
0x0008 XGIF	R	0	0	0	0	0	0	0	XGIF_78	XGF_77	XGIF_76	XGIF_75	XGIF_74	XGIF_73	XGIF_72	XGIF_71	XGIF_70
	W	▨	▨	▨	▨	▨	▨	▨									

		111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
0x000A XGIF	R	XGIF_6F	XGIF_6E	XGIF_6D	XGIF_6C	XGIF_6B	XGIF_6A	XGIF_69	XGIF_68	XGF_67	XGIF_66	XGIF_65	XGIF_64	XGIF_63	XGIF_62	XGIF_61	XGIF_60
	W																

		95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
0x000C XGIF	R	XGIF_5F	XGIF_5E	XGIF_5D	XGIF_5C	XGIF_5B	XGIF_5A	XGIF_59	XGIF_58	XGF_57	XGIF_56	XGIF_55	XGIF_54	XGIF_53	XGIF_52	XGIF_51	XGIF_50
	W																

		79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
0x000E XGIF	R	XGIF_4F	XGIF_4E	XGIF_4D	XGIF_4C	XGIF_4B	XGIF_4A	XGIF_49	XGIF_48	XGF _47	XGIF_46	XGIF_45	XGIF_44	XGIF_43	XGIF_42	XGIF_41	XGIF_40
	W																

Register Name		63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
0x0010 XGIF	R	XGIF_3F	XGIF_3E	XGIF_3D	XGIF_3C	XGIF_3B	XGIF_3A	XGIF_39	XGIF_38	XGF _37	XGIF_36	XGIF_35	XGIF_34	XGIF_33	XGIF_32	XGIF_31	XGIF_30
	W																

		47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
0x0012 XGIF	R	XGIF_2F	XGIF_2E	XGIF_2D	XGIF_2C	XGIF_2B	XGIF_2A	XGIF_29	XGIF_28	XGF _27	XGIF_26	XGIF_25	XGIF_24	XGIF_23	XGIF_22	XGIF_21	XGIF_20
	W																

		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0014 XGIF	R	XGIF_1F	XGIF_1E	XGIF_1D	XGIF_1C	XGIF_1B	XGIF_1A	XGIF_19	XGIF_18	XGF _17	XGIF_16	XGIF_15	XGIF_14	XGIF_13	XGIF_12	XGIF_11	XGIF_10
	W																

		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0016 XGIF	R	XGIF_0F	XGIF_0E	XGIF_0D	XGIF_0C	XGIF_0B	XGIF_0A	XGIF_09	0	0	0	0	0	0	0	0	0
	W								▨	▨	▨	▨	▨	▨	▨	▨	▨

▨ = Unimplemented or Reserved

Figure 5-2. XGATE Register Summary (Sheet 2 of 3)

		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0018 XGSWTM	R	0	0	0	0	0	0	0	0	XGSWT[7:0]							
	W	XGSWTM[7:0]															
0x001A XGSEMM	R	0	0	0	0	0	0	0	0	XGSEM[7:0]							
	W	XGSEMM[7:0]															
0x001C Reserved	R									▨	▨	▨	▨	▨	▨	▨	▨
	W																
0x001D XGCCR	R									0	0	0	0	XGN	XGZ	XGV	XGC
	W									▨	▨	▨	▨				
0x001E XGPC	R	XGPC															
	W																
0x0020 Reserved	R									▨	▨	▨	▨	▨	▨	▨	▨
	W																
0x0021 Reserved	R									▨	▨	▨	▨	▨	▨	▨	▨
	W																
0x0022 XGR1	R	XGR1															
	W																
0x0024 XGR2	R	XGR2															
	W																
0x0026 XGR3	R	XGR3															
	W																
0x0028 XGR4	R	XGR4															
	W																
0x002A XGR5	R	XGR5															
	W																
0x002C XGR6	R	XGR6															
	W																
0x002E XGR7	R	XGR7															
	W																

▨ = Unimplemented or Reserved

Figure 5-2. XGATE Register Summary (Sheet 3 of 3)

5.3.1.1 XGATE Control Register (XGMCTL)

All module level switches and flags are located in the module control register Figure 5-3.

Module Base +0x00000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	XGE	XGFRZ	XGDBG	XGSS	XGFACT	0	XG SWEIF	XGIE
W	XGEM	XG FRZM	XG DBGM	XG SSM	XG FACTM	▨	XG SWEIFM	XGIEM						▨		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 5-3. XGATE Control Register (XGMCTL)

Read: Anytime

Write: Anytime

Table 5-1. XGMCTL Field Descriptions (Sheet 1 of 3)

Field	Description
15 XGEM	**XGE Mask** — This bit controls the write access to the XGE bit. The XGE bit can only be set or cleared if a "1" is written to the XGEM bit in the same register access. Read: This bit will always read "0". Write: 0 Disable write access to the XGE in the same bus cycle 1 Enable write access to the XGE in the same bus cycle
14 XGFRZM	**XGFRZ Mask** — This bit controls the write access to the XGFRZ bit. The XGFRZ bit can only be set or cleared if a "1" is written to the XGFRZM bit in the same register access. Read: This bit will always read "0". Write: 0 Disable write access to the XGFRZ in the same bus cycle 1 Enable write access to the XGFRZ in the same bus cycle
13 XGDBGM	**XGDBG Mask** — This bit controls the write access to the XGDBG bit. The XGDBG bit can only be set or cleared if a "1" is written to the XGDBGM bit in the same register access. Read: This bit will always read "0". Write: 0 Disable write access to the XGDBG in the same bus cycle 1 Enable write access to the XGDBG in the same bus cycle
12 XGSSM	**XGSS Mask** — This bit controls the write access to the XGSS bit. The XGSS bit can only be set or cleared if a "1" is written to the XGSSM bit in the same register access. Read: This bit will always read "0". Write: 0 Disable write access to the XGSS in the same bus cycle 1 Enable write access to the XGSS in the same bus cycle

Table 5-1. XGMCTL Field Descriptions (Sheet 2 of 3)

Field	Description
11 XGFACTM	**XGFACT Mask** — This bit controls the write access to the XGFACT bit. The XGFACT bit can only be set or cleared if a "1" is written to the XGFACTM bit in the same register access. Read: This bit will always read "0". Write: 0 Disable write access to the XGFACT in the same bus cycle 1 Enable write access to the XGFACT in the same bus cycle
9 XGSWEIFM	**XGSWEIF Mask** — This bit controls the write access to the XGSWEIF bit. The XGSWEIF bit can only be cleared if a "1" is written to the XGSWEIFM bit in the same register access. Read: This bit will always read "0". Write: 0 Disable write access to the XGSWEIF in the same bus cycle 1 Enable write access to the XGSWEIF in the same bus cycle
8 XGIEM	**XGIE Mask** — This bit controls the write access to the XGIE bit. The XGIE bit can only be set or cleared if a "1" is written to the XGIEM bit in the same register access. Read: This bit will always read "0". Write: 0 Disable write access to the XGIE in the same bus cycle 1 Enable write access to the XGIE in the same bus cycle
7 XGE	**XGATE Module Enable** — This bit enables the XGATE module. If the XGATE module is disabled, pending XGATE requests will be ignored. The thread that is executed by the RISC core while the XGE bit is cleared will continue to run. Read: 0 XGATE module is disabled 1 XGATE module is enabled Write: 0 Disable XGATE module 1 Enable XGATE module
6 XGFRZ	**Halt XGATE in Freeze Mode** — The XGFRZ bit controls the XGATE operation in Freeze Mode (BDM active). Read: 0 RISC core operates normally in Freeze (BDM active) 1 RISC core stops in Freeze Mode (BDM active) Write: 0 Don’t stop RISC core in Freeze Mode (BDM active) 1 Stop RISC core in Freeze Mode (BDM active)
5 XGDBG	**XGATE Debug Mode** — This bit indicates that the XGATE is in Debug Mode (see Section 5.6, “Debug Mode”). Debug Mode can be entered by Software Breakpoints (BRK instruction), Tagged or Forced Breakpoints (see **S12X_DBG Section**), or by writing a "1" to this bit. Read: 0 RISC core is not in Debug Mode 1 RISC core is in Debug Mode Write: 0 Leave Debug Mode 1 Enter Debug Mode **Note:** Freeze Mode and Software Error Interrupts have no effect on the XGDBG bit.

Table 5-1. XGMCTL Field Descriptions (Sheet 3 of 3)

Field	Description
4 XGSS	**XGATE Single Step** — This bit forces the execution of a single instruction if the XGATE is in DEBUG Mode and no software error has occurred (XGSWEIF cleared). Read: 0 No single step in progress 1 Single step in progress Write 0 No effect 1 Execute a single RISC instruction **Note:** Invoking a Single Step will cause the XGATE to temporarily leave Debug Mode until the instruction has been executed.
3 XGFACT	**Fake XGATE Activity** — This bit forces the XGATE to flag activity to the MCU even when it is idle. When it is set the MCU will never enter system stop mode which assures that peripheral modules will be clocked during XGATE idle periods Read: 0 XGATE will only flag activity if it is not idle or in debug mode. 1 XGATE will always signal activity to the MCU. Write: 0 Only flag activity if not idle or in debug mode. 1 Always signal XGATE activity.
1 XGSWEIF	**XGATE Software Error Interrupt Flag** — This bit signals a pending Software Error Interrupt. It is set if the RISC core detects an error condition (see Section 5.4.5, “Software Error Detection”). The RISC core is stopped while this bit is set. Clearing this bit will terminate the current thread and cause the XGATE to become idle. Read: 0 Software Error Interrupt is not pending 1 Software Error Interrupt is pending if XGIE is set Write: 0 No effect 1 Clears the XGSWEIF bit
0 XGIE	**XGATE Interrupt Enable** — This bit acts as a global interrupt enable for the XGATE module Read: 0 All XGATE interrupts disabled 1 All XGATE interrupts enabled Write: 0 Disable all XGATE interrupts 1 Enable all XGATE interrupts

5.3.1.2 XGATE Channel ID Register (XGCHID)

The XGATE channel ID register (Figure 5-4) shows the identifier of the XGATE channel that is currently active. This register will read “$00” if the XGATE module is idle. In debug mode this register can be used to start and terminate threads (see Section 5.6.1, “Debug Features”).

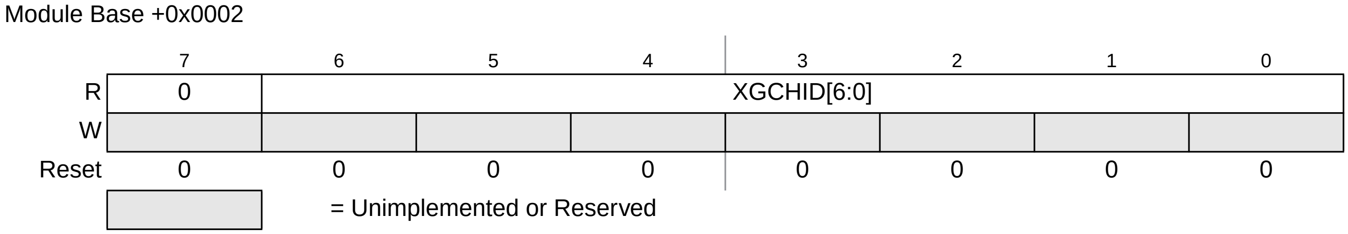


Figure 5-4. XGATE Channel ID Register (XGCHID)

Read: Anytime

Write: In Debug Mode

Table 5-2. XGCHID Field Descriptions

Field	Description
6–0 XGCHID[6:0]	**Request Identifier** — ID of the currently active channel

5.3.1.3 XGATE Vector Base Address Register (XGVBR)

The vector base address register (Figure 5-5 and Figure 5-6) determines the location of the XGATE vector block.

Figure 5-5. XGATE Vector Base Address Register (XGVBR)

Read: Anytime

Write: Only if the module is disabled (XGE = 0) and idle (XGCHID = $00))

Table 5-3. XGVBR Field Descriptions

Field	Description
15–1 XBVBR[15:1]	**Vector Base Address** — The XGVBR register holds the start address of the vector block in the XGATE memory map.

5.3.1.4 XGATE Channel Interrupt Flag Vector (XGIF)

The interrupt flag vector (Figure 5-6) provides access to the interrupt flags bits of each channel. Each flag may be cleared by writing a "1" to its bit location.

Module Base +0x0008

	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
R	0	0	0	0	0	0	0	XGIF_78	XGF_77	XGIF_76	XGIF_75	XGIF_74	XGIF_73	XGIF_72	XGIF_71	XGIF_70
W	▨	▨	▨	▨	▨	▨	▨									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
R	XGIF_6F	XGIF_6E	XGIF_6D	XGIF_6C	XGIF_6B	XGIF_6A	XGIF_69	XGIF_68	XGF_67	XGIF_66	XGIF_65	XGIF_64	XGIF_63	XGIF_62	XGIF_61	XGIF_60
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
R	XGIF_5F	XGIF_5E	XGIF_5D	XGIF_5C	XGIF_5B	XGIF_5A	XGIF_59	XGIF_58	XGF_57	XGIF_56	XGIF_55	XGIF_54	XGIF_53	XGIF_52	XGIF_51	XGIF_50
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
R	XGIF_4F	XGIF_4E	XGIF_4D	XGIF_4C	XGIF_4B	XGIF_4A	XGIF_49	XGIF_48	XGF _47	XGIF_46	XGIF_45	XGIF_44	XGIF_43	XGIF_42	XGIF_41	XGIF_40
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
R	XGIF_3F	XGIF_3E	XGIF_3D	XGIF_3C	XGIF_3B	XGIF_3A	XGIF_39	XGIF_38	XGF _37	XGIF_36	XGIF_35	XGIF_34	XGIF_33	XGIF_32	XGIF_31	XGIF_30
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
R	XGIF_2F	XGIF_2E	XGIF_2D	XGIF_2C	XGIF_2B	XGIF_2A	XGIF_29	XGIF_28	XGF _27	XGIF_26	XGIF_25	XGIF_24	XGIF_23	XGIF_22	XGIF_21	XGIF_20
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XGIF_1F	XGIF_1E	XGIF_1D	XGIF_1C	XGIF_1B	XGIF_1A	XGIF_19	XGIF_18	XGF _17	XGIF_16	XGIF_15	XGIF_14	XGIF_13	XGIF_12	XGIF_11	XGIF_10
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	XGIF_0F	XGIF_0E	XGIF_0D	XGIF_0C	XGIF_0B	XGIF_0A	XGIF_09	0	0	0	0	0	0	0	0	0
W								▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 5-6. XGATE Channel Interrupt Flag Vector (XGIF)

Read: Anytime

Write: Anytime

Table 5-4. XGIV Field Descriptions

Field	Description
120–9 XGIF_78 - XGIF_09	**Channel Interrupt Flags** — These bits signal pending channel interrupts. They can only be set by the RISC core. Each flag can be cleared by writing a "1" to its bit location. Unimplemented interrupt flags will always read "0". Refer to Section “Interrupts” of the **SoC Guide** for a list of implemented Interrupts. Read: 0 Channel interrupt is not pending 1 Channel interrupt is pending if XGIE is set Write: 0 No effect 1 Clears the interrupt flag

NOTE

Suggested Mnemonics for accessing the interrupt flag vector on a word basis are:

XGIF_7F_70 (XGIF[127:112]),
XGIF_6F_60 (XGIF[111:96]),
XGIF_5F_50 (XGIF[95:80]),
XGIF_4F_40 (XGIF[79:64]),
XGIF_3F_30 (XGIF[63:48]),
XGIF_2F_20 (XGIF[47:32]),
XGIF_1F_10 (XGIF[31:16]),
XGIF_0F_00 (XGIF[15:0])

5.3.1.5 XGATE Software Trigger Register (XGSWT)

The eight software triggers of the XGATE module can be set and cleared through the XGATE software trigger register (Figure 5-7). The upper byte of this register, the software trigger mask, controls the write access to the lower byte, the software trigger bits. These bits can be set or cleared if a "1" is written to the associated mask in the same bus cycle.

Module Base +0x00018

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	XGSWT[7:0]							
W	XGSWTM[7:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-7. XGATE Software Trigger Register (XGSWT)

Read: Anytime

Write: Anytime

Table 5-5. XGSWT Field Descriptions

Field	Description
15–8 XGSWTM[7:0]	**Software Trigger Mask** — These bits control the write access to the XGSWT bits. Each XGSWT bit can only be written if a "1" is written to the corresponding XGSWTM bit in the same access. Read: These bits will always read "0". Write: 0 Disable write access to the XGSWT in the same bus cycle 1 Enable write access to the corresponding XGSWT bit in the same bus cycle
7–0 XGSWT[7:0]	**Software Trigger Bits** — These bits act as interrupt flags that are able to trigger XGATE software channels. They can only be set and cleared by software. Read: 0 No software trigger pending 1 Software trigger pending if the XGIE bit is set Write: 0 Clear Software Trigger 1 Set Software Trigger

NOTE

The XGATE channel IDs that are associated with the eight software triggers are determined on chip integration level. (see Section “Interrupts” of the **Soc Guide**)

XGATE software triggers work like any peripheral interrupt. They can be used as XGATE requests as well as S12X_CPU interrupts. The target of the software trigger must be selected in the S12X_INT module.

5.3.1.6 XGATE Semaphore Register (XGSEM)

The XGATE provides a set of eight hardware semaphores that can be shared between the S12X_CPU and the XGATE RISC core. Each semaphore can either be unlocked, locked by the S12X_CPU or locked by the RISC core. The RISC core is able to lock and unlock a semaphore through its SSEM and CSEM instructions. The S12X_CPU has access to the semaphores through the XGATE semaphore register (Figure 5-8). Refer to section Section 5.4.4, “Semaphores” for details.

Module Base +0x0001A

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	XGSEM[7:0]							
W	XGSEMM[7:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-8. XGATE Semaphore Register (XGSEM)

Read: Anytime

Write: Anytime (see Section 5.4.4, “Semaphores”)

Table 5-6. XGSEM Field Descriptions

Field	Description
15–8 XGSEMM[7:0]	**Semaphore Mask** — These bits control the write access to the XGSEM bits. Read: These bits will always read "0". Write: 0 Disable write access to the XGSEM in the same bus cycle 1 Enable write access to the XGSEM in the same bus cycle
7–0 XGSEM[7:0]	**Semaphore Bits** — These bits indicate whether a semaphore is locked by the S12X_CPU. A semaphore can be attempted to be set by writing a "1" to the XGSEM bit and to the corresponding XGSEMM bit in the same write access. Only unlocked semaphores can be set. A semaphore can be cleared by writing a "0" to the XGSEM bit and a "1" to the corresponding XGSEMM bit in the same write access. Read: 0 Semaphore is unlocked or locked by the RISC core 1 Semaphore is locked by the S12X_CPU Write: 0 Clear semaphore if it was locked by the S12X_CPU 1 Attempt to lock semaphore by the S12X_CPU

5.3.1.7 XGATE Condition Code Register (XGCCR)

The XGCCR register (Figure 5-9) provides access to the RISC core’s condition code register.

Module Base +0x001D

	7	6	5	4	3	2	1	0
R	0	0	0	0	XGN	XGZ	XGV	XGC
W	▨	▨	▨	▨				
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 5-9. XGATE Condition Code Register (XGCCR)

Read: In debug mode if unsecured

Write: In debug mode if unsecured

Table 5-7. XGCCR Field Descriptions

Field	Description
3 XGN	**Sign Flag** — The RISC core’s Sign flag
2 XGZ	**Zero Flag** — The RISC core’s Zero flag
1 XGV	**Overflow Flag** — The RISC core’s Overflow flag
0 XGC	**Carry Flag** — The RISC core’s Carry flag

5.3.1.8 XGATE Program Counter Register (XGPC)

The XGPC register (Figure 5-10) provides access to the RISC core’s program counter.

Module Base +0x0001E

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

Figure 5-10. XGATE Program Counter Register (XGPC)

Figure 5-11.

Read: In debug mode if unsecured

Write: In debug mode if unsecured

Table 5-8. XGPC Field Descriptions

Field	Description
15–0 XGPC[15:0]	**Program Counter** — The RISC core’s program counter

5.3.1.9 XGATE Register 1 (XGR1)

The XGR1 register (Figure 5-12) provides access to the RISC core’s register 1.

Module Base +0x00022

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XGR1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-12. XGATE Register 1 (XGR1)

Read: In debug mode if unsecured

Write: In debug mode if unsecured

Table 5-9. XGR1 Field Descriptions

Field	Description
15–0 XGR1[15:0]	**XGATE Register 1** — The RISC core’s register 1

5.3.1.10 XGATE Register 2 (XGR2)

The XGR2 register (Figure 5-13) provides access to the RISC core’s register 2.

Module Base +0x00024

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XGR2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-13. XGATE Register 2 (XGR2)

Read: In debug mode if unsecured

Write: In debug mode if unsecured

Table 5-10. XGR2 Field Descriptions

Field	Description
15–0 XGR2[15:0]	**XGATE Register 2** — The RISC core’s register 2

5.3.1.11 XGATE Register 3 (XGR3)

The XGR3 register (Figure 5-14) provides access to the RISC core’s register 3.

Module Base +0x00026

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XGR3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-14. XGATE Register 3 (XGR3)

Read: In debug mode if unsecured

Write: In debug mode if unsecured

Table 5-11. XGR3 Field Descriptions

Field	Description
15–0 XGR3[15:0]	**XGATE Register 3** — The RISC core’s register 3

5.3.1.12 XGATE Register 4 (XGR4)

The XGR4 register (Figure 5-15) provides access to the RISC core’s register 4.

Module Base +0x00028

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XGR4															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-15. XGATE Register 4 (XGR4)

Read: In debug mode if unsecured

Write: In debug mode if unsecured

Table 5-12. XGR4 Field Descriptions

Field	Description
15–0 XGR4[15:0]	**XGATE Register 4** — The RISC core’s register 4

5.3.1.13 XGATE Register 5 (XGR5)

The XGR5 register (Figure 5-16) provides access to the RISC core’s register 5.

Module Base +0x0002A

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XGR5															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-16. XGATE Register 5 (XGR5)

Read: In debug mode if unsecured

Write: In debug mode if unsecured

Table 5-13. XGR5 Field Descriptions

Field	Description
15–0 XGR5[15:0]	**XGATE Register 5** — The RISC core’s register 5

5.3.1.14 XGATE Register 6 (XGR6)

The XGR6 register (Figure 5-17) provides access to the RISC core's register 6.

Module Base +0x0002C

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XGR6															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-17. XGATE Register 6 (XGR6)

Read: In debug mode if unsecured

Write: In debug mode if unsecured

Table 5-14. XGR6 Field Descriptions

Field	Description
15–0 XGR6[15:0]	**XGATE Register 6** — The RISC core's register 6

5.3.1.15 XGATE Register 7 (XGR7)

The XGR7 register (Figure 5-18) provides access to the RISC core's register 7.

Module Base +0x0002E

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XGR7															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-18. XGATE Register 7 (XGR7)

Read: In debug mode if unsecured

Write: In debug mode if unsecured

Table 5-15. XGR7 Field Descriptions

Field	Description
15–0 XGR7[15:0]	**XGATE Register 7** — The RISC core's register 7

5.4 Functional Description

The core of the XGATE module is a RISC processor which is able to access the MCU's internal memories and peripherals (see Figure 5-1). The RISC processor always remains in an idle state until it is triggered by an XGATE request. Then it executes a code sequence that is associated with the request and optionally triggers an interrupt to the S12X_CPU upon completion. Code sequences are not interruptible. A new XGATE request can only be serviced when the previous sequence is finished and the RISC core becomes idle.

The XGATE module also provides a set of hardware semaphores which are necessary to ensure data consistency whenever RAM locations or peripherals are shared with the S12X_CPU.

The following sections describe the components of the XGATE module in further detail.

5.4.1 XGATE RISC Core

The RISC core is a 16 bit processor with an instruction set that is well suited for data transfers, bit manipulations, and simple arithmetic operations (see Section 5.8, "Instruction Set").

It is able to access the MCU's internal memories and peripherals without blocking these resources from the S12X_CPU[1]. Whenever the S12X_CPU and the RISC core access the same resource, the RISC core will be stalled until the resource becomes available again[1].

The XGATE offers a high access rate to the MCU's internal RAM. Depending on the bus load, the RISC core can perform up to two RAM accesses per S12X_CPU bus cycle.

Bus accesses to peripheral registers or flash are slower. A transfer rate of one bus access per S12X_CPU cycle can not be exceeded.

The XGATE module is intended to execute short interrupt service routines that are triggered by peripheral modules or by software.

5.4.2 Programmer's Model

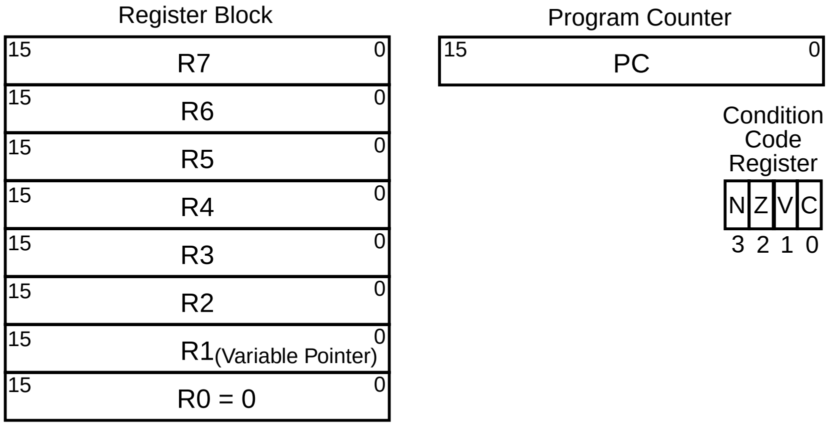


Figure 5-19. Programmer's Model

---

[1] With the exception of PRR registers (see Section "S12X_MMC").

The programmer's model of the XGATE RISC core is shown in Figure 5-19. The processor offers a set of seven general purpose registers (R1 - R7), which serve as accumulators and index registers. An additional eighth register (R0) is tied to the value "$0000". Register R1 has an additional functionality. It is preloaded with the initial variable pointer of the channel's service request vector (see Figure 5-20). The initial content of the remaining general purpose registers is undefined.

The 16 bit program counter allows the addressing of a 64 kbyte address space.

The condition code register contains four bits: the sign bit (S), the zero flag (Z), the overflow flag (V), and the carry bit (C). The initial content of the condition code register is undefined.

5.4.3 Memory Map

The XGATE's RISC core is able to access an address space of 64K bytes. The allocation of memory blocks within this address space is determined on chip level. Refer to the **S12X_MMC Section** for a detailed information.

The XGATE vector block assigns a start address and a variable pointer to each XGATE channel. Its position in the XGATE memory map can be adjusted through the XGVBR register (see Section 5.3.1.3, "XGATE Vector Base Address Register (XGVBR)"). Figure 5-20 shows the layout of the vector block. Each vector consists of two 16 bit words. The first contains the start address of the service routine. This value will be loaded into the program counter before a service routine is executed. The second word is a pointer to the service routine's variable space. This value will be loaded into register R1 before a service routine is executed.

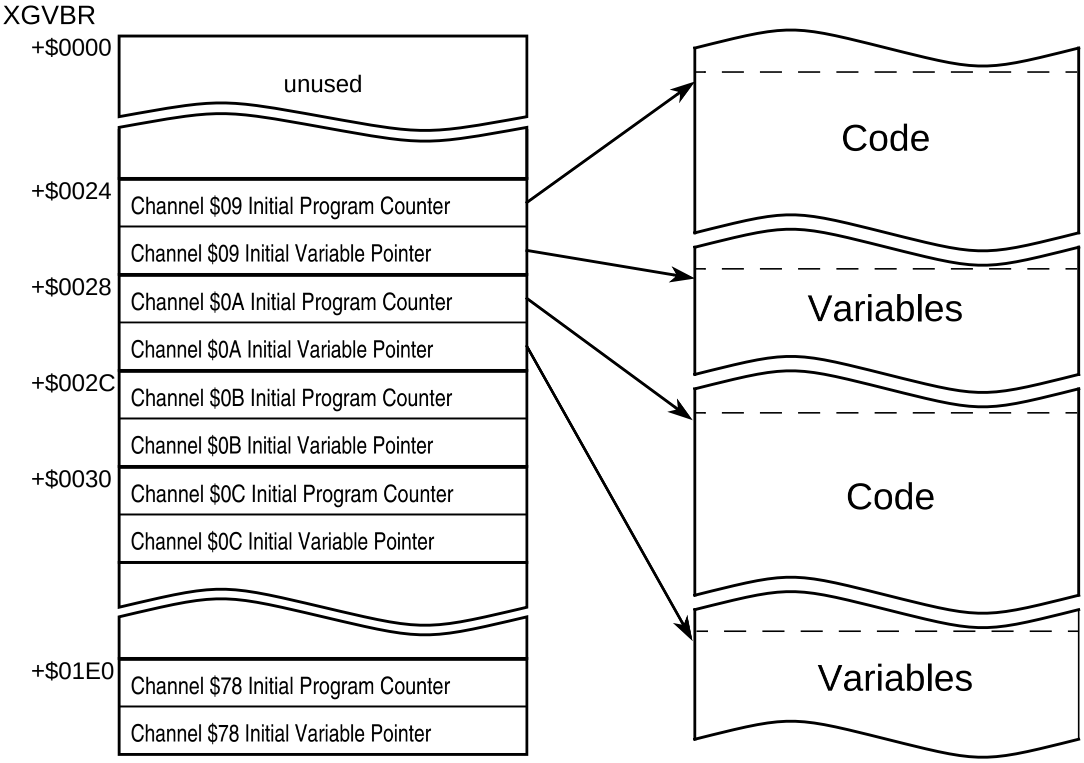


Figure 5-20. XGATE Vector Block

5.4.4 Semaphores

The XGATE module offers a set of eight hardware semaphores. These semaphores provide a mechanism to protect system resources that are shared between two concurrent threads of program execution; one thread running on the S12X_CPU and one running on the XGATE RISC core.

Each semaphore can only be in one of the three states: “Unlocked”, “Locked by S12X_CPU”, and “Locked by XGATE”. The S12X_CPU can check and change a semaphore’s state through the XGATE semaphore register (XGSEM, see Section 5.3.1.6, “XGATE Semaphore Register (XGSEM)”). The RISC core does this through its SSEM and CSEM instructions.

Figure 5-21 illustrates the valid state transitions.

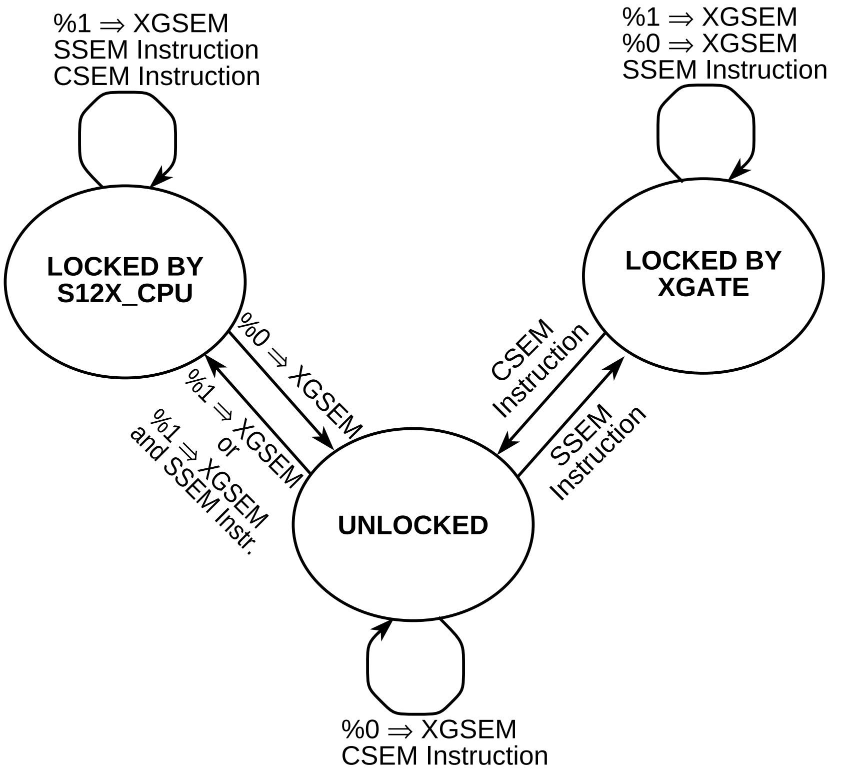


Figure 5-21. Semaphore State Transitions

Figure 5-22 gives an example of the typical usage of the XGATE hardware semaphores.

Two concurrent threads are running on the system. One is running on the S12X_CPU and the other is running on the RISC core. They both have a critical section of code that accesses the same system resource. To guarantee that the system resource is only accessed by one thread at a time, the critical code sequence must be embedded in a semaphore lock/release sequence as shown.

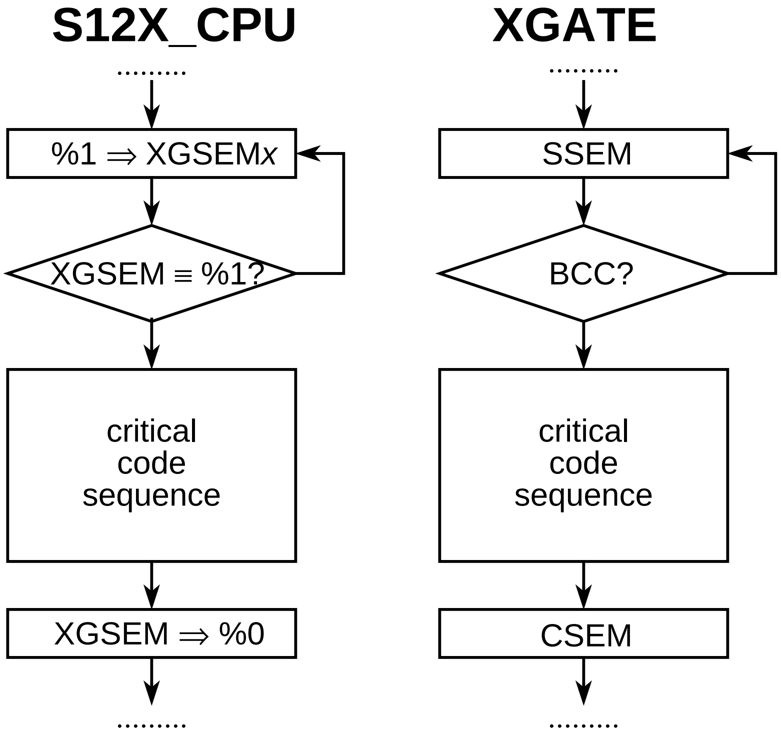


Figure 5-22. Algorithm for Locking and Releasing Semaphores

5.4.5 Software Error Detection

The XGATE module will immediately terminate program execution after detecting an error condition caused by erratic application code. There are three error conditions:

- Execution of an illegal opcode
- Illegal vector or opcode fetches
- Illegal load or store accesses

All opcodes which are not listed in section Section 5.8, “Instruction Set” are illegal opcodes. Illegal vector and opcode fetches as well as illegal load and store accesses are defined on chip level. Refer to the **S12X_MMC Section** for a detailed information.

5.5 Interrupts

5.5.1 Incoming Interrupt Requests

XGATE threads are triggered by interrupt requests which are routed to the XGATE module (see S12X_INT Section). Only a subset of the MCU's interrupt requests can be routed to the XGATE. Which specific interrupt requests these are and which channel ID they are assigned to is documented in Section “Interrupts” of the **SoC Guide**.

5.5.2 Outgoing Interrupt Requests

There are three types of interrupt requests which can be triggered by the XGATE module:

5. Channel interrupts

   For each XGATE channel there is an associated interrupt flag in the XGATE interrupt flag vector (XGIF, see Section 5.3.1.4, “XGATE Channel Interrupt Flag Vector (XGIF)”). These flags can be set through the "SIF" instruction by the RISC core. They are typically used to flag an interrupt to the S12X_CPU when the XGATE has completed one of its tasks.
6. Software triggers

   Software triggers are interrupt flags, which can be set and cleared by software (see Section 5.3.1.5, “XGATE Software Trigger Register (XGSWT)”). They are typically used to trigger XGATE tasks by the S12X_CPU software. However these interrupts can also be routed to the S12X_CPU (see **S12X_INT Section**) and triggered by the XGATE software.
7. Software error interrupt

   The software error interrupt signals to the S12X_CPU the detection of an error condition in the XGATE application code (see Section 5.4.5, “Software Error Detection”).

All XGATE interrupts can be disabled by the XGIE bit in the XGATE module control register (XGMCTL, see Section 5.3.1.1, “XGATE Control Register (XGMCTL)”).

5.6 Debug Mode

The XGATE debug mode is a feature to allow debugging of application code.

5.6.1 Debug Features

In debug mode the RISC core will be halted and the following debug features will be enabled:

- Read and Write accesses to RISC core registers (XGCCR, XGPC, XGR1–XGR7)[1]

  All RISC core registers can be modified. Leaving debug mode will cause the RISC core to continue program execution with the modified register values.

---

[1] Only possible if MCU is unsecured

- Single Stepping

  Writing a "1" to the XGSS bit will call the RISC core to execute a single instruction. All RISC core registers will be updated accordingly.
- Write accesses to the XGCHID register

  Three operations can be performed by writing to the XGCHID register:
  - Change of channel ID

    If a non-zero value is written to the XGCHID while a thread is active (XGCHID ≠ $00), then the current channel ID will be changed without any influence on the program counter or the other RISC core registers.
  - Start of a thread

    If a non-zero value is written to the XGCHID while the XGATE is idle (XGCHID = $00), then the thread that is associated with the new channel ID will be executed upon leaving debug mode.
  - Termination of a thread

    If zero is written to the XGCHID while a thread is active (XGCHID ≠ $00), then the current thread will be terminated and the XGATE will become idle.

5.6.2 Entering Debug Mode

Debug mode can be entered in four ways:

1. Setting XGDBG to "1"

   Writing a "1" to XGDBG and XGDBGM in the same write access causes the XGATE to enter debug mode upon completion of the current instruction.

   NOTE

   After writing to the XGDBG bit the XGATE will not immediately enter debug mode. Depending on the instruction that is executed at this time there may be a delay of several clock cycles. The XGDBG will read "0" until debug mode is entered.
2. Software breakpoints

   XGATE programs which are stored in the internal RAM allow the use of software breakpoints. A software breakpoint is set by replacing an instruction of the program code with the "BRK" instruction.

   As soon as the program execution reaches the "BRK" instruction, the XGATE enters debug mode. Additionally a software breakpoint request is sent to the S12X_DBG module (see section 4.9 of the **S12X_DBG Section**).

   Upon entering debug mode, the program counter will point to the "BRK" instruction. The other RISC core registers will hold the result of the previous instruction.

   To resume program execution, the "BRK" instruction must be replaced by the original instruction before leaving debug mode.

3. Tagged Breakpoints

   The S12X_DBG module is able to place tags on fetched opcodes. The XGATE is able to enter debug mode right before a tagged opcode is executed (see section 4.9 of the **S12X_DBG Section**). Upon entering debug mode, the program counter will point to the tagged instruction. The other RISC core registers will hold the result of the previous instruction.

4. Forced Breakpoints

   Forced breakpoints are triggered by the S12X_DBG module (see section 4.9 of the **S12X_DBG Section**). When a forced breakpoint occurs, the XGATE will enter debug mode upon completion of the current instruction.

5.6.3 Leaving Debug Mode

Debug mode can only be left by setting the XGDBG bit to "0". If a thread is active (XGCHID has not been cleared in debug mode), program execution will resume at the value of XGPC.

5.7 Security

In order to protect XGATE application code on secured S12X devices, a few restrictions in the debug features have been made. These are:

- Registers XGCCR, XGPC, and XGR1–XGR7 will read zero on a secured device
- Registers XGCCR, XGPC, and XGR1–XGR7 can not be written on a secured device
- Single stepping is not possible on a secured device

5.8 Instruction Set

5.8.1 Addressing Modes

For the ease of implementation the architecture is a strict Load/Store RISC machine, which means all operations must have one of the eight general purpose registers R0 … R7 as their source as well their destination.

All word accesses must work with a word aligned address, that is A[0] = 0!

5.8.1.1 Naming Conventions

RD	Destination register, allowed range is R0–R7
RD.L	Low byte of the destination register, bits [7:0]
RD.H	High byte of the destination register, bits [15:8]
RS, RS1, RS2	Source register, allowed range is R0–R7
RS.L, RS1.L, RS2.L	Low byte of the source register, bits [7:0]
RS.H, RS1.H, RS2.H	High byte of the source register, bits[15:8]
RB	Base register for indexed addressing modes, allowed range is R0–R7
RI	Offset register for indexed addressing modes with register offset, allowed range is R0–R7
RI+	Offset register for indexed addressing modes with register offset and post-increment, Allowed range is R0–R7 (R0+ is equivalent to R0)
–RI	Offset register for indexed addressing modes with register offset and pre-decrement, Allowed range is R0–R7 (–R0 is equivalent to R0)

NOTE

Even though register R1 is intended to be used as a pointer to the variable segment, it may be used as a general purpose data register as well.

Selecting R0 as destination register will discard the result of the instruction. Only the condition code register will be updated

5.8.1.2 Inherent Addressing Mode (INH)

Instructions that use this addressing mode either have no operands or all operands are in internal XGATE registers:.

Examples

```
BRK
RTS
```

5.8.1.3 Immediate 3-Bit Wide (IMM3)

Operands for immediate mode instructions are included in the instruction stream and are fetched into the instruction queue along with the rest of the 16 bit instruction. The ’#’ symbol is used to indicate an immediate addressing mode operand. This address mode is used for semaphore instructions.

Examples:

```
CSEM     #1       ; Unlock semaphore 1
SSEM     #3       ; Lock Semaphore 3
```

5.8.1.4 Immediate 4 Bit Wide (IMM4)

The 4 bit wide immediate addressing mode is supported by all shift instructions.

RD = RD $*$ imm4

Examples:

```
LSL     R4,#1    ; R4 = R4 << 1; shift register R4 by 1 bit to the left
LSR     R4,#3    ; R4 = R4 >> 3; shift register R4 by 3 bits to the right
```

5.8.1.5 Immediate 8 Bit Wide (IMM8)

The 8 bit wide immediate addressing mode is supported by four major commands (ADD, SUB, LD, CMP).

RD = RD $*$ imm8

Examples:

```
ADDL    R1,#1    ; adds an 8 bit value to register R1
SUBL    R2,#2    ; subtracts an 8 bit value from register R2
LDH     R3,#3    ; loads an 8 bit immediate into the high byte of Register R3
CMPL    R4,#4    ; compares the low byte of register R4 with an immediate value
```

5.8.1.6 Immediate 16 Bit Wide (IMM16)

The 16 bit wide immediate addressing mode is a construct to simplify assembler code. Instructions which offer this mode are translated into two opcodes using the eight bit wide immediate addressing mode.

RD = RD $*$ imm16

Examples:

```
LDW     R4,#$1234         ; translated to LDL R4,#$34; LDH R4,#$12
ADD     R4,#$5678         ; translated to ADDL R4,#$78; ADDH R4,#$56
```

5.8.1.7 Monadic Addressing (MON)

In this addressing mode only one operand is explicitly given. This operand can either be the source (f(RD)), the target (RD = f()), or both source and target of the operation (RD = f(RD)).

Examples:

```
JAL     R1       ; PC = R1, R1 = PC+2
SIF     R2       ; Trigger IRQ associated with the channel number in R2.L
```

5.8.1.8 Dyadic Addressing (DYA)

In this mode the result of an operation between two registers is stored in one of the registers used as operands.

RD = RD ∗ RS is the general register to register format, with register RD being the first operand and RS the second. RD and RS can be any of the 8 general purpose registers R0 … R7. If R0 is used as the destination register, only the condition code flags are updated. This addressing mode is used only for shift operations with a variable shift value

Examples:

```
LSL      R4,R5    ; R4 = R4 << R5
LSR      R4,R5    ; R4 = R4 >> R5
```

5.8.1.9 Triadic Addressing (TRI)

In this mode the result of an operation between two or three registers is stored into a third one. RD = RS1 ∗ RS2 is the general format used in the order RD, RS1, RS1. RD, RS1, RS2 can be any of the 8 general purpose registers R0 … R7. If R0 is used as the destination register RD, only the condition code flags are updated. This addressing mode is used for all arithmetic and logical operations.

Examples:

```
ADC      R5,R6,R7          ; R5 = R6 + R7 + Carry
SUB      R5,R6,R7          ; R5 = R6 - R7
```

5.8.1.10 Relative Addressing 9-Bit Wide (REL9)

A 9-bit signed word address offset is included in the instruction word. This addressing mode is used for conditional branch instructions.

Examples:

```
BCC      REL9              ; PC = PC + 2 + (REL9 << 1)
BEQ      REL9              ; PC = PC + 2 + (REL9 << 1)
```

5.8.1.11 Relative Addressing 10-Bit Wide (REL10)

An 11-bit signed word address offset is included in the instruction word. This addressing mode is used for the unconditional branch instruction.

Examples:

```
BRA      REL10             ; PC = PC + 2 + (REL10 << 1)
```

5.8.1.12 Index Register plus Immediate Offset (IDO5)

(RS, #offset5) provides an unsigned offset from the base register.

Examples:

```
LDB      R4,(R1,#offset)   ; loads a byte from R1+offset into R4
STW      R4,(R1,#offset)   ; stores R4 as a word to R1+offset
```

5.8.1.13 Index Register plus Register Offset (IDR)

For load and store instructions (RS, RI) provides a variable offset in a register.

Examples:

```
LDB     R4,(R1,R2)         ; loads a byte from R1+R2 into R4
STW     R4,(R1,R2)         ; stores R4 as a word to R1+R2
```

5.8.1.14 Index Register plus Register Offset with Post-increment (IDR+)

[RS, RI+] provides a variable offset in a register, which is incremented after accessing the memory. In case of a byte access the index register will be incremented by one. In case of a word access it will be incremented by two.

Examples:

```
LDB     R4,(R1,R2+)        ; loads a byte from R1+R2 into R4, R2+=1
STW     R4,(R1,R2+)        ; stores R4 as a word to R1+R2, R2+=2
```

5.8.1.15 Index Register plus Register Offset with Pre-decrement (–IDR)

[RS, -RI] provides a variable offset in a register, which is decremented before accessing the memory. In case of a byte access the index register will be decremented by one. In case of a word access it will be decremented by two.

Examples:

```
LDB     R4,(R1,-R2)        ; R2 -=1, loads a byte from R1+R2 into R4
STW     R4,(R1,-R2)        ; R2 -=2, stores R4 as a word to R1+R2
```

5.8.2 Instruction Summary and Usage

5.8.2.1 Load & Store Instructions

Any register can be loaded either with an immediate or from the address space using indexed addressing modes.

```
LDL     RD,#IMM8           ; loads an immediate 8 bit value to the lower byte of RD
LDW     RD,(RB,RI)         ; loads data using RB+RI as effective address

LDB     RD,(RB, RI+)       ; loads data using RB+RI as effective address
                           ; followed by an increment of RI depending on
                           ; the size of the operation
```

The same set of modes is available for the store instructions

```
STB     RS,(RB, RI)        ; stores data using RB+RI as effective address

STW     RS,(RB, RI+)       ; stores data using RB+RI as effective address
                           ; followed by an increment of RI depending on
                           ; the size of the operation.
```

5.8.2.2 Logic and Arithmetic Instructions

All logic and arithmetic instructions support the 8 bit immediate addressing mode (IMM8: RD = RD ∗ #IMM8) and the triadic addressing mode (TRI: RD = RS1 ∗ RS2).

All arithmetic is considered as signed, sign, overflow, zero and carry flag will be updated. The carry will not be affected for logical operations.

```
ADDL     R2,#1               ; increment R2
ANDH     R4,#$FE             ; R4.H = R4.H & $FE, clear lower bit of higher byte

ADD      R3,R4,R5            ; R3 = R4 + R5
SUB      R3,R4,R5            ; R3 = R4 - R5

AND      R3,R4,R5            ; R3 = R4 & R5 logical AND on the whole word
OR       R3,R4,R5            ; R3 = R4 | R5
```

5.8.2.3 Register – Register Transfers

This group comprises transfers from and to some special registers

```
TFR      R3,CCR              ; transfers the condition code register to the low byte of
                             ; register R3
```

`Branch Instructions`

The branch offset is +255 words or -256 words counted from the beginning of the next instruction. Since instructions have a fixed 16 bit width, the branch offsets are word aligned by shifting the offset value by 2.

```
BEQ      label               ; if Z flag = 1 branch to label
```

An unconditional branch allows a +511 words or -512 words branch distance.

```
BRA      label
```

5.8.2.4 Shift Instructions

Shift operations allow the use of a 4 bit wide immediate value to identify a shift width within a 16 bit word. For shift operations a value of 0 does not shift at all, while a value of 15 shifts the register RD by 15 bits. In a second form the shift value is contained in the bits 3:0 of the register RS.

Examples:

```
LSL      R4,#1    ; R4 = R4 << 1; shift register R4 by 1 bit to the left
LSR      R4,#3    ; R4 = R4 >> 3; shift register R4 by 3 bits to the right
ASR      R4,R2    ; R4 = R4 >> R2;arithmetic shift register R4 right by the amount
                  ;             of bits contained in R2[3:0].
```

5.8.2.5 Bit Field Operations

This addressing mode is used to identify the position and size of a bit field for insertion or extraction. The width and offset are coded in the lower byte of the source register 2, RS2. The content of the upper byte is ignored. An offset of 0 denotes the right most position and a width of 0 denotes 1 bit. These instructions are very useful to extract, insert, clear, set or toggle portions of a 16 bit word.

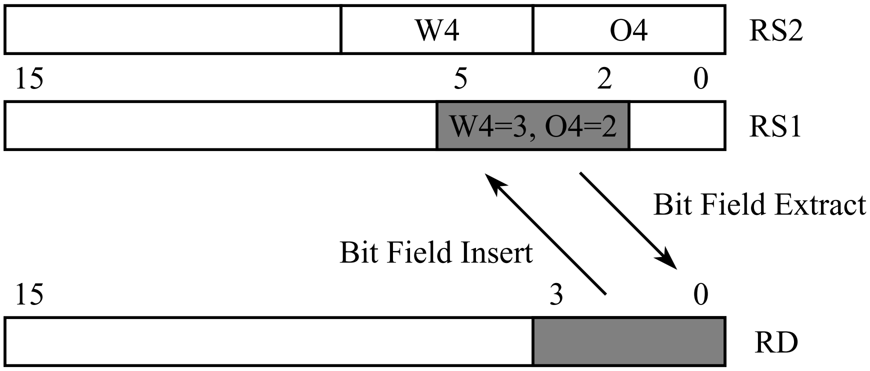


Figure 5-23. Bit Field Addressing

```
BFEXT    R3,R4,R5 ; R5: W4 bits offset O4, will be extracted from R4 into R3
```

5.8.2.6 Special Instructions for DMA Usage

The XGATE offers a number of additional instructions for flag manipulation, program flow control and debugging:

1. SIF: Set a channel interrupt flag
2. SSEM: Test and set a hardware semaphore
3. CSEM: Clear a hardware semaphore
4. BRK: Software breakpoint
5. NOP: No Operation
6. RTS: Terminate the current thread

5.8.3 Cycle Notation

Table 5-16 show the XGATE access detail notation. Each code letter equals one XGATE cycle. Each letter implies additional wait cycles if memories or peripherals are not accessible. Memories or peripherals are not accessible if they are blocked by the S12X_CPU. In addition to this Peripherals are only accessible every other XGATE cycle. Uppercase letters denote 16 bit operations. Lowercase letters denote 8 bit operations. The XGATE is able to perform two bus or wait cycles per S12X_CPU cycle.

Table 5-16. Access Detail Notation

`V` — Vector fetch: always an aligned word read, lasts for at least one RISC core cycle

`P` — Program word fetch: always an aligned word read, lasts for at least one RISC core cycle

`r` — 8 bit data read: lasts for at least one RISC core cycle

`R` — 16 bit data read: lasts for at least one RISC core cycle

`w` — 8 bit data write: lasts for at least one RISC core cycle

`W` — 16 bit data write: lasts for at least one RISC core cycle

`A` — Alignment cycle: no read or write, lasts for zero or one RISC core cycles

`f` — Free cycle: no read or write, lasts for one RISC core cycles

Special Cases

`PP/P` — Branch: `PP` if branch taken, `P` if not

5.8.4 Thread Execution

When the RISC core is triggered by an interrupt request (see Figure 5-1) it first executes a vector fetch sequence which performs three bus accesses:

1. A `V`-cycle to fetch the initial content of the program counter.
2. A `V`-cycle to fetch the initial content of the data segment pointer (R1).
3. A `P`-cycle to load the initial opcode.

Afterwards a sequence of instructions (thread) is executed which is terminated by an "RTS" instruction. If further interrupt requests are pending after a thread has been terminated, a new vector fetch will be performed. Otherwise the RISC core will idle until a new interrupt request is received. A thread can not be interrupted by an interrupt request.

5.8.5 Instruction Glossary

This section describes the XGATE instruction set in alphabetical order.

ADC Add with Carry ADC

Operation

RS1 + RS2 + C ⇒ RD

Adds the content of register RS1, the content of register RS2 and the value of the Carry bit using binary addition and stores the result in the destination register RD. The Zero Flag is also carried forward from the previous operation allowing 32 and more bit additions.

Example:

```
ADC     R6,R2,R2
ADC     R7,R3,R3 ; R7:R6 = R5:R4 + R3:R2
BCC              ; conditional branch on 32 bit addition
```

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000 and Z was set before this operation; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
RS1[15] & RS2[15] & $\overline{\text{RD[15]}}_{\text{new}}$ | $\overline{\text{RS1[15]}}$ & $\overline{\text{RS2[15]}}$ & $\text{RD[15]}_{\text{new}}$

C: Set if there is a carry from bit 15 of the result; cleared otherwise.
RS1[15] & RS2[15] | RS1[15] & $\overline{\text{RD[15]}}_{\text{new}}$ | RS2[15] & $\overline{\text{RD[15]}}_{\text{new}}$

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
ADC RD, RS1, RS2	TRI	0	0	0	1	1	RD	RS1	RS2	1	1	P

ADD Add without Carry ADD

Operation

RS1 + RS2 ⇒ RD
RD + IMM16 ⇒ RD (translates to ADDL RD, #IMM16[7:0]; ADDH RD, #[15:8])

Performs a 16 bit addition and stores the result in the destination register RD.

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RS1[15] \ \& \ RS2[15] \ \& \ \overline{RD[15]}_{new} \ | \ \overline{RS1[15]} \ \& \ \overline{RS2[15]} \ \& \ RD[15]_{new}$
Refer to ADDH instruction for #IMM16 operations.

C: Set if there is a carry from bit 15 of the result; cleared otherwise.
$RS1[15] \ \& \ RS2[15] \ | \ RS1[15] \ \& \ \overline{RD[15]}_{new} \ | \ RS2[15] \ \& \ \overline{RD[15]}_{new}$
Refer to ADDH instruction for #IMM16 operations.

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
ADD RD, RS1, RS2	TRI	0	0	0	1	1	RD	RS1	RS2	1	0	P
ADD RD, #IMM16	IMM8	1	1	1	0	0	RD	IMM16[7:0]				P
	IMM8	1	1	1	0	1	RD	IMM16[15:8]				P

ADDH Add Immediate 8 bit Constant (High Byte) ADDH

Operation

RD + IMM8:\$00 ⇒ RD

Adds the content of high byte of register RD and a signed immediate 8 bit constant using binary addition and stores the result in the high byte of the destination register RD. This instruction can be used after an ADDL for a 16 bit immediate addition.

Example:

```
ADDL     R2,#LOWBYTE
ADDH     R2,#HIGHBYTE      ; R2 = R2 + 16 bit immediate
```

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is \$0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RD[15]_{old}\ \&\ IMM8[7]\ \&\ \overline{RD[15]}_{new}\ |\ \overline{RD[15]}_{old}\ \&\ \overline{IMM8[7]}\ \&\ RD[15]_{new}$

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$RD[15]_{old}\ \&\ IMM8[7]\ |\ RD[15]_{old}\ \&\ \overline{RD[15]}_{new}\ |\ IMM8[7]\ \&\ \overline{RD[15]}_{new}$

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
ADDH RD, #IMM8	IMM8	1	1	1	0	1	RD	IMM8	P

ADDL Add Immediate 8 bit Constant (Low Byte) ADDL

Operation

$RD + \$00{:}IMM8 \Rightarrow RD$

Adds the content of register RD and an unsigned immediate 8 bit constant using binary addition and stores the result in the destination register RD. This instruction must be used first for a 16 bit immediate addition in conjunction with the ADDH instruction.

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is \$0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the 8 bit operation; cleared otherwise.
$\overline{RD[15]}_{old}$ & $RD[15]_{new}$

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$RD[15]_{old}$ & $\overline{RD[15]}_{new}$

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
ADDL RD, #IMM8	IMM8	1	1	1	0	0	RD	IMM8	P

AND Logical AND AND

Operation

RS1 & RS2 ⇒ RD
RD & IMM16 ⇒ RD (translates to ANDL RD, #IMM16[7:0]; ANDH RD, #IMM16[15:8])

Performs a bit wise logical AND of two 16 bit values and stores the result in the destination register RD.

Remark: There is no complement to the BITH and BITL functions. This can be imitated by using R0 as a destination register. AND R0, RS1, RS2 performs a bit wise test without storing a result.

CCR Effects

N	Z	V	C
Δ	Δ	0	—

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
Refer to ANDH instruction for #IMM16 operations.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
AND RD, RS1, RS2	TRI	0	0	0	1	0	RD	RS1	RS2	0	0	P
AND RD, #IMM16	IMM8	1	0	0	0	0	RD	IMM16[7:0]				P
	IMM8	1	0	0	0	1	RD	IMM16[15:8]				P

ANDH Logical AND Immediate 8 bit Constant (High Byte) ANDH

Operation

RD.H & IMM8 ⇒ RD.H

Performs a bit wise logical AND between the high byte of register RD and an immediate 8 bit constant and stores the result in the destination register RD.H. The low byte of RD is not affected.

CCR Effects

N	Z	V	C
Δ	Δ	0	—

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the 8 bit result is $00; cleared otherwise.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
ANDH RD, #IMM8	IMM8	1	0	0	0	1	RD	IMM8	P

ANDL Logical AND Immediate 8 bit Constant (Low Byte) ANDL

Operation

RD.L & IMM8 ⇒ RD.L

Performs a bit wise logical AND between the low byte of register RD and an immediate 8 bit constant and stores the result in the destination register RD.L. The high byte of RD is not affected.

CCR Effects

N	Z	V	C
Δ	Δ	0	—

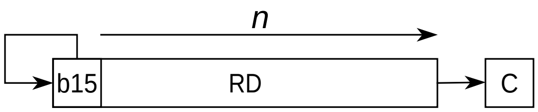
N: Set if bit 7 of the result is set; cleared otherwise.
Z: Set if the 8 bit result is $00; cleared otherwise.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
ANDL RD, #IMM8	IMM8	1	0	0	0	0	RD	IMM8	P

ASR Arithmetic Shift Right ASR

Operation



n = RS or IMM4

Shifts the bits in register RD n positions to the right. The higher n bits of the register RD become filled with the sign bit (RD[15]). The carry flag will be updated to the bit contained in RD[n-1] before the shift for $n > 0$.

n can range from 0 to 16.

In immediate address mode, n is determined by the operand IMM4. n is considered to be 16 in IMM4 is equal to 0.

In dyadic address mode, n is determined by the content of RS. n is considered to be 16 if the content of RS is greater than 15.

CCR Effects

N	Z	V	C
Δ	Δ	0	Δ

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RD[15]_{old}$ ^ $RD[15]_{new}$
C: Set if $n > 0$ and RD[n-1] = 1; if n = 0 unaffected.

Code and CPU Cycles

<table>
<tr><th>Source Form</th><th>Address Mode</th><th colspan="12">Machine Code</th><th>Cycles</th></tr>
<tr><td>ASR RD, #IMM4</td><td>IMM4</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>RD</td><td colspan="2">IMM4</td><td>1</td><td>0</td><td>0</td><td>1</td><td>P</td></tr>
<tr><td>ASR RD, RS</td><td>DYA</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>RD</td><td>RS</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>P</td></tr>
</table>

BCC Branch if Carry Cleared (Same as BHS) BCC

Operation

If C = 0, then PC + \$0002 + (REL9 << 1) ⇒ PC

Tests the Carry flag and branches if C = 0.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code								Cycles
BCC REL9	REL9	0	0	1	0	0	0	0	REL9	PP/P

BCS Branch if Carry Set (Same as BLO) BCS

Operation

If C = 1, then PC + \$0002 + (REL9 << 1) ⇒ PC

Tests the Carry flag and branches if C = 1.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code								Cycles
BCS REL9	REL9	0	0	1	0	0	0	1	REL9	PP/P

BEQ Branch if Equal BEQ

Operation

If Z = 1, then PC + \$0002 + (REL9 << 1) ⇒ PC

Tests the Zero flag and branches if Z = 1.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code								Cycles
BEQ REL9	REL9	0	0	1	0	0	1	1	REL9	PP/P

BFEXT Bit Field Extract BFEXT

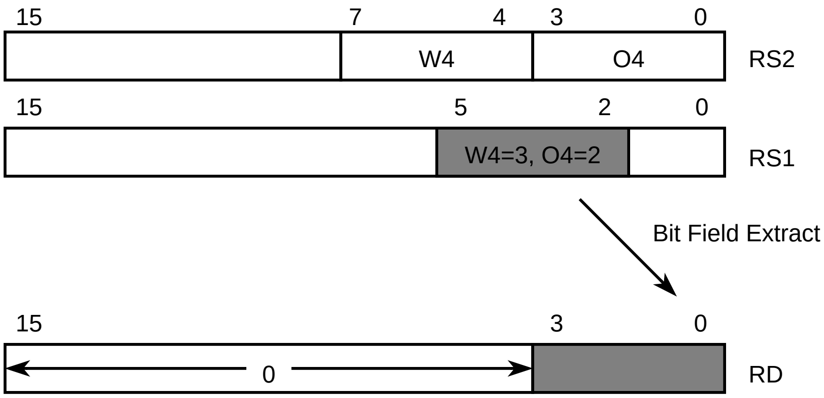
Operation

$RS1[(o+w):o] \Rightarrow RD[w:0];\ 0 \Rightarrow RD[15:(w+1)]$

$w = (RS2[7:4])$

$o = (RS2[3:0])$

Extracts $w+1$ bits from register RS1 starting at position o and writes them right aligned into register RD. The remaining bits in RD will be cleared. If (o+w) > 15 only bits [15:o] get extracted.



CCR Effects

N	Z	V	C
0	Δ	0	Δ

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
BFEXT RD, RS1, RS2	TRI	0	1	1	0	0	RD	RS1	RS2	1	1	P

BFFO Bit Field Find First One BFFO

Operation

FirstOne (RS) ⇒ RD;

Searches the first “1” in register RS (from MSB to LSB) and writes the bit position into the destination register RD. The upper bits of RD are cleared. In case the content of RS is equal to \$0000, RD will be cleared and the carry flag will be set. This is used to distinguish a “1” in position 0 versus no “1” in the whole RS register at all.

CCR Effects

N	Z	V	C
0	Δ	0	Δ

N: 0; cleared.
Z: Set if the result is \$0000; cleared otherwise.
V: 0; cleared.
C: Set if RS = \$0000[1]; cleared otherwise.

[1] Before executing the instruction

Code and CPU Cycles

Source Form	Address Mode	Machine Code												Cycles
BFFO RD, RS	DYA	0	0	0	0	1	RD	RS	1	0	0	0	0	P

BFINS Bit Field Insert BFINS

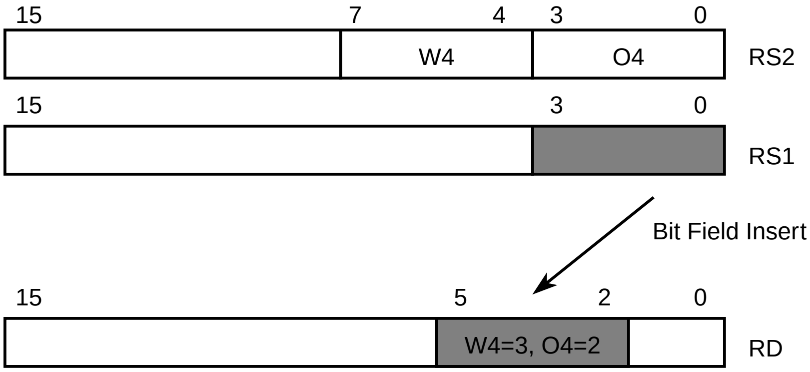
Operation

RS1[w:0] ⇒ RD[(w+o):o];

w = (RS2[7:4])

o = (RS2[3:0])

Extracts $w+1$ bits from register RS1 starting at position 0 and writes them into register RD starting at position o. The remaining bits in RD are not affected. If (o+w) > 15 the upper bits are ignored. Using R0 as a RS1, this command can be used to clear bits.



CCR Effects

N	Z	V	C
Δ	Δ	0	—

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
BFINS RD, RS1, RS2	TRI	0	1	1	0	1	RD	RS1	RS2	1	1	P

BFINSI Bit Field Insert and Invert BFINSI

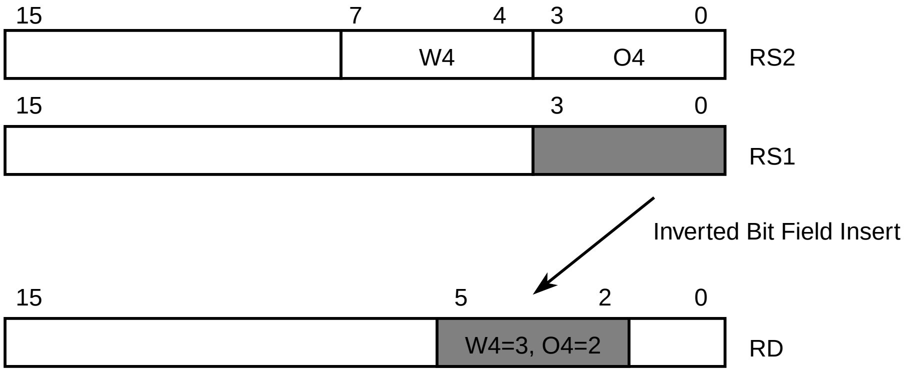
Operation

$!RS1[w{:}0] \Rightarrow RD[w{+}o{:}o];$

$w = (RS2[7{:}4])$

$o = (RS2[3{:}0])$

Extracts $w+1$ bits from register RS1 starting at position 0, inverts them and writes into register RD starting at position o. The remaining bits in RD are not affected. If (o+w) > 15 the upper bits are ignored. Using R0 as a RS1, this command can be used to set bits.



CCR Effects

N	Z	V	C
Δ	Δ	0	—

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: 0; cleared.

C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
BFINSI RD, RS1, RS2	TRI	0	1	1	1	0	RD	RS1	RS2	1	1	P

BFINSX Bit Field Insert and XNOR BFINSX

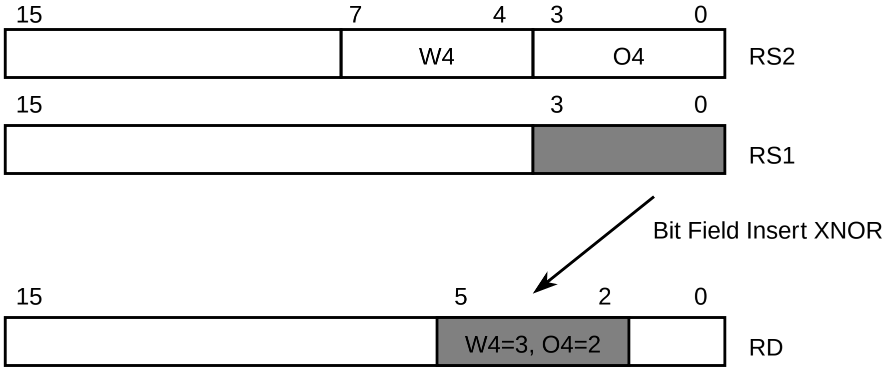
Operation

$!(RS1[w:0] \wedge RD[w+o:o]) \Rightarrow RD[w+o:o];$

$w = (RS2[7:4])$

$o = (RS2[3:0])$

Extracts *w+1* bits from register RS1 starting at position 0, performs an XNOR with RD[w+o:o] and writes the bits back io RD. The remaining bits in RD are not affected. If (o+w) > 15 the upper bits are ignored. Using R0 as a RS1, this command can be used to toggle bits.



CCR Effects

N	Z	V	C
Δ	Δ	0	—

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
BFINSX RD, RS1, RS2	TRI	0	1	1	1	1	RD	RS1	RS2	1	1	P

BGE Branch if Greater than or Equal to Zero BGE

Operation

If N ^ V = 0, then PC + \$0002 + (REL9 << 1) ⇒ PC

Branch instruction to compare signed numbers.

Branch if RS1 ≥ RS2:

```
        SUB     R0,RS1,RS2
        BGE     REL9
```

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code								Cycles
BGE REL9	REL9	0	0	1	1	0	1	0	REL9	PP/P

BGT Branch if Greater than Zero BGT

Operation

If Z | (N ^ V) = 0, then PC + $0002 + (REL9 << 1) ⇒ PC

Branch instruction to compare signed numbers.

Branch if RS1 > RS2:

```
SUB     R0,RS1,RS2
BGE     REL9
```

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code								Cycles
BGT REL9	REL9	0	0	1	1	1	0	0	REL9	PP/P

BHI Branch if Higher BHI

Operation

If C | Z = 0, then PC + \$0002 + (REL9 << 1) ⇒ PC

Branch instruction to compare unsigned numbers.

Branch if RS1 > RS2:

```
SUB     R0,RS1,RS2
BHI     REL9
```

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code								Cycles
BHI REL9	REL9	0	0	1	1	0	0	0	REL9	PP/P

BHS Branch if Higher or Same (Same as BCC) BHS

Operation

If C = 0, then PC + \$0002 + (REL9 << 1) ⇒ PC

Branch instruction to compare unsigned numbers.

Branch if RS1 ≥ RS2:

```
          SUB       R0,RS1,RS2
          BHS       REL9
```

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code								Cycles
BHS REL9	REL9	0	0	1	0	0	0	0	REL9	PP/P

BITH Bit Test Immediate 8 bit Constant (High Byte) BITH

Operation

RD.H & IMM8 ⇒ NONE

Performs a bit wise logical AND between the high byte of register RD and an immediate 8 bit constant. Only the condition code flags get updated, but no result is written back

CCR Effects

N	Z	V	C
Δ	Δ	0	—

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the 8 bit result is $00; cleared otherwise.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
BITH RD, #IMM8	IMM8	1	0	0	1	1	RD	IMM8	P

BITL Bit Test Immediate 8 bit Constant (Low Byte) BITL

Operation

RD.L & IMM8 ⇒ NONE

Performs a bit wise logical AND between the low byte of register RD and an immediate 8 bit constant. Only the condition code flags get updated, but no result is written back.

CCR Effects

N	Z	V	C
Δ	Δ	0	—

N: Set if bit 7 of the result is set; cleared otherwise.
Z: Set if the 8 bit result is $00; cleared otherwise.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
BITL RD, #IMM8	IMM8	1	0	0	1	0	RD	IMM8	P

BLE Branch if Less or Equal to Zero BLE

Operation

If Z | (N ^ V) = 1, then PC + \$0002 + (REL9 << 1) ⇒ PC

Branch instruction to compare signed numbers.

Branch if RS1 ≤ RS2:

```
         SUB      R0,RS1,RS2
         BLE      REL9
```

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code								Cycles
BLE REL9	REL9	0	0	1	1	1	0	1	REL9	PP/P

BLO Branch if Carry Set (Same as BCS) BLO

Operation

If C = 1, then PC + \$0002 + (REL9 << 1) ⇒ PC

Branch instruction to compare unsigned numbers.

Branch if RS1 < RS2:

```
SUB     R0,RS1,RS2
BLO     REL9
```

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code								Cycles
BLO REL9	REL9	0	0	1	0	0	0	1	REL9	PP/P

BLS Branch if Lower or Same BLS

Operation

If C | Z = 1, then PC + $0002 + (REL9 << 1) ⇒ PC

Branch instruction to compare unsigned numbers.

Branch if RS1 ≤ RS2:

```
SUB     R0,RS1,RS2
BLS     REL9
```

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code								Cycles
BLS REL9	REL9	0	0	1	1	0	0	1	REL9	PP/P

BLT Branch if Lower than Zero BLT

Operation

If N ^ V = 1, then PC + \$0002 + (REL9 << 1) ⇒ PC

Branch instruction to compare signed numbers.

Branch if RS1 < RS2:

```
SUB     R0,RS1,RS2
BLT     REL9
```

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code								Cycles
BLT REL9	REL9	0	0	1	1	0	1	1	REL9	PP/P

BMI Branch if Minus BMI

Operation

If N = 1, then PC + \$0002 + (REL9 << 1) ⇒ PC

Tests the Sign flag and branches if N = 1.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code								Cycles
BMI REL9	REL9	0	0	1	0	1	0	1	REL9	PP/P

BNE Branch if Not Equal BNE

Operation

If Z = 0, then PC + \$0002 + (REL9 << 1) ⇒ PC

Tests the Zero flag and branches if Z = 0.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code								Cycles
BNE REL9	REL9	0	0	1	0	0	1	0	REL9	PP/P

BPL Branch if Plus BPL

Operation

If N = 0, then PC + \$0002 + (REL9 << 1) ⇒ PC

Tests the Sign flag and branches if N = 0.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code								Cycles
BPL REL9	REL9	0	0	1	0	1	0	0	REL9	PP/P

BRA Branch Always BRA

Operation

PC + \$0002 + (REL10 << 1) ⇒ PC

Branches always

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
BRA REL10	REL10	0	0	1	1	1	1	REL10	PP

BRK **Break** BRK

Operation

Put XGATE into Debug Mode (see Section 5.6.2, “Entering Debug Mode”)and signals a Software breakpoint to the S12X_DBG module (see section 4.9 of the **S12X_DBG Section**).

NOTE

> It is not possible to single step over a BRK instruction. This instruction does not advance the program counter.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code																Cycles
BRK	INH	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PAff

BVC Branch if Overflow Cleared BVC

Operation

If V = 0, then PC + $0002 + (REL9 << 1) ⇒ PC

Tests the Overflow flag and branches if V = 0.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code								Cycles
BVC REL9	REL9	0	0	1	0	1	1	0	REL9	PP/P

BVS Branch if Overflow Set BVS

Operation

If V = 1, then PC + $0002 + (REL9 << 1) ⇒ PC

Tests the Overflow flag and branches if V = 1.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code								Cycles
BVS REL9	REL9	0	0	1	0	1	1	1	REL9	PP/P

CMP Compare CMP

Operation

RS1 – RS2 ⇒ NONE (translates to SUB R0, RS1, RS2)
RD – IMM16 ⇒ NONE (translates to CMPL RD, #IMM16[7:0]; CPCH RD, #IMM16[15:8])

Subtracts two 16 bit values and discards the result.

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RS1[15] \ \& \ \overline{RS2[15]} \ \& \ \overline{result[15]} \ | \ \overline{RS1[15]} \ \& \ RS2[15] \ \& \ result[15]$
$RD[15] \ \& \ \overline{IMM16[15]} \ \& \ \overline{result[15]} \ | \ \overline{RD[15]} \ \& \ IMM16[15] \ \& \ result[15]$

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$\overline{RS1[15]} \ \& \ RS2[15] \ | \ \overline{RS1[15]} \ \& \ result[15] \ | \ RS2[15] \ \& \ result[15]$
$\overline{RD[15]} \ \& \ IMM16[15] \ | \ \overline{RD[15]} \ \& \ result[15] \ | \ IMM16[15] \ \& \ result[15]$

Code and CPU Cycles

<table>
<tr><th>Source Form</th><th>Address Mode</th><th colspan="16">Machine Code</th><th>Cycles</th></tr>
<tr><td>CMP RS1, RS2</td><td>TRI</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td colspan="3">0 0 0</td><td colspan="3">RS1</td><td colspan="3">RS2</td><td>0</td><td>0</td><td>P</td></tr>
<tr><td rowspan="2">CMP RS, #IMM16</td><td>IMM8</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td colspan="3">RS</td><td colspan="8">IMM16[7:0]</td><td>P</td></tr>
<tr><td>IMM8</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td colspan="3">RS</td><td colspan="8">IMM16[15:8]</td><td>P</td></tr>
</table>

CMPL Compare Immediate 8 bit Constant (Low Byte) CMPL

Operation

RS.L – IMM8 ⇒ NONE, only condition code flags get updated

Subtracts the 8 bit constant IMM8 contained in the instruction code from the low byte of the source register RS.L using binary subtraction and updates the condition code register accordingly.

Remark: There is no equivalent operation using triadic addressing. Comparing the values of two registers can be performed by using the subtract instruction with R0 as destination register.

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

N: Set if bit 7 of the result is set; cleared otherwise.

Z: Set if the 8 bit result is $00; cleared otherwise.

V: Set if a two´s complement overflow resulted from the 8 bit operation; cleared otherwise.
$RS[7] \ \& \ \overline{IMM8[7]} \ \& \ \overline{result[7]} \ | \ \overline{RS[7]} \ \& \ IMM8[7] \ \& \ result[7]$

C: Set if there is a carry from the Bit 7 to Bit 8 of the result; cleared otherwise.
$\overline{RS[7]} \ \& \ IMM8[7] \ | \ \overline{RS[7]} \ \& \ result[7] \ | \ IMM8[7] \ \& \ result[7]$

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
CMPL RS, #IMM8	IMM8	1	1	0	1	0	RS	IMM8	P

COM One’s Complement COM

Operation

~RS $\Rightarrow$ RD (translates to XNOR RD, R0, RS)
~RD $\Rightarrow$ RD (translates to XNOR RD, R0, RD)

Performs a one’s complement on a general purpose register.

CCR Effects

N	Z	V	C
Δ	Δ	0	—

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
COM RD, RS	TRI	0	0	0	1	0	RD	0 0 0	RS	1	1	P
COM RD	TRI	0	0	0	1	0	RD	0 0 0	RD	1	1	P

CPC Compare with Carry CPC

Operation

RS2 – RS1 - C ⇒ NONE (translates to SBC R0, RS1, RS2)

Subtracts the carry bit and the content of register RS2 from the content of register RS1 using binary subtraction and discards the result.

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RS1[15] \,\&\, \overline{RS2[15]} \,\&\, \overline{result[15]} \mid \overline{RS1[15]} \,\&\, RS2[15] \,\&\, result[15]$

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$\overline{RS1[15]} \,\&\, RS2[15] \mid \overline{RS1[15]} \,\&\, result[15] \mid RS2[15] \,\&\, result[15]$

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
CPC RS1, RS2	TRI	0	0	0	1	1	0 0 0	RS1	RS2	0	1	P

CPCH Compare Immediate 8 bit Constant with Carry (High Byte) CPCH

Operation

RS.H - IMM8 - C ⇒ NONE, only condition code flags get updated

Subtracts the carry bit and the 8 bit constant IMM8 contained in the instruction code from the high byte of the source register RD using binary subtraction and updates the condition code register accordingly. The carry bit and Zero bits are taken into account to allow a 16 bit compare in the form of

```
CMPL     R2,#LOWBYTE
CPCH     R2,#HIGHBYTE
BCC                         ; branch condition
```

Remark: There is no equivalent operation using triadic addressing. Comparing the values of two registers can be performed by using the subtract instruction with R0 as destination register.

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $00 and Z was set before this operation; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$\text{RS[15]} \,\&\, \overline{\text{IMM8[7]}} \,\&\, \overline{\text{result[15]}} \mid \overline{\text{RS[15]}} \,\&\, \text{IMM8[7]} \,\&\, \text{result[15]}$

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$\overline{\text{RS[15]}} \,\&\, \text{IMM8[7]} \mid \overline{\text{RS[15]}} \,\&\, \text{result[15]} \mid \text{IMM8[7]} \,\&\, \text{result[15]}$

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
CPCH RD, #IMM8	IMM8	1	1	0	1	1	RS	IMM8	P

CSEM Clear Semaphore CSEM

Operation

Unlocks a semaphore that was locked by the RISC core.

In monadic address mode, bits RS[2:0] select the semaphore to be cleared.

CCR Effects

N	Z	V	C
—	—	—	—

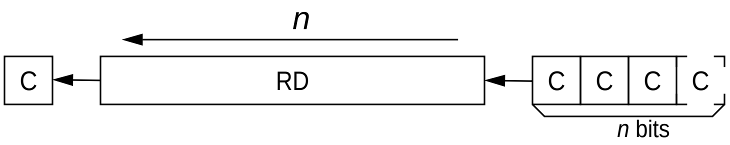
N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code													Cycles
CSEM #IMM3	IMM3	0	0	0	0	0	IMM3	1	1	1	1	0	0	0	0	PA
CSEM RS	MON	0	0	0	0	0	RS	1	1	1	1	0	0	0	1	PA

CSL Logical Shift Left with Carry CSL

Operation



n = RS or IMM4

Shifts the bits in register RD n positions to the left. The lower n bits of the register RD become filled with the carry flag. The carry flag will be updated to the bit contained in RD[16-n] before the shift for $n > 0$. n can range from 0 to 16.

In immediate address mode, n is determined by the operand IMM4. n is considered to be 16 in IMM4 is equal to 0.

In dyadic address mode, n is determined by the content of RS. n is considered to be 16 if the content of RS is greater than 15.

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

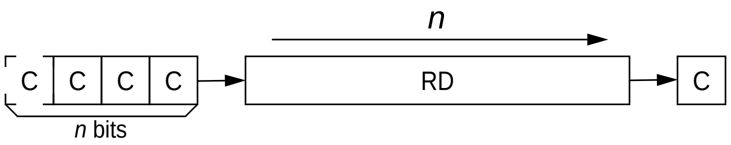
N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RD[15]_{old}$ ^ $RD[15]_{new}$
C: Set if $n > 0$ and RD[16-n] = 1; if n = 0 unaffected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code												Cycles
CSL RD, #IMM4	IMM4	0	0	0	0	1	RD	IMM4		1	0	1	0	P
CSL RD, RS	DYA	0	0	0	0	1	RD	RS	1	0	0	1	0	P

CSR Logical Shift Right with Carry CSR

Operation



n = RS or IMM4

Shifts the bits in register RD n positions to the right. The higher n bits of the register RD become filled with the carry flag. The carry flag will be updated to the bit contained in RD[n-1] before the shift for $n > 0$. n can range from 0 to 16.

In immediate address mode, n is determined by the operand IMM4. n is considered to be 16 in IMM4 is equal to 0.

In dyadic address mode, n is determined by the content of RS. n is considered to be 16 if the content of RS is greater than 15.

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RD[15]_{old}$ ^ $RD[15]_{new}$

C: Set if $n > 0$ and RD[n-1] = 1; if n = 0 unaffected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code												Cycles
CSR RD, #IMM4	IMM4	0	0	0	0	1	RD	IMM4		1	0	1	1	P
CSR RD, RS	DYA	0	0	0	0	1	RD	RS	1	0	0	1	1	P

JAL Jump and Link JAL

Operation

PC + \$0002 ⇒ RD; RD ⇒ PC

Jumps to the address stored in RD and saves the return address in RD.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code														Cycles
JAL RD	MON	0	0	0	0	0	RD	1	1	1	1	0	1	1	0	PP

LDB Load Byte from Memory (Low Byte) LDB

Operation

M[RB, #OFFS5 ⇒ RD.L; $00 ⇒ RD.H
M[RB, RI] ⇒ RD.L; $00 ⇒ RD.H
M[RB, RI] ⇒ RD.L; $00 ⇒ RD.H; RI+1 ⇒ RI;[1]
RI-1 ⇒ RI; M[RS, RI] ⇒ RD.L; $00 ⇒ RD.H

Loads a byte from memory into the low byte of register RD. The high byte is cleared.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
LDB RD, (RB, #OFFS5)	IDO5	0	1	0	0	0	RD	RB	OFFS5			Pr
LDB RD, (RS, RI)	IDR	0	1	1	0	0	RD	RB	RI	0	0	Pr
LDB RD, (RS, RI+)	IDR+	0	1	1	0	0	RD	RB	RI	0	1	Pr
LDB RD, (RS, -RI)	-IDR	0	1	1	0	0	RD	RB	RI	1	0	Pr

---

[1] If the same general purpose register is used as index (RI) and destination register (RD), the content of the register will not be incremented after the data move: M[RB, RI] ⇒ RD.L; $00 ⇒ RD.H

LDH Load Immediate 8 bit Constant (High Byte) LDH

Operation

IMM8 ⇒ RD.H;

Loads an eight bit immediate constant into the high byte of register RD. The low byte is not affected.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
LDH RD, #IMM8	IMM8	1	1	1	1	1	RD	IMM8	P

LDL Load Immediate 8 bit Constant (Low Byte) LDL

Operation

IMM8 ⇒ RD.L; \$00 ⇒ RD.H

Loads an eight bit immediate constant into the low byte of register RD. The high byte is cleared.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
LDL RD, #IMM8	IMM8	1	1	1	1	0	RD	IMM8	P

LDW Load Word from Memory LDW

Operation

M[RB, #OFFS5] ⇒ RD
M[RB, RI] ⇒ RD
M[RB, RI] ⇒ RD; RI+2 ⇒ RI[1]
RI-2 ⇒ RI; M[RS, RI] ⇒ RD
IMM16 ⇒ RD (translates to LDL RD, #IMM16[7:0]; LDH RD, #IMM16[15:8])

Loads a 16 bit value into the register RD.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

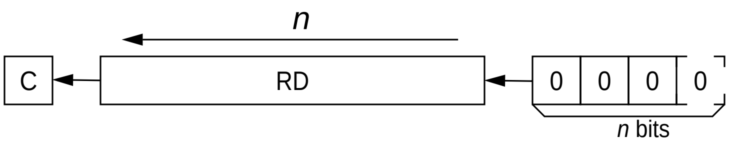
Source Form	Address Mode	Machine Code										Cycles
LDW RD, (RB, #OFFS5)	IDO5	0	1	0	0	1	RD	RB	OFFS5			PR
LDW RD, (RB, RI)	IDR	0	1	1	0	1	RD	RB	RI	0	0	PR
LDW RD, (RB, RI+)	IDR+	0	1	1	0	1	RD	RB	RI	0	1	PR
LDW RD, (RB, -RI)	-IDR	0	1	1	0	1	RD	RB	RI	1	0	PR
LDW RD, #IMM16	IMM8	1	1	1	1	0	RD	IMM16[7:0]				P
	IMM8	1	1	1	1	1	RD	IMM16[15:8]				P

---

[1] If the same general purpose register is used as index (RI) and destination register (RD), the content of the register will not be incremented after the data move: M[RB, RI] ⇒ RD

LSL Logical Shift Left LSL

Operation



n = RS or IMM4

Shifts the bits in register RD *n* positions to the left. The lower *n* bits of the register RD become filled with zeros. The carry flag will be updated to the bit contained in RD[16-n] before the shift for *n* > 0.

n can range from 0 to 16.

In immediate address mode, *n* is determined by the operand IMM4. *n* is considered to be 16 in IMM4 is equal to 0.

In dyadic address mode, *n* is determined by the content of RS. *n* is considered to be 16 if the content of RS is greater than 15.

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

N:	Set if bit 15 of the result is set; cleared otherwise.
Z:	Set if the result is $0000; cleared otherwise.
V:	Set if a two´s complement overflow resulted from the operation; cleared otherwise. $RD[15]_{old}$ ^ $RD[15]_{new}$
C:	Set if *n* > 0 and RD[16-n] = 1; if n = 0 unaffected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code												Cycles
LSL RD, #IMM4	IMM4	0	0	0	0	1	RD	IMM4		1	1	0	0	P
LSL RD, RS	DYA	0	0	0	0	1	RD	RS	1	0	1	0	0	P

LSR Logical Shift Right LSR

Operation

n = RS or IMM4

Shifts the bits in register RD n positions to the right. The higher n bits of the register RD become filled with zeros. The carry flag will be updated to the bit contained in RD[n-1] before the shift for $n > 0$.

n can range from 0 to 16.

In immediate address mode, n is determined by the operand IMM4. n is considered to be 16 in IMM4 is equal to 0.

In dyadic address mode, n is determined by the content of RS. n is considered to be 16 if the content of RS is greater than 15.

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is \$0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RD[15]_{old} \wedge RD[15]_{new}$

C: Set if $n > 0$ and RD[n-1] = 1; if n = 0 unaffected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code												Cycles
LSR RD, #IMM4	IMM4	0	0	0	0	1	RD	IMM4		1	1	0	1	P
LSR RD, RS	DYA	0	0	0	0	1	RD	RS	1	0	1	0	1	P

MOV Move Register Content MOV

Operation

RS ⇒ RD (translates to OR RD, R0, RS)

Copies the content of RS to RD.

CCR Effects

N	Z	V	C
Δ	Δ	0	—

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
MOV RD, RS	TRI	0	0	0	1	0	RD	0 0 0	RS	1	0	P

NEG Two’s Complement NEG

Operation

–RS $\Rightarrow$ RD (translates to SUB RD, R0, RS)
–RD $\Rightarrow$ RD (translates to SUB RD, R0, RD)

Performs a two’s complement on a general purpose register.

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RS[15]$ & $RD[15]_{new}$
C: Set if there is a carry from the bit 15 of the result; cleared otherwise
$RS[15] \mid RD[15]_{new}$

Code and CPU Cycles

Source Form	Address Mode	Machine Code														Cycles
NEG RD, RS	TRI	0	0	0	1	1	RD	0	0	0	RS	0	0	P
NEG RD	TRI	0	0	0	1	1	RD	0	0	0	RD	0	0	P

NOP No Operation NOP

Operation

No Operation for one cycle.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code																Cycles
NOP	INH	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	P

OR Logical OR OR

Operation

RS1 | RS2 ⇒ RD
RD | IMM16⇒ RD (translates to ORL RD, #IMM16[7:0]; ORH RD, #IMM16[15:8]

Performs a bit wise logical OR between two 16 bit values and stores the result in the destination register RD.

CCR Effects

N	Z	V	C
Δ	Δ	0	—

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
Refer to ORH instruction for #IMM16 operations.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
OR RD, RS1, RS2	TRI	0	0	0	1	0	RD	RS1	RS2	1	0	P
OR RD, #IMM16	IMM8	1	0	1	0	0	RD	IMM16[7:0]				P
	IMM8	1	0	1	0	1	RD	IMM16[15:8]				P

ORH Logical OR Immediate 8 bit Constant (High Byte) ORH

Operation

RD.H | IMM8 ⇒ RD.H

Performs a bit wise logical OR between the high byte of register RD and an immediate 8 bit constant and stores the result in the destination register RD.H. The low byte of RD is not affected.

CCR Effects

N	Z	V	C
Δ	Δ	0	—

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the 8 bit result is $00; cleared otherwise.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
ORH RD, #IMM8	IMM8	1	0	1	0	1	RD	IMM8	P

ORL Logical OR Immediate 8 bit Constant (Low Byte) ORL

Operation

RD.L | IMM8 ⇒ RD.L

Performs a bit wise logical OR between the low byte of register RD and an immediate 8 bit constant and stores the result in the destination register RD.L. The high byte of RD is not affected.

CCR Effects

N	Z	V	C
Δ	Δ	0	—

N: Set if bit 7 of the result is set; cleared otherwise.
Z: Set if the 8 bit result is $00; cleared otherwise.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
ORL RD, #IMM8	IMM8	1	0	1	0	0	RD	IMM8	P

PAR Calculate Parity PAR

Operation

Calculates the number of ones in the register RD. The Carry flag will be set if the number is odd, otherwise it will be cleared.

CCR Effects

N	Z	V	C
0	Δ	0	Δ

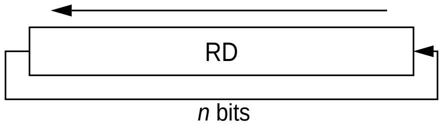
N: 0; cleared.
Z: Set if RD is $0000; cleared otherwise.
V: 0; cleared.
C: Set if there the number of ones in the register RD is odd; cleared otherwise.

Code and CPU Cycles

Source Form	Address Mode	Machine Code														Cycles
PAR, RD	MON	0	0	0	0	0	RD	1	1	1	1	0	1	0	1	P

ROL Rotate Left ROL

Operation



n = RS or IMM4

Rotates the bits in register RD *n* positions to the left. The lower *n* bits of the register RD are filled with the upper *n* bits. Two source forms are available. In the first form, the parameter *n* is contained in the instruction code as an immediate operand. In the second form, the parameter is contained in the lower bits of the source register RS[3:0]. All other bits in RS are ignored. If *n* is zero, no shift will take place and the register RD will be unaffected; however, the condition code flags will be updated.

CCR Effects

N	Z	V	C
Δ	Δ	0	—

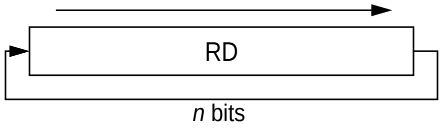
N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

<table>
<tr><th>Source Form</th><th>Address Mode</th><th colspan="12">Machine Code</th><th>Cycles</th></tr>
<tr><td>ROL RD, #IMM4</td><td>IMM4</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>RD</td><td colspan="2">IMM4</td><td>1</td><td>1</td><td>1</td><td>0</td><td>P</td></tr>
<tr><td>ROL RD, RS</td><td>DYA</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>RD</td><td>RS</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>P</td></tr>
</table>

ROR Rotate Right ROR

Operation



n = RS or IMM4

Rotates the bits in register RD *n* positions to the right. The upper *n* bits of the register RD are filled with the lower *n* bits. Two source forms are available. In the first form, the parameter *n* is contained in the instruction code as an immediate operand. In the second form, the parameter is contained in the lower bits of the source register RS[3:0]. All other bits in RS are ignored. If *n* is zero no shift will take place and the register RD will be unaffected; however, the condition code flags will be updated.

CCR Effects

N	Z	V	C
Δ	Δ	0	—

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code												Cycles
ROR RD, #IMM4	IMM4	0	0	0	0	1	RD	IMM4		1	1	1	1	P
ROR RD, RS	DYA	0	0	0	0	1	RD	RS	1	0	1	1	1	P

RTS Return to Scheduler RTS

Operation

Terminates the current thread of program execution and remains idle until a new thread is started by the hardware scheduler.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code																Cycles
RTS	INH	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	PA

SBC Subtract with Carry SBC

Operation

RS1 - RS2 - C $\Rightarrow$ RD

Subtracts the content of register RS2 and the value of the Carry bit from the content of register RS1 using binary subtraction and stores the result in the destination register RD. Also the zero flag is carried forward from the previous operation allowing 32 and more bit subtractions.

Example:

```
         SUB      R6,R4,R2
         SBC      R7,R5,R3            ; R7:R6 = R5:R4 - R3:R2
         BCC                          ; conditional branch on 32 bit subtraction
```

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000 and Z was set before this operation; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RS1[15] \,\&\, \overline{RS2[15]} \,\&\, \overline{RD[15]}_{new} \mid \overline{RS1[15]} \,\&\, RS2[15] \,\&\, RD[15]_{new}$

C: Set if there is a carry from bit 15 of the result; cleared otherwise.
$\overline{RS1[15]} \,\&\, RS2[15] \mid \overline{RS1[15]} \,\&\, RD[15]_{new} \mid RS2[15] \,\&\, RD[15]_{new}$

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
SBC RD, RS1, RS2	TRI	0	0	0	1	1	RD	RS1	RS2	0	1	P

SEX Sign Extend Byte to Word SEX

Operation

The result in RD is the 16 bit sign extended representation of the original two’s complement number in the low byte of RD.L.

CCR Effects

N	Z	V	C
Δ	Δ	0	—

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code														Cycles
SEX RD	MON	0	0	0	0	0	RD	1	1	1	1	0	1	0	0	P

SIF Set Interrupt Flag SIF

Operation

Sets the Interrupt Flag of an XGATE Channel. This instruction supports two source forms. If inherent address mode is used, then the interrupt flag of the current channel (XGCHID) will be set. If the monadic address form is used, the interrupt flag associated with the channel id number contained in RS[6:0] is set. The content of RS[15:7] is ignored.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code																Cycles
SIF	INH	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	PA
SIF RS	MON	0	0	0	0	0	RS			1	1	1	1	0	1	1	1	PA

SSEM Set Semaphore SSEM

Operation

Attempts to set a semaphore. The state of the semaphore will be stored in the Carry-Flag:

1 = Semaphore is locked by the RISC core
0 = Semaphore is locked by the S12X_CPU

In monadic address mode, bits RS[2:0] select the semaphore to be set.

CCR Effects

N	Z	V	C
—	—	—	Δ

N: Not affected.
Z: Not affected.
V: Not affected.
C: Set if semaphore is locked by the RISC core; cleared otherwise.

Code and CPU Cycles

Source Form	Address Mode	Machine Code														Cycles
SSEM #IMM3	IMM3	0	0	0	0	0	IMM3	1	1	1	1	0	0	1	0	PA
SSEM RS	MON	0	0	0	0	0	RS	1	1	1	1	0	0	1	1	PA

STB Store Byte to Memory (Low Byte) STB

Operation

RS.L ⇒ M[RB, #OFFS5]
RS.L ⇒ M[RB, RI]
RS.L ⇒ M[RB, RI]; RI+1 ⇒ RI;
RI–1 ⇒ RI; RS.L ⇒ M[RB, RI][1]

Stores the low byte of register RD to memory.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
STB RS, (RB, #OFFS5),	IDO5	0	1	0	1	0	RS	RB	OFFS5			Pw
STB RS, (RB, RI)	IDR	0	1	1	1	0	RS	RB	RI	0	0	Pw
STB RS, (RB, RI+)	IDR+	0	1	1	1	0	RS	RB	RI	0	1	Pw
STB RS, (RB, -RI)	-IDR	0	1	1	1	0	RS	RB	RI	1	0	Pw

---

[1] If the same general purpose register is used as index (RI) and source register (RS), the unmodified content of the source register is written to the memory: RS.L ⇒ M[RB, RS-1]; RS-1 ⇒ RS

STW Store Word to Memory STW

Operation

$RS \Rightarrow M[RB, \#OFFS5]$
$RS \Rightarrow M[RB, RI]$
$RS \Rightarrow M[RB, RI]; RI+2 \Rightarrow RI;$
$RI-2 \Rightarrow RI; RS \Rightarrow M[RB, RI]$[1]

Stores the content of register RS to memory.

CCR Effects

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
STW RS, (RB, #OFFS5)	IDO5	0	1	0	1	1	RS	RB	OFFS5			PW
STW RS, (RB, RI)	IDR	0	1	1	1	1	RS	RB	RI	0	0	PW
STW RS, (RB, RI+)	IDR+	0	1	1	1	1	RS	RB	RI	0	1	PW
STW RS, (RB, -RI)	-IDR	0	1	1	1	1	RS	RB	RI	1	0	PW

---

[1] If the same general purpose register is used as index (RI) and source register (RS), the unmodified content of the source register is written to the memory: $RS \Rightarrow M[RB, RS-2]; RS-2 \Rightarrow RS$

SUB Subtract without Carry SUB

Operation

RS1 – RS2 ⇒ RD
RD – IMM16 ⇒ RD (translates to SUBL RD, #IMM16[7:0]; SUBH RD, #IMM16{15:8])

Subtracts two 16 bit values and stores the result in the destination register RD.

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RS1[15] \,\&\, \overline{RS2[15]} \,\&\, \overline{RD[15]}_{new} \mid \overline{RS1[15]} \,\&\, RS2[15] \,\&\, RD[15]_{new}$
Refer to SUBH instruction for #IMM16 operations.

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$\overline{RS1[15]} \,\&\, RS2[15] \mid \overline{RS1[15]} \,\&\, RD[15]_{new} \mid RS2[15] \,\&\, RD[15]_{new}$
Refer to SUBH instruction for #IMM16 operations.

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
SUB RD, RS1, RS2	TRI	0	0	0	1	1	RD	RS1	RS2	0	0	P
SUB RD, #IMM16	IMM8	1	1	0	0	0	RD	IMM16[7:0]				P
	IMM8	1	1	0	0	1	RD	IMM16[15:8]				P

SUBH Subtract Immediate 8 bit Constant (High Byte) SUBH

Operation

$RD - IMM8{:}\$00 \Rightarrow RD$

Subtracts a signed immediate 8 bit constant from the content of high byte of register RD and using binary subtraction and stores the result in the high byte of destination register RD. This instruction can be used after an SUBL for a 16 bit immediate subtraction.

Example:

```
SUBL     R2,#LOWBYTE
SUBH     R2,#HIGHBYTE      ; R2 = R2 - 16 bit immediate
```

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is $0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RD[15]_{old} \;\&\; \overline{IMM8[7]} \;\&\; \overline{RD[15]_{new}} \mid \overline{RD[15]_{old}} \;\&\; IMM8[7] \;\&\; RD[15]_{new}$

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$\overline{RD[15]_{old}} \;\&\; IMM8[7] \mid \overline{RD[15]_{old}} \;\&\; RD[15]_{new} \mid IMM8[7] \;\&\; RD[15]_{new}$

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
SUBH RD, #IMM8	IMM8	1	1	0	0	1	RD	IMM8	P

SUBL Subtract Immediate 8 bit Constant (Low Byte) SUBL

Operation

RD – \$00:IMM8 ⇒ RD

Subtracts an immediate 8 bit constant from the content of register RD using binary subtraction and stores the result in the destination register RD.

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is \$0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the 8 bit operation; cleared otherwise.
$RD[15]_{old}$ & $\overline{RD[15]}_{new}$

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$\overline{RD[15]}_{old}$ & $RD[15]_{new}$

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
SUBL RD, #IMM8	IMM8	1	1	0	0	0	RD	IMM8	P

TFR Transfer from and to Special Registers TFR

Operation

TFR RD,CCR: CCR ⇒ RD[3:0]; 0 ⇒ RD[15:4]
TFR CCR,RD: RD[3:0] ⇒ CCR
TFR RD,PC: PC+4 ⇒ RD

Transfers the content of one RISC core register to another.
The TFR RD,PC instruction can be used to implement relative subroutine calls.

Example:

```
         TFR      R7,PC    ;Return address (RETADDR) is stored in R7
         BRA      SUBR     ;Relative branch to subroutine (SUBR)
RETADDR  ...

SUBR     ...
         JAL      R7       ;Jump to return address (RETADDR)
```

CCR Effects

TFR RD,CCR, TFR RD,PC:

N	Z	V	C
—	—	—	—

N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

TFR CCR,RS:

N	Z	V	C
Δ	Δ	Δ	Δ

N: RS[3].
Z: RS[2].
V: RS[1].
C: RS[0].

Code and CPU Cycles

Source Form	Address Mode	Machine Code														Cycles
TFR RD,CCR CCR ⇒ RD	MON	0	0	0	0	0	RD	1	1	1	1	1	0	0	0	P
TFR CCR,RS RS ⇒ CCR	MON	0	0	0	0	0	RS	1	1	1	1	1	0	0	1	P
TFR RD,PCPC+4 ⇒ RD	MON	0	0	0	0	0	RD	1	1	1	1	1	0	1	0	P

TST Test Register TST

Operation

RS – 0 ⇒ NONE (translates to SUB R0, RS, R0)

Subtracts zero from the content of register RS using binary subtraction and discards the result.

CCR Effects

N	Z	V	C
Δ	Δ	Δ	Δ

N: Set if bit 15 of the result is set; cleared otherwise.

Z: Set if the result is \$0000; cleared otherwise.

V: Set if a two´s complement overflow resulted from the operation; cleared otherwise.
$RS[15] \& \overline{result[15]}$

C: Set if there is a carry from the bit 15 of the result; cleared otherwise.
$\overline{RS1[15]} \& result[15]$

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
TST RS	TRI	0	0	0	1	1	0 0 0	RS1	0 0 0	0	0	P

XNOR Logical Exclusive NOR XNOR

Operation

~(RS1 ^ RS2) ⇒ RD
~(RD ^ IMM16)⇒ RD
(translates to XNOR RD, #IMM16{15:8]; XNOR RD, #IMM16[7:0])

Performs a bit wise logical exclusive NOR between two 16 bit values and stores the result in the destination register RD.

Remark: Using R0 as a source registers will calculate the one's complement of the other source register. Using R0 as both source operands will fill RD with $FFFF.

CCR Effects

N	Z	V	C
Δ	Δ	0	—

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the result is $0000; cleared otherwise.
Refer to XNORH instruction for #IMM16 operations.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code										Cycles
XNOR RD, RS1, RS2	TRI	0	0	0	1	0	RD	RS1	RS2	1	1	P
XNOR RD, #IMM16	IMM8	1	0	1	1	0	RD	IMM16[7:0]				P
	IMM8	1	0	1	1	1	RD	IMM16[15:8]				P

XNORH Logical Exclusive NOR Immediate 8 bit Constant (High Byte) XNORH

Operation

~(RD.H ^ IMM8) ⇒ RD.H

Performs a bit wise logical exclusive NOR between the high byte of register RD and an immediate 8 bit constant and stores the result in the destination register RD.H. The low byte of RD is not affected.

CCR Effects

N	Z	V	C
Δ	Δ	0	—

N: Set if bit 15 of the result is set; cleared otherwise.
Z: Set if the 8 bit result is $00; cleared otherwise.
V: 0; cleared.
C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
XNORH RD, #IMM8	IMM8	1	0	1	1	1	RD	IMM8	P

XNORL Logical Exclusive NOR Immediate 8 bit Constant (Low Byte) XNORL

Operation

~(RD.L ^ IMM8) ⇒ RD.L

Performs a bit wise logical exclusive NOR between the low byte of register RD and an immediate 8 bit constant and stores the result in the destination register RD.L. The high byte of RD is not affected.

CCR Effects

N	Z	V	C
Δ	Δ	0	—

N:	Set if bit 7 of the result is set; cleared otherwise.
Z:	Set if the 8 bit result is $00; cleared otherwise.
V:	0; cleared.
C:	Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
XNORL RD, #IMM8	IMM8	1	0	1	1	0	RD	IMM8	P

5.8.6 Instruction Coding

Table 5-17 summarizes all XGATE instructions in the order of their machine coding.

Table 5-17. Instruction Set Summary (Sheet 1 of 3)

Functionality	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Return to Scheduler and Others																
BRK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NOP	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
RTS	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
SIF	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
Semaphore Instructions																
CSEM IMM3	0	0	0	0	0	IMM3			1	1	1	1	0	0	0	0
CSEM RS	0	0	0	0	0	RS			1	1	1	1	0	0	0	1
SSEM IMM3	0	0	0	0	0	IMM3			1	1	1	1	0	0	1	0
SSEM RS	0	0	0	0	0	RS			1	1	1	1	0	0	1	1
Single Register Instructions																
SEX RD	0	0	0	0	0	RD			1	1	1	1	0	1	0	0
PAR RD	0	0	0	0	0	RD			1	1	1	1	0	1	0	1
JAL RD	0	0	0	0	0	RD			1	1	1	1	0	1	1	0
SIF RS	0	0	0	0	0	RS			1	1	1	1	0	1	1	1
Special Move instructions																
TFR RD,CCR	0	0	0	0	0	RD			1	1	1	1	1	0	0	0
TFR CCR,RS	0	0	0	0	0	RS			1	1	1	1	1	0	0	1
TFR RD,PC	0	0	0	0	0	RD			1	1	1	1	1	0	1	0
Shift instructions Dyadic																
BFFO RD, RS	0	0	0	0	1	RD			RS			1	0	0	0	0
ASR RD, RS	0	0	0	0	1	RD			RS			1	0	0	0	1
CSL RD, RS	0	0	0	0	1	RD			RS			1	0	0	1	0
CSR RD, RS	0	0	0	0	1	RD			RS			1	0	0	1	1
LSL RD, RS	0	0	0	0	1	RD			RS			1	0	1	0	0
LSR RD, RS	0	0	0	0	1	RD			RS			1	0	1	0	1
ROL RD, RS	0	0	0	0	1	RD			RS			1	0	1	1	0
ROR RD, RS	0	0	0	0	1	RD			RS			1	0	1	1	1
Shift instructions immediate																
ASR RD, #IMM4	0	0	0	0	1	RD			IMM4				1	0	0	1
CSL RD, #IMM4	0	0	0	0	1	RD			IMM4				1	0	1	0
CSR RD, #IMM4	0	0	0	0	1	RD			IMM4				1	0	1	1
LSL RD, #IMM4	0	0	0	0	1	RD			IMM4				1	1	0	0
LSR RD, #IMM4	0	0	0	0	1	RD			IMM4				1	1	0	1
ROL RD, #IMM4	0	0	0	0	1	RD			IMM4				1	1	1	0
ROR RD, #IMM4	0	0	0	0	1	RD			IMM4				1	1	1	1

Table 5-17. Instruction Set Summary (Sheet 2 of 3)

Functionality	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Logical Triadic																
AND RD, RS1, RS2	0	0	0	1	0	RD			RS1			RS2			0	0
OR RD, RS1, RS2	0	0	0	1	0	RD			RS1			RS2			1	0
XNOR RD, RS1, RS2	0	0	0	1	0	RD			RS1			RS2			1	1
Arithmetic Triadic	For compare use SUB R0,Rs1,Rs2															
SUB RD, RS1, RS2	0	0	0	1	1	RD			RS1			RS2			0	0
SBC RD, RS1, RS2	0	0	0	1	1	RD			RS1			RS2			0	1
ADD RD, RS1, RS2	0	0	0	1	1	RD			RS1			RS2			1	0
ADC RD, RS1, RS2	0	0	0	1	1	RD			RS1			RS2			1	1
Branches																
BCC REL9	0	0	1	0	0	0	0	REL9								
BCS REL9	0	0	1	0	0	0	1	REL9								
BNE REL9	0	0	1	0	0	1	0	REL9								
BEQ REL9	0	0	1	0	0	1	1	REL9								
BPL REL9	0	0	1	0	1	0	0	REL9								
BMI REL9	0	0	1	0	1	0	1	REL9								
BVC REL9	0	0	1	0	1	1	0	REL9								
BVS REL9	0	0	1	0	1	1	1	REL9								
BHI REL9	0	0	1	1	0	0	0	REL9								
BLS REL9	0	0	1	1	0	0	1	REL9								
BGE REL9	0	0	1	1	0	1	0	REL9								
BLT REL9	0	0	1	1	0	1	1	REL9								
BGT REL9	0	0	1	1	1	0	0	REL9								
BLE REL9	0	0	1	1	1	0	1	REL9								
BRA REL10	0	0	1	1	1	1	REL10									
Load and Store Instructions																
LDB RD, (RB, #OFFS5)	0	1	0	0	0	RD			RB			OFFS5				
LDW RD, (RB, #OFFS5)	0	1	0	0	1	RD			RB			OFFS5				
STB RS, (RB, #OFFS5)	0	1	0	1	0	RS			RB			OFFS5				
STW RS, (RB, #OFFS5)	0	1	0	1	1	RS			RB			OFFS5				
LDB RD, (RB, RI)	0	1	1	0	0	RD			RB			RI			0	0
LDW RD, (RB, RI)	0	1	1	0	1	RD			RB			RI			0	0
STB RS, (RB, RI)	0	1	1	1	0	RS			RB			RI			0	0
STW RS, (RB, RI)	0	1	1	1	1	RS			RB			RI			0	0
LDB RD, (RB, RI+)	0	1	1	0	0	RD			RB			RI			0	1
LDW RD, (RB, RI+)	0	1	1	0	1	RD			RB			RI			0	1
STB RS, (RB, RI+)	0	1	1	1	0	RS			RB			RI			0	1
STW RS, (RB, RI+)	0	1	1	1	1	RS			RB			RI			0	1
LDB RD, (RB, –RI)	0	1	1	0	0	RD			RB			RI			1	0
LDW RD, (RB, –RI)	0	1	1	0	1	RD			RB			RI			1	0
STB RS, (RB, –RI)	0	1	1	1	0	RS			RB			RI			1	0
STW RS, (RB, –RI)	0	1	1	1	1	RS			RB			RI			1	0
Bit Field Instructions																
BFEXT RD, RS1, RS2	0	1	1	0	0	RD			RS1			RS2			1	1
BFINS RD, RS1, RS2	0	1	1	0	1	RD			RS1			RS2			1	1
BFINSI RD, RS1, RS2	0	1	1	1	0	RD			RS1			RS2			1	1
BFINSX RD, RS1, RS2	0	1	1	1	1	RD			RS1			RS2			1	1
Logic Immediate Instructions																
ANDL RD, #IMM8	1	0	0	0	0	RD			IMM8							

Table 5-17. Instruction Set Summary (Sheet 3 of 3)

Functionality	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ANDH RD, #IMM8	1	0	0	0	1	RD			IMM8							
BITL RD, #IMM8	1	0	0	1	0	RD			IMM8							
BITH RD, #IMM8	1	0	0	1	1	RD			IMM8							
ORL RD, #IMM8	1	0	1	0	0	RD			IMM8							
ORH RD, #IMM8	1	0	1	0	1	RD			IMM8							
XNORL RD, #IMM8	1	0	1	1	0	RD			IMM8							
XNORH RD, #IMM8	1	0	1	1	1	RD			IMM8							
Arithmetic Immediate Instructions																
SUBL RD, #IMM8	1	1	0	0	0	RD			IMM8							
SUBH RD, #IMM8	1	1	0	0	1	RD			IMM8							
CMPL RS, #IMM8	1	1	0	1	0	RS			IMM8							
CPCH RS, #IMM8	1	1	0	1	1	RS			IMM8							
ADDL RD, #IMM8	1	1	1	0	0	RD			IMM8							
ADDH RD, #IMM8	1	1	1	0	1	RD			IMM8							
LDL RD, #IMM8	1	1	1	1	0	RD			IMM8							
LDH RD, #IMM8	1	1	1	1	1	RD			IMM8							

5.9 Initialization and Application Information

5.9.1 Initialization

The recommended initialization of the XGATE is as follows:

1. Clear the XGE bit to suppress any incoming service requests.
2. Make sure that no thread is running on the XGATE. This can be done in several ways:
    a) Poll the XGCHID register until it reads $00. Also poll XGDBG and XGSWEIF to make sure that the XGATE has not been stopped.
    b) Enter Debug Mode by setting the XGDBG bit. Clear the XGCHID register. Clear the XGDBG bit.

    The recommended method is a).
3. Set the XGVBR register to the lowest address of the XGATE vector space.
4. Clear all Channel ID flags.
5. Copy XGATE vectors and code into the RAM.
6. Initialize the S12X_INT module.
7. Enable the XGATE by setting the XGE bit.

The following code example implements the XGATE initialization sequence.

5.9.2 Code Example (Transmit "Hello World!" on SCI)

```
                        CPU   S12X
                        ;############################################
                        ;#                  SYMBOLS                 #
                        ;############################################
SCI_REGS                EQU   $00C8           ;SCI register space
SCIBDH                  EQU   SCI_REGS+$00    ;SCI Baud Rate Register
SCIBDL                  EQU   SCI_REGS+$00    ;SCI Baud Rate Register
SCICR2                  EQU   SCI_REGS+$03    ;SCI Control Register 2
SCISR1                  EQU   SCI_REGS+$04    ;SCI Status  Register 1
SCIDRL                  EQU   SCI_REGS+$07    ;SCI Control Register 2
TIE                     EQU   $80             ;TIE bit mask
TE                      EQU   $08             ;TE bit mask
RE                      EQU   $04             ;RE bit mask
SCI_VEC                 EQU   $D6             ;SCI vector number

INT_REGS                      EQU             $0120           ;S12X_INT register space
INT_CFADDR              EQU   INT_REGS+$07    ;Interrupt Configuration Address Register
INT_CFDATA              EQU   INT_REGS+$08    ;Interrupt Configuration Data Registers
RQST                    EQU   $80             ;RQST bit mask

XGATE_REGS              EQU   $0380           ;XGATE register space
XGMCTL                  EQU   XGATE_REGS+$00  ;XGATE Module Control Register
XGMCTL_CLEAR            EQU   $FA02           ;Clear all XGMCTL bits
XGMCTL_ENABLE           EQU   $8282           ;Enable XGATE
XGCHID                  EQU   XGATE_REGS+$02  ;XGATE Channel ID Register
XGVBR                   EQU   XGATE_REGS+$06  ;XGATE ISP Select Register
XGIF                    EQU   XGATE_REGS+$08  ;XGATE Interrupt Flag Vector
```

```
XGSWT                   EQU   XGATE_REGS+$18  ;XGATE Software Trigger Register
XGSEM                   EQU   XGATE_REGS+$1A  ;XGATE Semaphore Register

RPAGE                   EQU   $0016

RAM_SIZE                EQU   32*$400         ;32k RAM

RAM_START               EQU   $1000
RAM_START_XG            EQU   $10000-RAM_SIZE
RAM_START_GLOB          EQU   $100000-RAM_SIZE

XGATE_VECTORS           EQU   RAM_START
XGATE_VECTORS_XG        EQU   RAM_START_XG

XGATE_DATA              EQU   RAM_START+(4*128)
XGATE_DATA_XG           EQU   RAM_START_XG+(4*128)

XGATE_CODE              EQU   XGATE_DATA+(XGATE_CODE_FLASH-XGATE_DATA_FLASH)
XGATE_CODE_XG           EQU   XGATE_DATA_XG+(XGATE_CODE_FLASH-XGATE_DATA_FLASH)

BUS_FREQ_HZ             EQU   40000000

                        ;###########################################
                        ;#           S12XE VECTOR TABLE            #
                        ;###########################################
                        ORG   $FF10 ;non-maskable interrupts
                        DW    DUMMY_ISR DUMMY_ISR DUMMY_ISR DUMMY_ISR

                        ORG   $FFF4 ;non-maskable interrupts
                        DW    DUMMY_ISR DUMMY_ISR DUMMY_ISR

                        ;###########################################
                        ;#                DISABLE COP              #
                        ;###########################################
                        ORG   $FF0E
                        DW    $FFFE

                        ORG   $C000
START_OF_CODE
                        ;###########################################
                        ;#          INITIALIZE S12XE CORE          #
                        ;###########################################
                        SEI
                        MOVB  #(RAM_START_GLOB>>12), RPAGE;set RAM page

                        ;###########################################
                        ;#              INITIALIZE SCI             #
                        ;###########################################
INIT_SCI                MOVW  #(BUS_FREQ_HZ/(16*9600)), SCIBDH;set baud rate
                        MOVB  #(TIE|TE), SCICR2;enable tx buffer empty interrupt

                        ;###########################################
                        ;#           INITIALIZE S12X_INT           #
                        ;###########################################
INIT_INT                MOVB  #(SCI_VEC&$F0), INT_CFADDR ;switch SCI interrupts to XGATE
                        MOVB  #RQST|$01, INT_CFDATA+((SCI_VEC&$0F)>>1)
```

```
                        ;###########################################
                        ;#              INITIALIZE XGATE           #
                        ;###########################################
INIT_XGATE              MOVW #XGMCTL_CLEAR , XGMCTL;clear all XGMCTL bits

INIT_XGATE_BUSY_LOOP    TST  XGCHID          ;wait until current thread is finished
                        BNE  INIT_XGATE_BUSY_LOOP

                        LDX  #XGIF           ;clear all channel interrupt flags
                        LDD  #$FFFF
                        STD  2,X+
                        STD  2,X+
                        STD  2,X+
                        STD  2,X+
                        STD  2,X+
                        STD  2,X+
                        STD  2,X+
                        STD  2,X+

                        MOVW #XGATE_VECTORS_XG, XGVBR;set vector base register

                        MOVW #$FF00, XGSWT   ;clear all software triggers

                        ;###########################################
                        ;#       INITIALIZE XGATE VECTOR TABLE     #
                        ;###########################################
                        LDAA #128            ;build XGATE vector table
                        LDY  #XGATE_VECTORS
INIT_XGATE_VECTAB_LOOP  MOVW #XGATE_DUMMY_ISR_XG, 4,Y+
                        DBNE A, INIT_XGATE_VECTAB_LOOP

                        MOVW #XGATE_CODE_XG, RAM_START+(2*SCI_VEC)
                        MOVW #XGATE_DATA_XG, RAM_START+(2*SCI_VEC)+2

                        ;###########################################
                        ;#              COPY XGATE CODE            #
                        ;###########################################
COPY_XGATE_CODE         LDX  #XGATE_DATA_FLASH
COPY_XGATE_CODE_LOOP    MOVW 2,X+, 2,Y+
                        MOVW 2,X+, 2,Y+
                        MOVW 2,X+, 2,Y+
                        MOVW 2,X+, 2,Y+
                        CPX  #XGATE_CODE_FLASH_END
                        BLS  COPY_XGATE_CODE_LOOP

                        ;###########################################
                        ;#                START XGATE              #
                        ;###########################################
START_XGATE             MOVW #XGMCTL_ENABLE, XGMCTL;enable XGATE
                        BRA  *

                        ;###########################################
                        ;#     DUMMY INTERRUPT SERVICE ROUTINE     #
                        ;###########################################
DUMMY_ISR               RTI

                        CPU  XGATE
```

```
                        ;###########################################
                        ;#                XGATE DATA               #
                        ;###########################################
                        ALIGN 1
XGATE_DATA_FLASH        EQU   *
XGATE_DATA_SCI          EQU   *-XGATE_DATA_FLASH
                        DW    SCI_REGS        ;pointer to SCI register space
XGATE_DATA_IDX          EQU   *-XGATE_DATA_FLASH
                        DB    XGATE_DATA_MSG  ;string pointer
XGATE_DATA_MSG          EQU   *-XGATE_DATA_FLASH
                        FCC   "Hello World!   ;ASCII string
                        DB    $0D             ;CR

                        ;###########################################
                        ;#               XGATE CODE                #
                        ;###########################################
                        ALIGN 1
XGATE_CODE_FLASH        LDW   R2,(R1,#XGATE_DATA_SCI);SCI -> R2
                        LDB   R3,(R1,#XGATE_DATA_IDX);msg -> R3
                        LDB   R4,(R1,R3+)     ;curr. char -> R4
                        STB   R3,(R1,#XGATE_DATA_IDX);R3 -> idx
                        LDB   R0,(R2,#(SCISR1-SCI_REGS));initiate SCI transmit
                        STB   R4,(R2,#(SCIDRL-SCI_REGS));initiate SCI transmit
                        CMPL  R4,#$0D
                        BEQ   XGATE_CODE_DONE
                        RTS
XGATE_CODE_DONE         LDL   R4,#$00                         ;disable SCI interrupts
                        STB   R4,(R2,#(SCICR2-SCI_REGS))
                        LDL   R3,#XGATE_DATA_MSG;reset R3
                        STB   R3,(R1,#XGATE_DATA_IDX)
XGATE_CODE_FLASH_END    RTS
XGATE_DUMMY_ISR_XG      EQU   (XGATE_CODE_FLASH_END-XGATE_CODE_FLASH)+XGATE_CODE_XG
```

Chapter 6
Security (S12X9SECV2)

Table 6-1. Revision History

Revision Number	Revision Date	Sections Affected	Description of Changes
02.00	27 Aug 2004		reviewed and updated for S12XD architecture
02.01	21 Feb 2007		added S12XE, S12XF and S12XS architectures
02.02	19 Apr 2007		corrected statement about Backdoor key access via BDM on XE, XF, XS

6.1 Introduction

This specification describes the function of the security mechanism in the S12XD chip family (9SEC).

NOTE

> No security feature is absolutely secure. However, Freescale's strategy is to make reading or copying the FLASH and/or EEPROM difficult for unauthorized users.

6.1.1 Features

The user must be reminded that part of the security must lie with the application code. An extreme example would be application code that dumps the contents of the internal memory. This would defeat the purpose of security. At the same time, the user may also wish to put a backdoor in the application program. An example of this is the user downloads a security key through the SCI, which allows access to a programming routine that updates parameters stored in another section of the Flash memory.

The security features of the S12X chip family (in secure mode) are:

- Protect the content of non-volatile memories (Flash, EEPROM)
- Execution of NVM commands is restricted
- Disable access to internal memory via background debug module (BDM)
- Disable access to internal Flash/EEPROM in expanded modes
- Disable debugging features for the CPU and XGATE

6.1.2 Modes of Operation

Table 6-2 gives an overview over availability of security relevant features in unsecure and secure modes.

Table 6-2. Feature Availability in Unsecure and Secure Modes on S12XD

	Unsecure Mode						Secure Mode					
	NS	**SS**	**NX**	**ES**	**EX**	**ST**	**NS**	**SS**	**NX**	**ES**	**EX**	**ST**
Flash Array Access	✔	✔	✔(1)	✔[1]	✔[1]	✔[1]	✔	✔	—	—	—	—
EEPROM Array Access	✔	✔	✔	✔	✔	✔	✔	✔	—	—	—	—
NVM Commands	✔(2)	✔	✔[2]	✔[2]	✔[2]	✔	✔[2]	✔[2]	✔[2]	✔[2]	✔[2]	✔[2]
BDM	✔	✔	✔	✔	✔	✔	—	✔(3)	—	—	—	—
DBG Module Trace	✔	✔	✔	✔	✔	✔	—	—	—	—	—	—
XGATE Debugging	✔	✔	✔	✔	✔	✔	—	—	—	—	—	—
External Bus Interface	—	—	✔	✔	✔	✔	—	—	✔	✔	✔	✔
Internal status visible multiplexed on external bus	—	—	—	✔	✔	—	—	—	—	✔	✔	—
Internal accesses visible on external bus	—	—	—	—	—	✔	—	—	—	—	—	✔

1. Availability of Flash arrays in the memory map depends on ROMCTL/EROMCTL pins and/or the state of the ROMON/EROMON bits in the MMCCTL1 register. Please refer to the S12X_MMC block guide for detailed information.
2. Restricted NVM command set only. Please refer to the NVM wrapper block guides for detailed information.
3. BDM hardware commands restricted to peripheral registers only.

6.1.3 Securing the Microcontroller

Once the user has programmed the Flash and EEPROM, the chip can be secured by programming the security bits located in the options/security byte in the Flash memory array. These non-volatile bits will keep the device secured through reset and power-down.

The options/security byte is located at address 0xFF0F (= global address 0x7F_FF0F) in the Flash memory array. This byte can be erased and programmed like any other Flash location. Two bits of this byte are used for security (SEC[1:0]). On devices which have a memory page window, the Flash options/security byte is also available at address 0xBF0F by selecting page 0x3F with the PPAGE register. The contents of this byte are copied into the Flash security register (FSEC) during a reset sequence.

	7	6	5	4	3	2	1	0
0xFF0F	KEYEN1	KEYEN0	NV5	NV4	NV3	NV2	SEC1	SEC0

Figure 6-1. Flash Options/Security Byte

The meaning of the bits KEYEN[1:0] is shown in Table 6-3. Please refer to Section 6.1.5.1, “Unsecuring the MCU Using the Backdoor Key Access” for more information.

Table 6-3. Backdoor Key Access Enable Bits

KEYEN[1:0]	Backdoor Key Access Enabled
00	0 (disabled)
01	0 (disabled)
10	1 (enabled)
11	0 (disabled)

The meaning of the security bits SEC[1:0] is shown in Table 6-4. For security reasons, the state of device security is controlled by two bits. To put the device in unsecured mode, these bits must be programmed to SEC[1:0] = ‘10’. All other combinations put the device in a secured mode. The recommended value to put the device in secured state is the inverse of the unsecured state, i.e. SEC[1:0] = ‘01’.

Table 6-4. Security Bits

SEC[1:0]	Security State
00	1 (secured)
01	1 (secured)
10	***0 (unsecured)***
11	1 (secured)

NOTE

Please refer to the Flash block guide for actual security configuration (in section “Flash Module Security”).

6.1.4 Operation of the Secured Microcontroller

By securing the device, unauthorized access to the EEPROM and Flash memory contents can be prevented. However, it must be understood that the security of the EEPROM and Flash memory contents also depends on the design of the application program. For example, if the application has the capability of downloading code through a serial port and then executing that code (e.g. an application containing bootloader code), then this capability could potentially be used to read the EEPROM and Flash memory contents even when the microcontroller is in the secure state. In this example, the security of the application could be enhanced by requiring a challenge/response authentication before any code can be downloaded.

Secured operation has the following effects on the microcontroller:

6.1.4.1 Normal Single Chip Mode (NS)

- Background debug module (BDM) operation is completely disabled.
- Execution of Flash and EEPROM commands is restricted. Please refer to the NVM block guide for details.
- Tracing code execution using the DBG module is disabled.
- Debugging XGATE code (breakpoints, single-stepping) is disabled.

6.1.4.2 Special Single Chip Mode (SS)

- BDM firmware commands are disabled.
- BDM hardware commands are restricted to the register space.
- Execution of Flash and EEPROM commands is restricted. Please refer to the NVM block guide for details.
- Tracing code execution using the DBG module is disabled.
- Debugging XGATE code (breakpoints, single-stepping) is disabled.

Special single chip mode means BDM is active after reset. The availability of BDM firmware commands depends on the security state of the device. The BDM secure firmware first performs a blank check of both the Flash memory and the EEPROM. If the blank check succeeds, security will be temporarily turned off and the state of the security bits in the appropriate Flash memory location can be changed If the blank check fails, security will remain active, only the BDM hardware commands will be enabled, and the accessible memory space is restricted to the peripheral register area. This will allow the BDM to be used to erase the EEPROM and Flash memory without giving access to their contents. After erasing both Flash memory and EEPROM, another reset into special single chip mode will cause the blank check to succeed and the options/security byte can be programmed to “unsecured” state via BDM.

While the BDM is executing the blank check, the BDM interface is completely blocked, which means that all BDM commands are temporarily blocked.

6.1.4.3 Expanded Modes (NX, ES, EX, and ST)

- BDM operation is completely disabled.
- Internal Flash memory and EEPROM are disabled.
- Execution of Flash and EEPROM commands is restricted. Please refer to the NVM block guide for details.
- Tracing code execution using the DBG module is disabled.
- Debugging XGATE code (breakpoints, single-stepping) is disabled

6.1.5 Unsecuring the Microcontroller

Unsecuring the microcontroller can be done by three different methods:

1. Backdoor key access
2. Reprogramming the security bits
3. Complete memory erase (special modes)

6.1.5.1 Unsecuring the MCU Using the Backdoor Key Access

In normal modes (single chip and expanded), security can be temporarily disabled using the backdoor key access method. This method requires that:

- The backdoor key at 0xFF00–0xFF07 (= global addresses 0x7F_FF00–0x7F_FF07) has been programmed to a valid value.

- The KEYEN[1:0] bits within the Flash options/security byte select ‘enabled’.
- In single chip mode, the application program programmed into the microcontroller must be designed to have the capability to write to the backdoor key locations.

The backdoor key values themselves would not normally be stored within the application data, which means the application program would have to be designed to receive the backdoor key values from an external source (e.g. through a serial port). It is not possible to download the backdoor keys using background debug mode.

The backdoor key access method allows debugging of a secured microcontroller without having to erase the Flash. This is particularly useful for failure analysis.

NOTE

> No word of the backdoor key is allowed to have the value 0x0000 or 0xFFFF.

6.1.5.2 Backdoor Key Access Sequence

These are the necessary steps for a successful backdoor key access sequence:

1. Set the KEYACC bit in the Flash configuration register FCNFG.
2. Write the first 16-bit word of the backdoor key to 0xFF00 (0x7F_FF00).
3. Write the second 16-bit word of the backdoor key to 0xFF02 (0x7F_FF02).
4. Write the third 16-bit word of the backdoor key to 0xFF04 (0x7F_FF04).
5. Write the fourth 16-bit word of the backdoor key to 0xFF06 (0x7F_FF06).
6. Clear the KEYACC bit in the Flash Configuration register FCNFG.

NOTE

> Flash cannot be read while KEYACC is set. Therefore the code for the backdoor key access sequence must execute from RAM.

If all four 16-bit words match the Flash contents at 0xFF00–0xFF07 (0x7F_FF00–0x7F_FF07), the microcontroller will be unsecured and the security bits SEC[1:0] in the Flash Security register FSEC will be forced to the unsecured state (‘10’). The contents of the Flash options/security byte are not changed by this procedure, and so the microcontroller will revert to the secure state after the next reset unless further action is taken as detailed below.

If any of the four 16-bit words does not match the Flash contents at 0xFF00–0xFF07 (0x7F_FF00–0x7F_FF07), the microcontroller will remain secured.

6.1.6 Reprogramming the Security Bits

In normal single chip mode (NS), security can also be disabled by erasing and reprogramming the security bits within Flash options/security byte to the unsecured value. Because the erase operation will erase the entire sector from 0xFE00–0xFFFF (0x7F_FE00–0x7F_FFFF), the backdoor key and the interrupt vectors will also be erased; this method is not recommended for normal single chip mode. The application software can only erase and program the Flash options/security byte if the Flash sector containing the Flash options/security byte is not protected (see Flash protection). Thus Flash protection is a useful means of preventing this method. The microcontroller will enter the unsecured state after the next reset following the programming of the security bits to the unsecured value.

This method requires that:

- The application software previously programmed into the microcontroller has been designed to have the capability to erase and program the Flash options/security byte, or security is first disabled using the backdoor key method, allowing BDM to be used to issue commands to erase and program the Flash options/security byte.
- The Flash sector containing the Flash options/security byte is not protected.

6.1.7 Complete Memory Erase (Special Modes)

The microcontroller can be unsecured in special modes by erasing the entire EEPROM and Flash memory contents.

When a secure microcontroller is reset into special single chip mode (SS), the BDM firmware verifies whether the EEPROM and Flash memory are erased. If any EEPROM or Flash memory address is not erased, only BDM hardware commands are enabled. BDM hardware commands can then be used to write to the EEPROM and Flash registers to mass erase the EEPROM and all Flash memory blocks.

When next reset into special single chip mode, the BDM firmware will again verify whether all EEPROM and Flash memory are erased, and this being the case, will enable all BDM commands, allowing the Flash options/security byte to be programmed to the unsecured value. The security bits SEC[1:0] in the Flash security register will indicate the unsecure state following the next reset.

Special single chip erase and unsecure sequence:

1. Reset into special single chip mode.
2. Write an appropriate value to the ECLKDIV register for correct timing.
3. Write 0xFF to the EPROT register to disable protection.
4. Write 0x30 to the ESTAT register to clear the PVIOL and ACCERR bits.
5. Write 0x0000 to the EDATA register (0x011A–0x011B).
6. Write 0x0000 to the EADDR register (0x0118–0x0119).
7. Write 0x41 (mass erase) to the ECMD register.
8. Write 0x80 to the ESTAT register to clear CBEIF.
9. Write an appropriate value to the FCLKDIV register for correct timing.
10. Write 0x00 to the FCNFG register to select Flash block 0.
11. Write 0x10 to the FTSTMOD register (0x0102) to set the WRALL bit, so the following writes affect all Flash blocks.
12. Write 0xFF to the FPROT register to disable protection.
13. Write 0x30 to the FSTAT register to clear the PVIOL and ACCERR bits.
14. Write 0x0000 to the FDATA register (0x010A–0x010B).
15. Write 0x0000 to the FADDR register (0x0108–0x0109).
16. Write 0x41 (mass erase) to the FCMD register.
17. Write 0x80 to the FSTAT register to clear CBEIF.
18. Wait until all CCIF flags are set.
19. Reset back into special single chip mode.
20. Write an appropriate value to the FCLKDIV register for correct timing.
21. Write 0x00 to the FCNFG register to select Flash block 0.
22. Write 0xFF to the FPROT register to disable protection.
23. Write 0xFFBE to Flash address 0xFF0E.
24. Write 0x20 (program) to the FCMD register.
25. Write 0x80 to the FSTAT register to clear CBEIF.
26. Wait until the CCIF flag in FSTAT is are set.
27. Reset into any mode.

Chapter 7
Clocks and Reset Generator (S12CRGV6)

7.1 Introduction

This specification describes the function of the clocks and reset generator (CRG).

7.1.1 Features

The main features of this block are:

- Phase locked loop (PLL) frequency multiplier
  - Reference divider
  - Automatic bandwidth control mode for low-jitter operation
  - Automatic frequency lock detector
  - Interrupt request on entry or exit from locked condition
  - Self clock mode in absence of reference clock
- System clock generator
  - Clock quality check
  - User selectable fast wake-up from Stop in self-clock mode for power saving and immediate program execution
  - Clock switch for either oscillator or PLL based system clocks
- Computer operating properly (COP) watchdog timer with time-out clear window
- System reset generation from the following possible sources:
  - Power on reset
  - Low voltage reset
  - Illegal address reset
  - COP reset
  - Loss of clock reset
  - External pin reset
- Real-time interrupt (RTI)

7.1.2 Modes of Operation

This subsection lists and briefly describes all operating modes supported by the CRG.

- Run mode

  All functional parts of the CRG are running during normal run mode. If RTI or COP functionality is required, the individual bits of the associated rate select registers (COPCTL, RTICTL) have to be set to a nonzero value.

- Wait mode

  In this mode, the PLL can be disabled automatically depending on the PLLSEL bit in the CLKSEL register.

- Stop mode

  Depending on the setting of the PSTP bit, stop mode can be differentiated between full stop mode (PSTP = 0) and pseudo stop mode (PSTP = 1).

  - Full stop mode

    The oscillator is disabled and thus all system and core clocks are stopped. The COP and the RTI remain frozen.

  - Pseudo stop mode

    The oscillator continues to run and most of the system and core clocks are stopped. If the respective enable bits are set, the COP and RTI will continue to run, or else they remain frozen.

- Self clock mode

  Self clock mode will be entered if the clock monitor enable bit (CME) and the self clock mode enable bit (SCME) are both asserted and the clock monitor in the oscillator block detects a loss of clock. As soon as self clock mode is entered, the CRG starts to perform a clock quality check. Self clock mode remains active until the clock quality check indicates that the required quality of the incoming clock signal is met (frequency and amplitude). Self clock mode should be used for safety purposes only. It provides reduced functionality to the MCU in case a loss of clock is causing severe system conditions.

7.1.3 Block Diagram

Figure 7-1 shows a block diagram of the CRG.

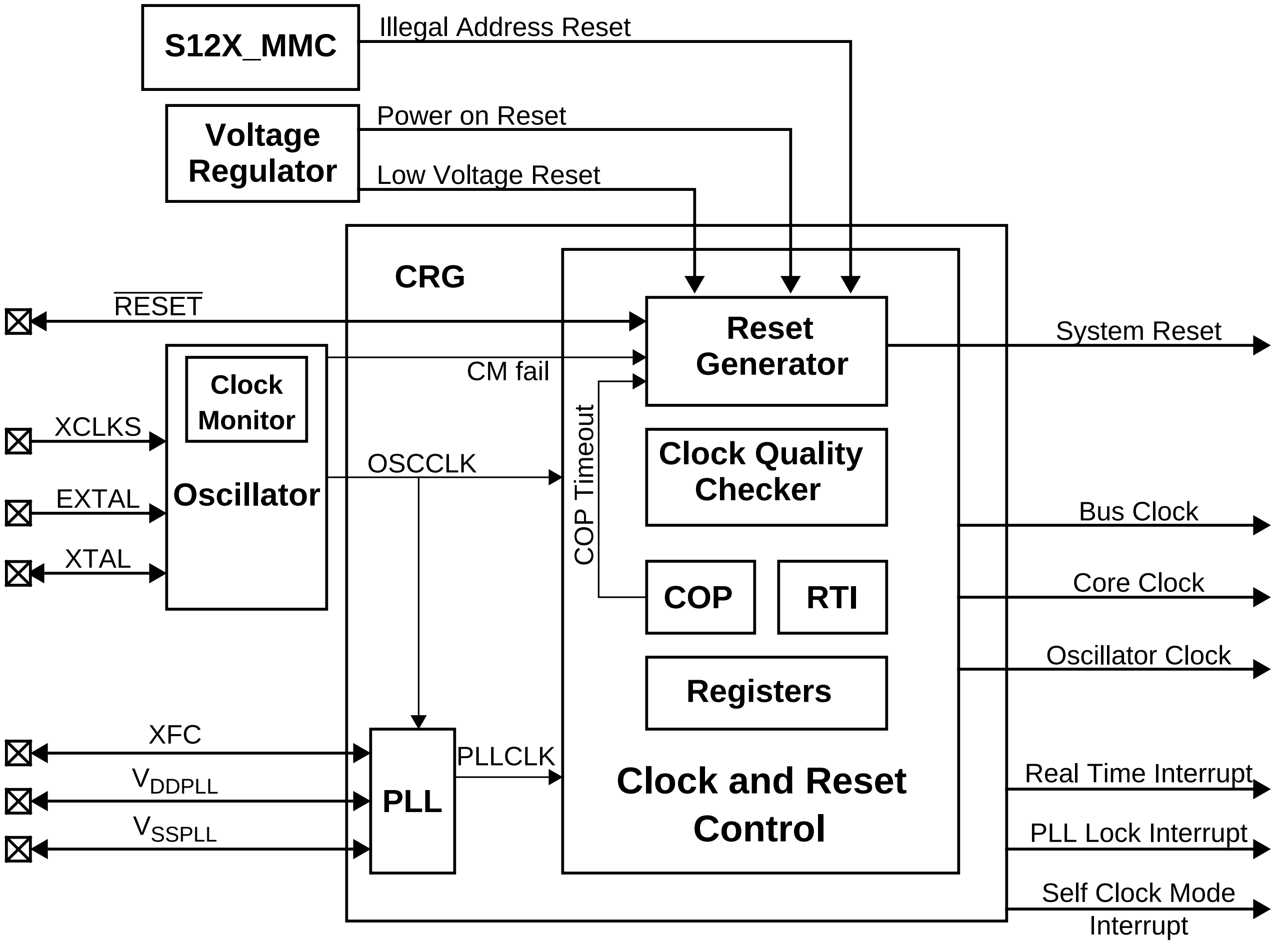


Figure 7-1. CRG Block Diagram

7.2 External Signal Description

This section lists and describes the signals that connect off chip.

7.2.1 V_{DDPLL} and V_{SSPLL} — Operating and Ground Voltage Pins

These pins provide operating voltage (V_{DDPLL}) and ground (V_{SSPLL}) for the PLL circuitry. This allows the supply voltage to the PLL to be independently bypassed. Even if PLL usage is not required, V_{DDPLL} and V_{SSPLL} must be connected to properly.

7.2.2 XFC — External Loop Filter Pin

A passive external loop filter must be placed on the XFC pin. The filter is a second-order, low-pass filter that eliminates the VCO input ripple. The value of the external filter network and the reference frequency determines the speed of the corrections and the stability of the PLL. Refer to the device specification for calculation of PLL Loop Filter (XFC) components**.** If PLL usage is not required, the XFC pin must be tied to V_{DDPLL}.

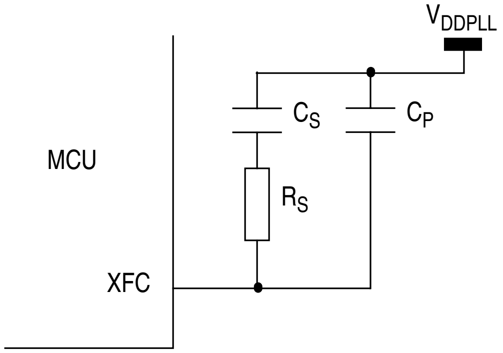


Figure 7-2. PLL Loop Filter Connections

7.2.3 $\overline{\text{RESET}}$ — Reset Pin

$\overline{\text{RESET}}$ is an active low bidirectional reset pin. As an input. it initializes the MCU asynchronously to a known start-up state. As an open-drain output, it indicates that a system reset (internal to the MCU) has been triggered.

7.3 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in the CRG.

7.3.1 Module Memory Map

Table 7-1 gives an overview on all CRG registers.

Table 7-1. CRG Memory Map

Address Offset	Use	Access
0x_00	CRG Synthesizer Register (SYNR)	R/W
0x_01	CRG Reference Divider Register (REFDV)	R/W
0x_02	CRG Test Flags Register (CTFLG)[1]	R/W
0x_03	CRG Flags Register (CRGFLG)	R/W
0x_04	CRG Interrupt Enable Register (CRGINT)	R/W
0x_05	CRG Clock Select Register (CLKSEL)	R/W
0x_06	CRG PLL Control Register (PLLCTL)	R/W
0x_07	CRG RTI Control Register (RTICTL)	R/W
0x_08	CRG COP Control Register (COPCTL)	R/W
0x_09	CRG Force and Bypass Test Register (FORBYP)[2]	R/W
0x_0A	CRG Test Control Register (CTCTL)[3]	R/W
0x_0B	CRG COP Arm/Timer Reset (ARMCOP)	R/W

[1] CTFLG is intended for factory test purposes only.

[2] FORBYP is intended for factory test purposes only.

[3] CTCTL is intended for factory test purposes only.

NOTE

Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level.

7.3.2 Register Descriptions

This section describes in address order all the CRG registers and their individual bits.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x_00 SYNR	R	0	0	SYN5	SYN4	SYN3	SYN2	SYN1	SYN0
	W	▨	▨						
0x_01 REFDV	R	0	0	REFDV5	REFDV4	REFDV3	REFDV2	REFDV1	REFDV0
	W	▨	▨						
0x_02 CTFLG	R	0	0	0	0	0	0	0	0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x_03 CRGFLG	R	RTIF	PORF	LVRF	LOCKIF	LOCK	TRACK	SCMIF	SCM
	W					▨	▨		▨
0x_04 CRGINT	R	RTIE	ILAF	0	LOCKIE	0	0	SCMIE	0
	W			▨		▨	▨		▨
0x_05 CLKSEL	R	PLLSEL	PSTP	0	0	PLLWAI	0	RTIWAI	COPWAI
	W			▨	▨		▨		
0x_06 PLLCTL	R	CME	PLLON	AUTO	ACQ	FSTWKP	PRE	PCE	SCME
	W								
0x_07 RTICTL	R	RTDEC	RTR6	RTR5	RTR4	RTR3	RTR2	RTR1	RTR0
	W								
0x_08 COPCTL	R	WCOP	RSBCK	0	0	0	CR2	CR1	CR0
	W			WRTMASK	▨	▨			
0x_09 FORBYP	R	0	0	0	0	0	0	0	0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x_0A CTCTL	R	1	0	0	0	0	0	0	0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x_0B ARMCOP	R	0	0	0	0	0	0	0	0
	W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

▨ = Unimplemented or Reserved

Figure 7-3. S12CRGV6 Register Summary

7.3.2.1 CRG Synthesizer Register (SYNR)

The SYNR register controls the multiplication factor of the PLL. If the PLL is on, the count in the loop divider (SYNR) register effectively multiplies up the PLL clock (PLLCLK) from the reference frequency by 2 x (SYNR + 1). PLLCLK will not be below the minimum VCO frequency (f_{SCM}).

$$\mathrm{PLLCLK} = 2\mathrm{x}\mathrm{OSCCLK}\mathrm{x}\frac{(\mathrm{SYNR}+1)}{(\mathrm{REFDV}+1)}$$

NOTE

If PLL is selected (PLLSEL=1), Bus Clock = PLLCLK / 2
Bus Clock must not exceed the maximum operating system frequency.

Module Base +0x_00

	7	6	5	4	3	2	1	0
R	0	0	SYN5	SYN4	SYN3	SYN2	SYN1	SYN0
W	▨	▨						
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 7-4. CRG Synthesizer Register (SYNR)

Read: Anytime

Write: Anytime except if PLLSEL = 1

NOTE

Write to this register initializes the lock detector bit and the track detector bit.

7.3.2.2 CRG Reference Divider Register (REFDV)

The REFDV register provides a finer granularity for the PLL multiplier steps. The count in the reference divider divides OSCCLK frequency by REFDV + 1.

Module Base +0x_01

	7	6	5	4	3	2	1	0
R	0	0	REFDV5	REFDV4	REFDV3	REFDV2	REFDV1	REFDV0
W	▨	▨						
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 7-5. CRG Reference Divider Register (REFDV)

Read: Anytime

Write: Anytime except when PLLSEL = 1

NOTE

Write to this register initializes the lock detector bit and the track detector bit.

7.3.2.3 Reserved Register (CTFLG)

This register is reserved for factory testing of the CRG module and is not available in normal modes.

Module Base +0x_02

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 7-6. Reserved Register (CTFLG)

Read: Always reads 0x_00 in normal modes

Write: Unimplemented in normal modes

NOTE

Writing to this register when in special mode can alter the CRG fucntionality.

7.3.2.4 CRG Flags Register (CRGFLG)

This register provides CRG status bits and flags.

Module Base +0x_03

	7	6	5	4	3	2	1	0
R	RTIF	PORF	LVRF	LOCKIF	LOCK	TRACK	SCMIF	SCM
W					▨	▨		▨
Reset	0	[1]	[2]	0	0	0	0	0

1. PORF is set to 1 when a power on reset occurs. Unaffected by system reset.
2. LVRF is set to 1 when a low-voltage reset occurs. Unaffected by system reset.

▨ = Unimplemented or Reserved

Figure 7-7. CRG Flags Register (CRGFLG)

Read: Anytime

Write: Refer to each bit for individual write conditions

Table 7-2. CRGFLG Field Descriptions

Field	Description
7 RTIF	**Real Time Interrupt Flag** — RTIF is set to 1 at the end of the RTI period. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (RTIE = 1), RTIF causes an interrupt request. 0 RTI time-out has not yet occurred. 1 RTI time-out has occurred.
6 PORF	**Power on Reset Flag** — PORF is set to 1 when a power on reset occurs. This flag can only be cleared by writing a 1. Writing a 0 has no effect. 0 Power on reset has not occurred. 1 Power on reset has occurred.
5 LVRF	**Low Voltage Reset Flag** — If low voltage reset feature is not available (see device specification) LVRF always reads 0. LVRF is set to 1 when a low voltage reset occurs. This flag can only be cleared by writing a 1. Writing a 0 has no effect. 0 Low voltage reset has not occurred. 1 Low voltage reset has occurred.
4 LOCKIF	**PLL Lock Interrupt Flag** — LOCKIF is set to 1 when LOCK status bit changes. This flag can only be cleared by writing a 1. Writing a 0 has no effect.If enabled (LOCKIE = 1), LOCKIF causes an interrupt request. 0 No change in LOCK bit. 1 LOCK bit has changed.
3 LOCK	**Lock Status Bit** — LOCK reflects the current state of PLL lock condition. This bit is cleared in self clock mode. Writes have no effect. 0 PLL VCO is not within the desired tolerance of the target frequency. 1 PLL VCO is within the desired tolerance of the target frequency.
2 TRACK	**Track Status Bit** — TRACK reflects the current state of PLL track condition. This bit is cleared in self clock mode. Writes have no effect. 0 Acquisition mode status. 1Tracking mode status.
1 SCMIF	**Self Clock Mode Interrupt Flag** — SCMIF is set to 1 when SCM status bit changes. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (SCMIE = 1), SCMIF causes an interrupt request. 0 No change in SCM bit. 1 SCM bit has changed.
0 SCM	**Self Clock Mode Status Bit** — SCM reflects the current clocking mode. Writes have no effect. 0 MCU is operating normally with OSCCLK available. 1 MCU is operating in self clock mode with OSCCLK in an unknown state. All clocks are derived from PLLCLK running at its minimum frequency f_{SCM}.

7.3.2.5 CRG Interrupt Enable Register (CRGINT)

This register enables CRG interrupt requests.

Module Base +0x_04

	7	6	5	4	3	2	1	0
R	RTIE	ILAF	0	LOCKIE	0	0	SCMIE	0
W			▨		▨	▨		▨
Reset	0	[1]	0	0	0	0	0	0

1. ILAF is set to 1 when an illegal address reset occurs. Unaffected by system reset. Cleared by power on or low voltage reset.

▨ = Unimplemented or Reserved

Figure 7-8. CRG Interrupt Enable Register (CRGINT)

Read: Anytime

Write: Anytime

Table 7-3. CRGINT Field Descriptions

Field	Description
7 RTIE	**Real Time Interrupt Enable Bit** 0 Interrupt requests from RTI are disabled. 1 Interrupt will be requested whenever RTIF is set.
6 ILAF	**Illegal Address Reset Flag** — ILAF is set to 1 when an illegal address reset occurs. Refer to S12XMMC Block Guide for details. This flag can only be cleared by writing a 1. Writing a 0 has no effect. 0 Illegal address reset has not occurred. 1 Illegal address reset has occurred.
4 LOCKIE	**Lock Interrupt Enable Bit** 0 LOCK interrupt requests are disabled. 1 Interrupt will be requested whenever LOCKIF is set.
1 SCMIE	**Self Clock Mode Interrupt Enable Bit** 0 SCM interrupt requests are disabled. 1 Interrupt will be requested whenever SCMIF is set.

7.3.2.6 CRG Clock Select Register (CLKSEL)

This register controls CRG clock selection. Refer to Figure 7-17 for more details on the effect of each bit.

Module Base +0x_05

	7	6	5	4	3	2	1	0
R	PLLSEL	PSTP	0	0	PLLWAI	0	RTIWAI	COPWAI
W			▨	▨		▨		
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 7-9. CRG Clock Select Register (CLKSEL)

Read: Anytime

Write: Refer to each bit for individual write conditions

Table 7-4. CLKSEL Field Descriptions

Field	Description
7 PLLSEL	**PLL Select Bit** — Write anytime. Writing a1 when LOCK = 0 and AUTO = 1, or TRACK = 0 and AUTO = 0 has no effect This prevents the selection of an unstable PLLCLK as SYSCLK. PLLSEL bit is cleared when the MCU enters self clock mode, Stop mode or wait mode with PLLWAI bit set. **It is recommended to read back the PLLSEL bit to make sure PLLCLK has really been selected as SYSCLK, as LOCK status bit could theoretically change at the very moment writing the PLLSEL bit.** 0 System clocks are derived from OSCCLK ($f_{BUS} = f_{OSC} / 2$). 1 System clocks are derived from PLLCLK ($f_{BUS} = f_{PLL} / 2$).
6 PSTP	**Pseudo Stop Bit** Write: Anytime This bit controls the functionality of the oscillator during stop mode. 0 Oscillator is disabled in stop mode. 1 Oscillator continues to run in stop mode (pseudo stop). **Note:** Pseudo stop mode allows for faster STOP recovery and reduces the mechanical stress and aging of the resonator in case of frequent STOP conditions at the expense of a slightly increased power consumption.
3 PLLWAI	**PLL Stops in Wait Mode Bit** Write: Anytime If PLLWAI is set, the CRG will clear the PLLSEL bit before entering wait mode. The PLLON bit remains set during wait mode, but the PLL is powered down. Upon exiting wait mode, the PLLSEL bit has to be set manually if PLL clock is required. While the PLLWAI bit is set, the AUTO bit is set to 1 in order to allow the PLL to automatically lock on the selected target frequency after exiting wait mode. 0 PLL keeps running in wait mode. 1 PLL stops in wait mode.
1 RTIWAI	**RTI Stops in Wait Mode Bit** Write: Anytime 0 RTI keeps running in wait mode. 1 RTI stops and initializes the RTI dividers whenever the part goes into wait mode.
0 COPWAI	**COP Stops in Wait Mode Bit** Normal modes: Write once Special modes: Write anytime 0 COP keeps running in wait mode. 1 COP stops and initializes the COP counter whenever the part goes into wait mode.

7.3.2.7 CRG PLL Control Register (PLLCTL)

This register controls the PLL functionality.

Module Base +0x_06

	7	6	5	4	3	2	1	0
R W	CME	PLLON	AUTO	ACQ	FSTWKP	PRE	PCE	SCME
Reset	1	1	1	1	0	0	0	1

Figure 7-10. CRG PLL Control Register (PLLCTL)

Read: Anytime

Write: Refer to each bit for individual write conditions

Table 7-5. PLLCTL Field Descriptions

Field	Description
7 CME	**Clock Monitor Enable Bit** — CME enables the clock monitor. Write anytime except when SCM = 1. 0 Clock monitor is disabled. 1 Clock monitor is enabled. Slow or stopped clocks will cause a clock monitor reset sequence or self clock mode. **Note:** Operating with CME = 0 will not detect any loss of clock. In case of poor clock quality, this could cause unpredictable operation of the MCU! **Note:** In stop mode (PSTP = 0) the clock monitor is disabled independently of the CME bit setting and any loss of external clock will not be detected. Also after wake-up from stop mode (PSTP = 0) with fast wake-up enabled (FSTWKP = 1) the clock monitor is disabled independently of the CME bit setting and any loss of external clock will not be detected.
6 PLLON	**Phase Lock Loop On Bit** — PLLON turns on the PLL circuitry. In self clock mode, the PLL is turned on, but the PLLON bit reads the last latched value. Write anytime except when PLLSEL = 1. 0 PLL is turned off. 1 PLL is turned on. If AUTO bit is set, the PLL will lock automatically.
5 AUTO	**Automatic Bandwidth Control Bit** — AUTO selects either the high bandwidth (acquisition) mode or the low bandwidth (tracking) mode depending on how close to the desired frequency the VCO is running. Write anytime except when PLLWAI = 1, because PLLWAI sets the AUTO bit to 1. 0 Automatic mode control is disabled and the PLL is under software control, using ACQ bit. 1 Automatic mode control is enabled and ACQ bit has no effect.
4 ACQ	**Acquisition Bit** Write anytime. If AUTO=1 this bit has no effect. 0 Low bandwidth filter is selected. 1 High bandwidth filter is selected.
3 FSTWKP	**Fast Wake-up from Full Stop Bit** — FSTWKP enables fast wake-up from full stop mode. Write anytime. If self-clock mode is disabled (SCME = 0) this bit has no effect. 0 Fast wake-up from full stop mode is disabled. 1 Fast wake-up from full stop mode is enabled. When waking up from full stop mode the system will immediately resume operation i self-clock mode (see Section 7.4.1.4, “Clock Quality Checker”). The SCMIF flag will not be set. The system will remain in self-clock mode with oscillator and clock monitor disabled until FSTWKP bit is cleared. The clearing of FSTWKP will start the oscillator, the clock monitor and the clock quality check. If the clock quality check is successful, the CRG will switch all system clocks to OSCCLK. The SCMIF flag will be set. See application examples in Figure 7-23 and Figure 7-24.

Table 7-5. PLLCTL Field Descriptions (continued)

Field	Description
2 PRE	**RTI Enable during Pseudo Stop Bit** — PRE enables the RTI during pseudo stop mode. Write anytime. 0 RTI stops running during pseudo stop mode. 1 RTI continues running during pseudo stop mode. **Note:** If the PRE bit is cleared the RTI dividers will go static while pseudo stop mode is active. The RTI dividers will not initialize like in wait mode with RTIWAI bit set.
1 PCE	**COP Enable during Pseudo Stop Bit** — PCE enables the COP during pseudo stop mode. Write anytime. 0 COP stops running during pseudo stop mode 1 COP continues running during pseudo stop mode **Note:** If the PCE bit is cleared, the COP dividers will go static while pseudo stop mode is active. The COP dividers will not initialize like in wait mode with COPWAI bit set.
0 SCME	**Self Clock Mode Enable Bit** Normal modes: Write once Special modes: Write anytime SCME can not be cleared while operating in self clock mode (SCM = 1). 0 Detection of crystal clock failure causes clock monitor reset (see Section 7.5.2, “Clock Monitor Reset”). 1 Detection of crystal clock failure forces the MCU in self clock mode (see Section 7.4.2.2, “Self Clock Mode”).

7.3.2.8 CRG RTI Control Register (RTICTL)

This register selects the timeout period for the real time interrupt.

Module Base +0x_07

	7	6	5	4	3	2	1	0
R W	RTDEC	RTR6	RTR5	RTR4	RTR3	RTR2	RTR1	RTR0
Reset	0	0	0	0	0	0	0	0

Figure 7-11. CRG RTI Control Register (RTICTL)

Read: Anytime

Write: Anytime

NOTE

A write to this register initializes the RTI counter.

Table 7-6. RTICTL Field Descriptions

Field	Description
7 RTDEC	**Decimal or Binary Divider Select Bit** — RTDEC selects decimal or binary based prescaler values. 0 Binary based divider value. See Table 7-7 1 Decimal based divider value. See Table 7-8
6–4 RTR[6:4]	**Real Time Interrupt Prescale Rate Select Bits** — These bits select the prescale rate for the RTI. See Table 7-7 and Table 7-8.
3–0 RTR[3:0]	**Real Time Interrupt Modulus Counter Select Bits** — These bits select the modulus counter target value to provide additional granularity.Table 7-7 and Table 7-8 show all possible divide values selectable by the RTICTL register. The source clock for the RTI is OSCCLK.

Table 7-7. RTI Frequency Divide Rates for RTDEC = 0

RTR[3:0]	RTR[6:4] =							
	000 (OFF)	**001 (2^{10})**	**010 (2^{11})**	**011 (2^{12})**	**100 (2^{13})**	**101 (2^{14})**	**110 (2^{15})**	**111 (2^{16})**
0000 (÷1)	OFF*	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
0001 (÷2)	OFF	$2x2^{10}$	$2x2^{11}$	$2x2^{12}$	$2x2^{13}$	$2x2^{14}$	$2x2^{15}$	$2x2^{16}$
0010 (÷3)	OFF	$3x2^{10}$	$3x2^{11}$	$3x2^{12}$	$3x2^{13}$	$3x2^{14}$	$3x2^{15}$	$3x2^{16}$
0011 (÷4)	OFF	$4x2^{10}$	$4x2^{11}$	$4x2^{12}$	$4x2^{13}$	$4x2^{14}$	$4x2^{15}$	$4x2^{16}$
0100 (÷5)	OFF	$5x2^{10}$	$5x2^{11}$	$5x2^{12}$	$5x2^{13}$	$5x2^{14}$	$5x2^{15}$	$5x2^{16}$
0101 (÷6)	OFF	$6x2^{10}$	$6x2^{11}$	$6x2^{12}$	$6x2^{13}$	$6x2^{14}$	$6x2^{15}$	$6x2^{16}$
0110 (÷7)	OFF	$7x2^{10}$	$7x2^{11}$	$7x2^{12}$	$7x2^{13}$	$7x2^{14}$	$7x2^{15}$	$7x2^{16}$
0111 (÷8)	OFF	$8x2^{10}$	$8x2^{11}$	$8x2^{12}$	$8x2^{13}$	$8x2^{14}$	$8x2^{15}$	$8x2^{16}$
1000 (÷9)	OFF	$9x2^{10}$	$9x2^{11}$	$9x2^{12}$	$9x2^{13}$	$9x2^{14}$	$9x2^{15}$	$9x2^{16}$
1001 (÷10)	OFF	$10x2^{10}$	$10x2^{11}$	$10x2^{12}$	$10x2^{13}$	$10x2^{14}$	$10x2^{15}$	$10x2^{16}$
1010 (÷11)	OFF	$11x2^{10}$	$11x2^{11}$	$11x2^{12}$	$11x2^{13}$	$11x2^{14}$	$11x2^{15}$	$11x2^{16}$
1011 (÷12)	OFF	$12x2^{10}$	$12x2^{11}$	$12x2^{12}$	$12x2^{13}$	$12x2^{14}$	$12x2^{15}$	$12x2^{16}$
1100 (÷13)	OFF	$13x2^{10}$	$13x2^{11}$	$13x2^{12}$	$13x2^{13}$	$13x2^{14}$	$13x2^{15}$	$13x2^{16}$
1101 (÷14)	OFF	$14x2^{10}$	$14x2^{11}$	$14x2^{12}$	$14x2^{13}$	$14x2^{14}$	$14x2^{15}$	$14x2^{16}$
1110 (÷15)	OFF	$15x2^{10}$	$15x2^{11}$	$15x2^{12}$	$15x2^{13}$	$15x2^{14}$	$15x2^{15}$	$15x2^{16}$
1111 (÷16)	OFF	$16x2^{10}$	$16x2^{11}$	$16x2^{12}$	$16x2^{13}$	$16x2^{14}$	$16x2^{15}$	$16x2^{16}$

* Denotes the default value out of reset.This value should be used to disable the RTI to ensure future backwards compatibility.

Table 7-8. RTI Frequency Divide Rates for RTDEC = 1

RTR[3:0]	RTR[6:4] =							
	000 ($1x10^3$)	**001 ($2x10^3$)**	**010 ($5x10^3$)**	**011 ($10x10^3$)**	**100 ($20x10^3$)**	**101 ($50x10^3$)**	**110 ($100x10^3$)**	**111 ($200x10^3$)**
0000 (÷1)	$1x10^3$	$2x10^3$	$5x10^3$	$10x10^3$	$20x10^3$	$50x10^3$	$100x10^3$	$200x10^3$
0001 (÷2)	$2x10^3$	$4x10^3$	$10x10^3$	$20x10^3$	$40x10^3$	$100x10^3$	$200x10^3$	$400x10^3$
0010 (÷3)	$3x10^3$	$6x10^3$	$15x10^3$	$30x10^3$	$60x10^3$	$150x10^3$	$300x10^3$	$600x10^3$
0011 (÷4)	$4x10^3$	$8x10^3$	$20x10^3$	$40x10^3$	$80x10^3$	$200x10^3$	$400x10^3$	$800x10^3$
0100 (÷5)	$5x10^3$	$10x10^3$	$25x10^3$	$50x10^3$	$100x10^3$	$250x10^3$	$500x10^3$	$1x10^6$
0101 (÷6)	$6x10^3$	$12x10^3$	$30x10^3$	$60x10^3$	$120x10^3$	$300x10^3$	$600x10^3$	$1.2x10^6$
0110 (÷7)	$7x10^3$	$14x10^3$	$35x10^3$	$70x10^3$	$140x10^3$	$350x10^3$	$700x10^3$	$1.4x10^6$
0111 (÷8)	$8x10^3$	$16x10^3$	$40x10^3$	$80x10^3$	$160x10^3$	$400x10^3$	$800x10^3$	$1.6x10^6$
1000 (÷9)	$9x10^3$	$18x10^3$	$45x10^3$	$90x10^3$	$180x10^3$	$450x10^3$	$900x10^3$	$1.8x10^6$
1001 (÷10)	$10 x10^3$	$20x10^3$	$50x10^3$	$100x10^3$	$200x10^3$	$500x10^3$	$1x10^6$	$2x10^6$
1010 (÷11)	$11 x10^3$	$22x10^3$	$55x10^3$	$110x10^3$	$220x10^3$	$550x10^3$	$1.1x10^6$	$2.2x10^6$
1011 (÷12)	$12x10^3$	$24x10^3$	$60x10^3$	$120x10^3$	$240x10^3$	$600x10^3$	$1.2x10^6$	$2.4x10^6$
1100 (÷13)	$13x10^3$	$26x10^3$	$65x10^3$	$130x10^3$	$260x10^3$	$650x10^3$	$1.3x10^6$	$2.6x10^6$
1101 (÷14)	$14x10^3$	$28x10^3$	$70x10^3$	$140x10^3$	$280x10^3$	$700x10^3$	$1.4x10^6$	$2.8x10^6$
1110 (÷15)	$15x10^3$	$30x10^3$	$75x10^3$	$150x10^3$	$300x10^3$	$750x10^3$	$1.5x10^6$	$3x10^6$
1111 (÷16)	$16x10^3$	$32x10^3$	$80x10^3$	$160x10^3$	$320x10^3$	$800x10^3$	$1.6x10^6$	$3.2x10^6$

7.3.2.9 CRG COP Control Register (COPCTL)

This register controls the COP (computer operating properly) watchdog.

Module Base +0x_08

	7	6	5	4	3	2	1	0
R	WCOP	RSBCK	0	0	0	CR2	CR1	CR0
W			WRTMASK	▨	▨			
Reset[1]		0	0	0	0			

[1] Refer to Device User Guide (Section: CRG) for reset values of WCOP, CR2, CR1, and CR0.

▨ = Unimplemented or Reserved

Figure 7-12. CRG COP Control Register (COPCTL)

Read: Anytime

Write:

1. RSBCK: Anytime in special modes; write to “1” but not to “0” in all other modes
2. WCOP, CR2, CR1, CR0:
   - — Anytime in special modes
   - — Write once in all other modes
     Writing CR[2:0] to “000” has no effect, but counts for the “write once” condition.
     Writing WCOP to “0” has no effect, but counts for the “write once” condition.

The COP time-out period is restarted if one these two conditions is true:

1. Writing a nonzero value to CR[2:0] (anytime in special modes, once in all other modes) with WRTMASK = 0.

   or

2. Changing RSBCK bit from “0” to “1”.

Table 7-9. COPCTL Field Descriptions

Field	Description
7 WCOP	**Window COP Mode Bit** — When set, a write to the ARMCOP register must occur in the last 25% of the selected period. A write during the first 75% of the selected period will reset the part. As long as all writes occur during this window, 0x_55 can be written as often as desired. Once 0x_AA is written after the 0x_55, the time-out logic restarts and the user must wait until the next window before writing to ARMCOP. Table 7-10 shows the duration of this window for the seven available COP rates. 0 Normal COP operation 1 Window COP operation
6 RSBCK	**COP and RTI Stop in Active BDM Mode Bit** 0 Allows the COP and RTI to keep running in active BDM mode. 1 Stops the COP and RTI counters whenever the part is in active BDM mode.

Table 7-9. COPCTL Field Descriptions (continued)

Field	Description
5 WRTMASK	**Write Mask for WCOP and CR[2:0] Bit** — This write-only bit serves as a mask for the WCOP and CR[2:0] bits while writing the COPCTL register. It is intended for BDM writing the RSBCK without touching the contents of WCOP and CR[2:0]. 0 Write of WCOP and CR[2:0] has an effect with this write of COPCTL 1 Write of WCOP and CR[2:0] has no effect with this write of COPCTL. (Does not count for “write once”.)
2–0 CR[1:0]	**COP Watchdog Timer Rate Select** — These bits select the COP time-out rate (see Table 7-10). The COP time-out period is OSCCLK period divided by CR[2:0] value. Writing a nonzero value to CR[2:0] enables the COP counter and starts the time-out period. A COP counter time-out causes a system reset. This can be avoided by periodically (before time-out) reinitializing the COP counter via the ARMCOP register. While all of the following three conditions are true the CR[2:0], WCOP bits are ignored and the COP operates at highest time-out period (2^{24} cycles) in normal COP mode (Window COP mode disabled): 1) COP is enabled (CR[2:0] is not 000) 2) BDM mode active 3) RSBCK = 0 4) Operation in emulation or special modes

Table 7-10. COP Watchdog Rates[1]

CR2	CR1	CR0	OSCCLK Cycles to Time-out
0	0	0	COP disabled
0	0	1	2^{14}
0	1	0	2^{16}
0	1	1	2^{18}
1	0	0	2^{20}
1	0	1	2^{22}
1	1	0	2^{23}
1	1	1	2^{24}

[1] OSCCLK cycles are referenced from the previous COP time-out reset (writing 0x_55/0x_AA to the ARMCOP register)

7.3.2.10 Reserved Register (FORBYP)

NOTE

This reserved register is designed for factory test purposes only, and is not intended for general user access. Writing to this register when in special modes can alter the CRG’s functionality.

Module Base +0x_09

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 7-13. Reserved Register (FORBYP)

Read: Always read 0x_00 except in special modes

Write: Only in special modes

7.3.2.11 Reserved Register (CTCTL)

NOTE

This reserved register is designed for factory test purposes only, and is not intended for general user access. Writing to this register when in special test modes can alter the CRG’s functionality.

Module Base +0x_0A

	7	6	5	4	3	2	1	0
R	1	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 7-14. Reserved Register (CTCTL)

Read: always read 0x_80 except in special modes

Write: only in special modes

7.3.2.12 CRG COP Timer Arm/Reset Register (ARMCOP)

This register is used to restart the COP time-out period.

Module Base +0x_0B

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 7-15. ARMCOP Register Diagram

Read: Always reads 0x_00

Write: Anytime

When the COP is disabled (CR[2:0] = “000”) writing to this register has no effect.

When the COP is enabled by setting CR[2:0] nonzero, the following applies:

> Writing any value other than 0x_55 or 0x_AA causes a COP reset. To restart the COP time-out period you must write 0x_55 followed by a write of 0x_AA. Other instructions may be executed between these writes but the sequence (0x_55, 0x_AA) must be completed prior to COP end of time-out period to avoid a COP reset. Sequences of 0x_55 writes or sequences of 0x_AA writes are allowed. When the WCOP bit is set, 0x_55 and 0x_AA writes must be done in the last 25% of the selected time-out period; writing any value in the first 75% of the selected period will cause a COP reset.

7.4 Functional Description

7.4.1 Functional Blocks

7.4.1.1 Phase Locked Loop (PLL)

The PLL is used to run the MCU from a different time base than the incoming OSCCLK. For increased flexibility, OSCCLK can be divided in a range of 1 to 16 to generate the reference frequency. This offers a finer multiplication granularity. The PLL can multiply this reference clock by a multiple of 2, 4, 6,... 126,128 based on the SYNR register.

$$\mathrm{PLLCLK} = 2 \times \mathrm{OSCCLK} \times \frac{[\mathrm{SYNR}+1]}{[\mathrm{REFDV}+1]}$$

CAUTION

> Although it is possible to set the two dividers to command a very high clock frequency, do not exceed the specified bus frequency limit for the MCU.
> If (PLLSEL = 1), Bus Clock = PLLCLK / 2

The PLL is a frequency generator that operates in either acquisition mode or tracking mode, depending on the difference between the output frequency and the target frequency. The PLL can change between acquisition and tracking modes either automatically or manually.

The VCO has a minimum operating frequency, which corresponds to the self clock mode frequency f_{SCM}.

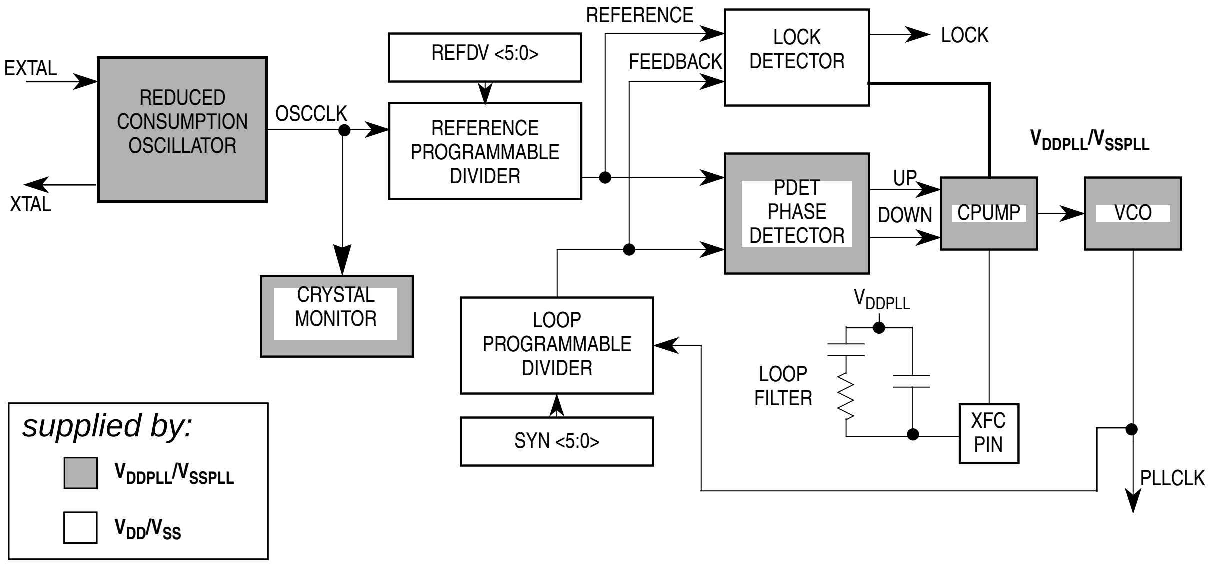


Figure 7-16. PLL Functional Diagram

7.4.1.1.1 PLL Operation

The oscillator output clock signal (OSCCLK) is fed through the reference programmable divider and is divided in a range of 1 to 64 (REFDV + 1) to output the REFERENCE clock. The VCO output clock, (PLLCLK) is fed back through the programmable loop divider and is divided in a range of 2 to 128 in increments of [2 x (SYNR + 1)] to output the FEEDBACK clock. Figure 7-16.

The phase detector then compares the FEEDBACK clock, with the REFERENCE clock. Correction pulses are generated based on the phase difference between the two signals. The loop filter then slightly alters the DC voltage on the external filter capacitor connected to XFC pin, based on the width and direction of the correction pulse. The filter can make fast or slow corrections depending on its mode, as described in the next subsection. The values of the external filter network and the reference frequency determine the speed of the corrections and the stability of the PLL.

The minimum VCO frequency is reached with the XFC pin forced to V_{DDPLL}. This is the self clock mode frequency.

7.4.1.1.2 Acquisition and Tracking Modes

The lock detector compares the frequencies of the FEEDBACK clock, and the REFERENCE clock. Therefore, the speed of the lock detector is directly proportional to the final reference frequency. The circuit determines the mode of the PLL and the lock condition based on this comparison.

The PLL filter can be manually or automatically configured into one of two possible operating modes:

- Acquisition mode

  In acquisition mode, the filter can make large frequency corrections to the VCO. This mode is used at PLL start-up or when the PLL has suffered a severe noise hit and the VCO frequency is far off the desired frequency. When in acquisition mode, the TRACK status bit is cleared in the CRGFLG register.
- Tracking mode

  In tracking mode, the filter makes only small corrections to the frequency of the VCO. PLL jitter is much lower in tracking mode, but the response to noise is also slower. The PLL enters tracking mode when the VCO frequency is nearly correct and the TRACK bit is set in the CRGFLG register.

The PLL can change the bandwidth or operational mode of the loop filter manually or automatically.

In automatic bandwidth control mode (AUTO = 1), the lock detector automatically switches between acquisition and tracking modes. Automatic bandwidth control mode also is used to determine when the PLL clock (PLLCLK) is safe to use as the source for the system and core clocks. If PLL LOCK interrupt requests are enabled, the software can wait for an interrupt request and then check the LOCK bit. If interrupt requests are disabled, software can poll the LOCK bit continuously (during PLL start-up, usually) or at periodic intervals. In either case, only when the LOCK bit is set, is the PLLCLK clock safe to use as the source for the system and core clocks. If the PLL is selected as the source for the system and core clocks and the LOCK bit is clear, the PLL has suffered a severe noise hit and the software must take appropriate action, depending on the application.

The following conditions apply when the PLL is in automatic bandwidth control mode (AUTO = 1):

- The TRACK bit is a read-only indicator of the mode of the filter.
- The TRACK bit is set when the VCO frequency is within a certain tolerance, Δ_{trk}, and is clear when the VCO frequency is out of a certain tolerance, Δ_{unt}.
- The LOCK bit is a read-only indicator of the locked state of the PLL.
- The LOCK bit is set when the VCO frequency is within a certain tolerance, Δ_{Lock}, and is cleared when the VCO frequency is out of a certain tolerance, Δ_{unl}.
- Interrupt requests can occur if enabled (LOCKIE = 1) when the lock condition changes, toggling the LOCK bit.

The PLL can also operate in manual mode (AUTO = 0). Manual mode is used by systems that do not require an indicator of the lock condition for proper operation. Such systems typically operate well below the maximum system frequency (f_{sys}) and require fast start-up. The following conditions apply when in manual mode:

- ACQ is a writable control bit that controls the mode of the filter. Before turning on the PLL in manual mode, the ACQ bit should be asserted to configure the filter in acquisition mode.
- After turning on the PLL by setting the PLLON bit software must wait a given time (t_{acq}) before entering tracking mode (ACQ = 0).
- After entering tracking mode software must wait a given time (t_{al}) before selecting the PLLCLK as the source for system and core clocks (PLLSEL = 1).

7.4.1.2 System Clocks Generator

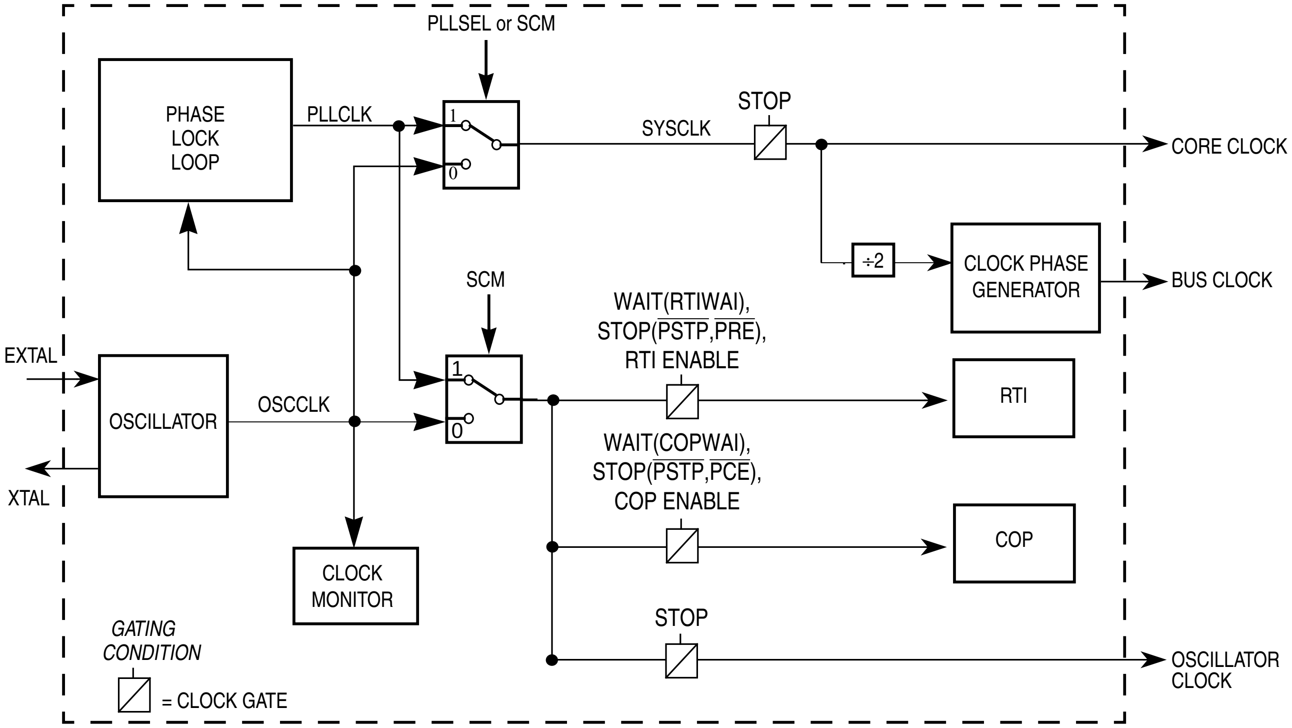


Figure 7-17. System Clocks Generator

The clock generator creates the clocks used in the MCU (see Figure 7-17). The gating condition placed on top of the individual clock gates indicates the dependencies of different modes (STOP, WAIT) and the setting of the respective configuration bits.

The peripheral modules use the bus clock. Some peripheral modules also use the oscillator clock. The memory blocks use the bus clock. If the MCU enters self clock mode (see Section 7.4.2.2, “Self Clock Mode”) oscillator clock source is switched to PLLCLK running at its minimum frequency f_{SCM}. The bus clock is used to generate the clock visible at the ECLK pin. The core clock signal is the clock for the CPU. The core clock is twice the bus clock as shown in Figure 7-18. But note that a CPU cycle corresponds to one bus clock.

PLL clock mode is selected with PLLSEL bit in the CLKSEL registerr. When selected, the PLL output clock drives SYSCLK for the main system including the CPU and peripherals. The PLL cannot be turned off by clearing the PLLON bit, if the PLL clock is selected. When PLLSEL is changed, it takes a maximum of 4 OSCCLK plus 4 PLLCLK cycles to make the transition. During the transition, all clocks freeze and CPU activity ceases.

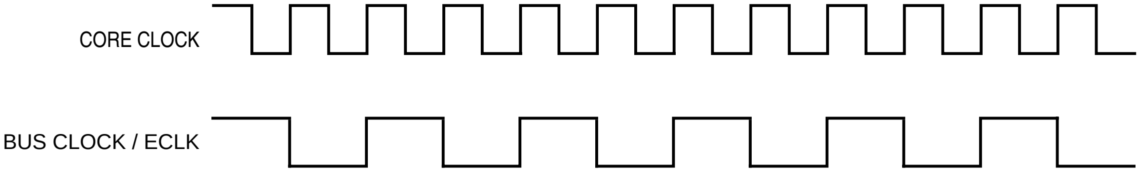


Figure 7-18. Core Clock and Bus Clock Relationship

7.4.1.3 Clock Monitor (CM)

If no OSCCLK edges are detected within a certain time, the clock monitor within the oscillator block generates a clock monitor fail event. The CRG then asserts self clock mode or generates a system reset depending on the state of SCME bit. If the clock monitor is disabled or the presence of clocks is detected no failure is indicated by the oscillator block.The clock monitor function is enabled/disabled by the CME control bit.

7.4.1.4 Clock Quality Checker

The clock monitor performs a coarse check on the incoming clock signal. The clock quality checker provides a more accurate check in addition to the clock monitor.

A clock quality check is triggered by any of the following events:

- Power on reset (*POR*)
- Low voltage reset (*LVR*)
- Wake-up from full stop mode (*exit full stop*)
- Clock monitor fail indication (*CM fail*)

A time window of 50,000 VCO clock cycles[1] is called *check window*.

---

[1] VCO clock cycles are generated by the PLL when running at minimum frequency f_{SCM}.

A number greater equal than 4096 rising OSCCLK edges within a *check window* is called *osc ok*. Note that *osc ok* immediately terminates the current *check window*. See Figure 7-19 as an example.

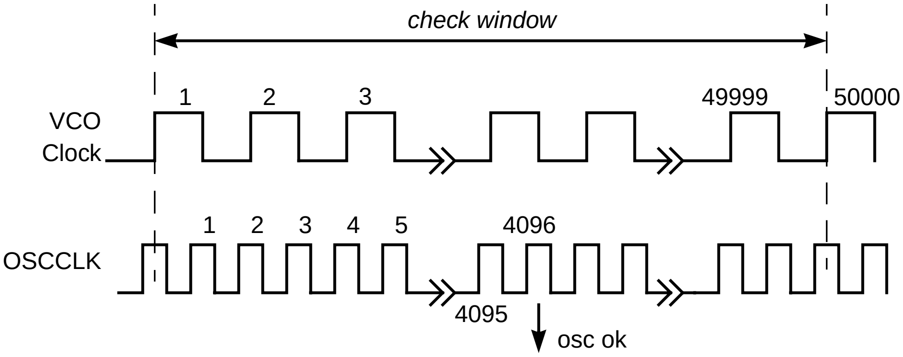


Figure 7-19. Check Window Example

The sequence for clock quality check is shown in Figure 7-20.

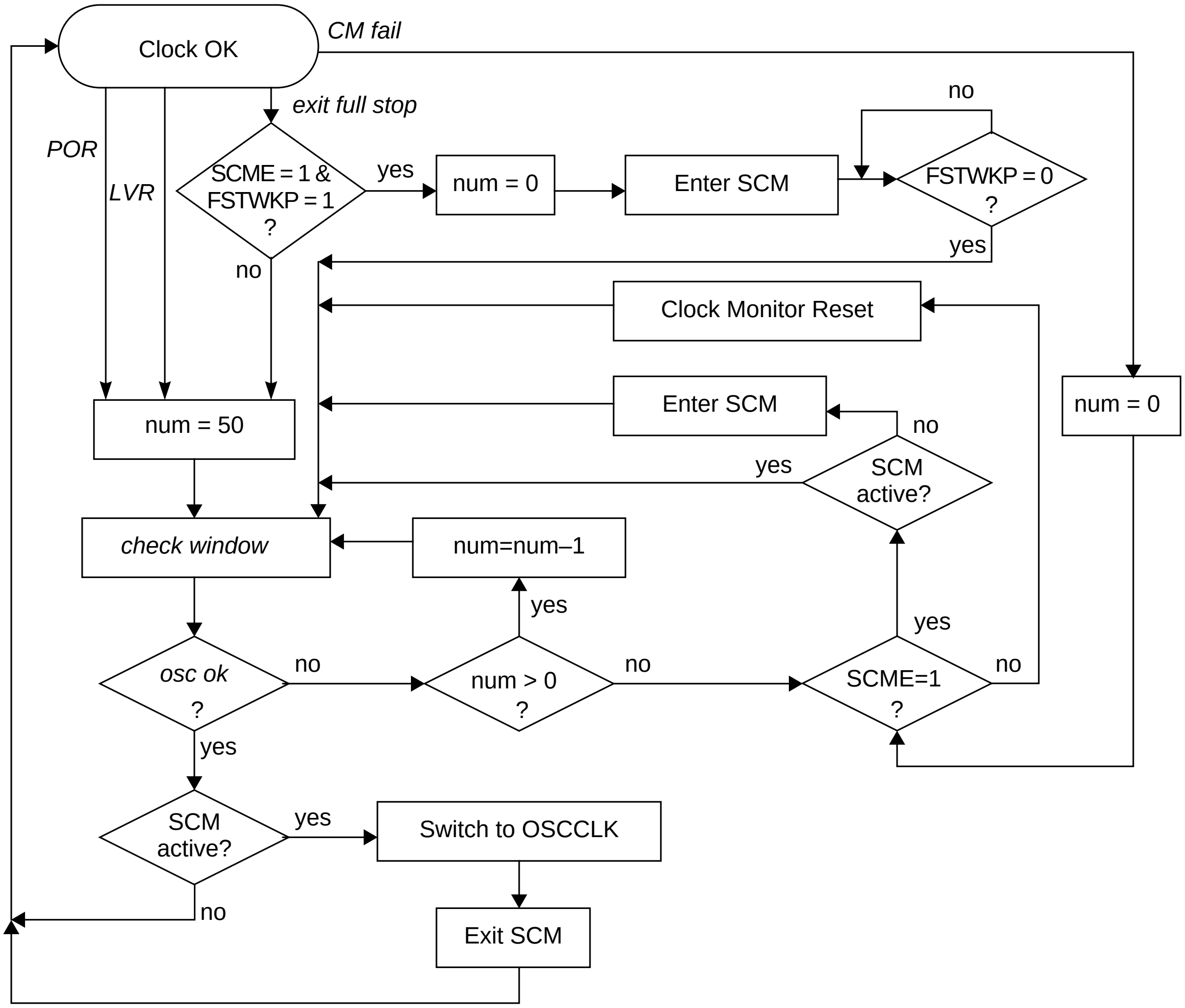


Figure 7-20. Sequence for Clock Quality Check

NOTE

Remember that in parallel to additional actions caused by self clock mode or clock monitor reset[1] handling the clock quality checker continues to check the OSCCLK signal.

The clock quality checker enables the PLL and the voltage regulator (VREG) anytime a clock check has to be performed. An ongoing clock quality check could also cause a running PLL (f_{SCM}) and an active VREG during pseudo stop mode or wait mode.

7.4.1.5 Computer Operating Properly Watchdog (COP)

The COP (free running watchdog timer) enables the user to check that a program is running and sequencing properly. When the COP is being used, software is responsible for keeping the COP from timing out. If the COP times out it is an indication that the software is no longer being executed in the intended sequence; thus a system reset is initiated (see Section 7.4.1.5, "Computer Operating Properly Watchdog (COP)"). The COP runs with a gated OSCCLK. Three control bits in the COPCTL register allow selection of seven COP time-out periods.

When COP is enabled, the program must write 0x_55 and 0x_AA (in this order) to the ARMCOP register during the selected time-out period. Once this is done, the COP time-out period is restarted. If the program fails to do this and the COP times out, the part will reset. Also, if any value other than 0x_55 or 0x_AA is written, the part is immediately reset.

Windowed COP operation is enabled by setting WCOP in the COPCTL register. In this mode, writes to the ARMCOP register to clear the COP timer must occur in the last 25% of the selected time-out period. A premature write will immediately reset the part.

If PCE bit is set, the COP will continue to run in pseudo stop mode.

7.4.1.6 Real Time Interrupt (RTI)

The RTI can be used to generate a hardware interrupt at a fixed periodic rate. If enabled (by setting RTIE = 1), this interrupt will occur at the rate selected by the RTICTL register. The RTI runs with a gated OSCCLK. At the end of the RTI time-out period the RTIF flag is set to 1 and a new RTI time-out period starts immediately.

A write to the RTICTL register restarts the RTI time-out period.

If the PRE bit is set, the RTI will continue to run in pseudo stop mode.

7.4.2 Operating Modes

7.4.2.1 Normal Mode

The CRG block behaves as described within this specification in all normal modes.

---

[1] A Clock Monitor Reset will always set the SCME bit to logical 1.

7.4.2.2 Self Clock Mode

The VCO has a minimum operating frequency, f_{SCM}. If the external clock frequency is not available due to a failure or due to long crystal start-up time, the bus clock and the core clock are derived from the VCO running at minimum operating frequency; this mode of operation is called self clock mode. This requires CME = 1 and SCME = 1. If the MCU was clocked by the PLL clock prior to entering self clock mode, the PLLSEL bit will be cleared. If the external clock signal has stabilized again, the CRG will automatically select OSCCLK to be the system clock and return to normal mode. Section 7.4.1.4, "Clock Quality Checker" for more information on entering and leaving self clock mode.

NOTE

In order to detect a potential clock loss the CME bit should be always enabled (CME = 1)!

If CME bit is disabled and the MCU is configured to run on PLL clock (PLLCLK), a loss of external clock (OSCCLK) will not be detected and will cause the system clock to drift towards the VCO's minimum frequency f_{SCM}. As soon as the external clock is available again the system clock ramps up to its PLL target frequency. If the MCU is running on external clock any loss of clock will cause the system to go static.

7.4.3 Low Power Options

This section summarizes the low power options available in the CRG.

7.4.3.1 Run Mode

The RTI can be stopped by setting the associated rate select bits to 0.

The COP can be stopped by setting the associated rate select bits to 0.

7.4.3.2 Wait Mode

The WAI instruction puts the MCU in a low power consumption stand-by mode depending on setting of the individual bits in the CLKSEL register. All individual wait mode configuration bits can be superposed. This provides enhanced granularity in reducing the level of power consumption during wait mode. Table 7-11 lists the individual configuration bits and the parts of the MCU that are affected in wait mode

Table 7-11. MCU Configuration During Wait Mode

	PLLWAI	RTIWAI	COPWAI
PLL	Stopped	—	—
RTI	—	Stopped	—
COP	—	—	Stopped

After executing the WAI instruction the core requests the CRG to switch MCU into wait mode. The CRG then checks whether the PLLWAI bit is asserted (Figure 7-21). Depending on the configuration, the CRG switches the system and core clocks to OSCCLK by clearing the PLLSEL bit and disables the PLL. As soon as all clocks are switched off wait mode is active.

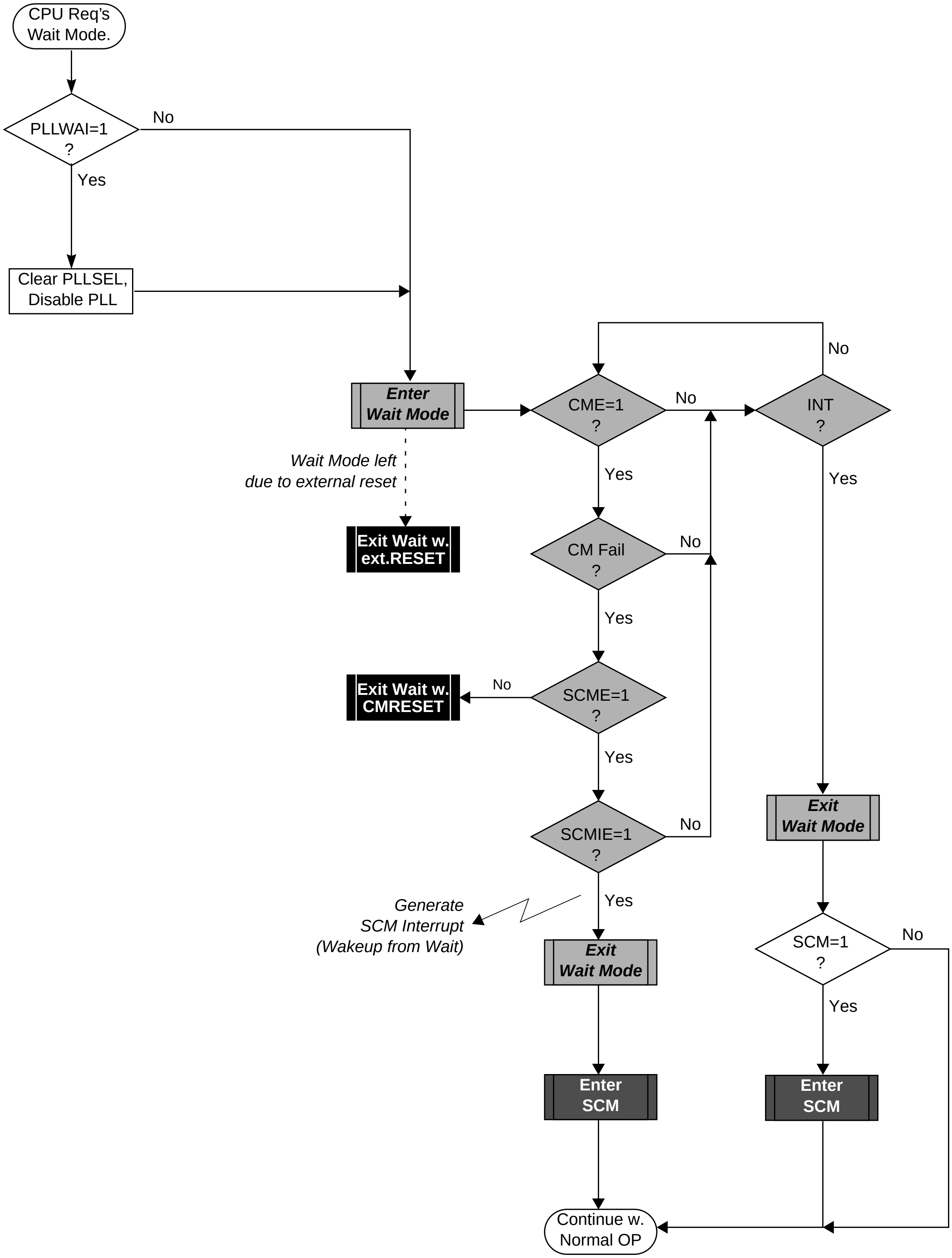


Figure 7-21. Wait Mode Entry/Exit Sequence

There are four different scenarios for the CRG to restart the MCU from wait mode:

- External reset
- Clock monitor reset
- COP reset
- Any interrupt

If the MCU gets an external reset or COP reset during wait mode active, the CRG asynchronously restores all configuration bits in the register space to its default settings and starts the reset generator. After completing the reset sequence processing begins by fetching the normal or COP reset vector. Wait mode is left and the MCU is in run mode again.

If the clock monitor is enabled (CME = 1) the MCU is able to leave wait mode when loss of oscillator/external clock is detected by a clock monitor fail. If the SCME bit is not asserted the CRG generates a clock monitor fail reset (CMRESET). The CRG’s behavior for CMRESET is the same compared to external reset, but another reset vector is fetched after completion of the reset sequence. If the SCME bit is asserted the CRG generates a SCM interrupt if enabled (SCMIE = 1). After generating the interrupt the CRG enters self-clock mode and starts the clock quality checker (Section 7.4.1.4, “Clock Quality Checker”). Then the MCU continues with normal operation.If the SCM interrupt is blocked by SCMIE = 0, the SCMIF flag will be asserted and clock quality checks will be performed but the MCU will not wake-up from wait-mode.

If any other interrupt source (e.g., RTI) triggers exit from wait mode, the MCU immediately continues with normal operation. If the PLL has been powered-down during wait mode, the PLLSEL bit is cleared and the MCU runs on OSCCLK after leaving wait mode. The software must manually set the PLLSEL bit again, in order to switch system and core clocks to the PLLCLK.

If wait mode is entered from self-clock mode the CRG will continue to check the clock quality until clock check is successful. The PLL and voltage regulator (VREG) will remain enabled.

Table 7-12 summarizes the outcome of a clock loss while in wait mode.

7.4.3.3 System Stop Mode

All clocks are stopped in STOP mode, dependent of the setting of the PCE, PRE, and PSTP bit. The oscillator is disabled in STOP mode unless the PSTP bit is set. All counters and dividers remain frozen but do not initialize. If the PRE or PCE bits are set, the RTI or COP continues to run in pseudo stop mode. In addition to disabling system and core clocks the CRG requests other functional units of the MCU (e.g., voltage-regulator) to enter their individual power saving modes (if available). This is the main difference between pseudo stop mode and wait mode.

If the PLLSEL bit is still set when entering stop mode, the CRG will switch the system and core clocks to OSCCLK by clearing the PLLSEL bit. Then the CRG disables the PLL, disables the core clock and finally disables the remaining system clocks. As soon as all clocks are switched off, stop mode is active.

If pseudo stop mode (PSTP = 1) is entered from self-clock mode, the CRG will continue to check the clock quality until clock check is successful. The PLL and the voltage regulator (VREG) will remain enabled. If full stop mode (PSTP = 0) is entered from self-clock mode, an ongoing clock quality check will be stopped. A complete timeout window check will be started when stop mode is left again.

Wake-up from stop mode also depends on the setting of the PSTP bit.

Table 7-12. Outcome of Clock Loss in Wait Mode

CME	SCME	SCMIE	CRG Actions
0	X	X	Clock failure --> No action, clock loss not detected.
1	0	X	Clock failure --> CRG performs Clock Monitor Reset immediately
1	1	0	Clock failure --> Scenario 1: OSCCLK recovers prior to exiting wait mode. – MCU remains in wait mode, – VREG enabled, – PLL enabled, – SCM activated, – Start clock quality check, – Set SCMIF interrupt flag. *Some time later OSCCLK recovers.* – CM no longer indicates a failure, – 4096 OSCCLK cycles later clock quality check indicates clock o.k., – SCM deactivated, – PLL disabled depending on PLLWAI, – VREG remains enabled *(never gets disabled in wait mode)*. – MCU remains in wait mode. *Some time later either a wakeup interrupt occurs (no SCM interrupt)* – Exit wait mode using OSCCLK as system clock (SYSCLK), – Continue normal operation. *or an External Reset is applied.* – Exit wait mode using OSCCLK as system clock, – Start reset sequence. Scenario 2: OSCCLK does not recover prior to exiting wait mode. – MCU remains in wait mode, – VREG enabled, – PLL enabled, – SCM activated, – Start clock quality check, – Set SCMIF interrupt flag, – Keep performing clock quality checks (could continue infinitely) while in wait mode. *Some time later either a wakeup interrupt occurs (no SCM interrupt)* – Exit wait mode in SCM using PLL clock (f_{SCM}) as system clock, – Continue to perform additional clock quality checks until OSCCLK is o.k. again. or an External RESET is applied. – Exit wait mode in SCM using PLL clock (f_{SCM}) as system clock, – Start reset sequence, – Continue to perform additional clock quality checks until OSCCLK is o.k.again.
1	1	1	Clock failure --> – VREG enabled, – PLL enabled, – SCM activated, – Start clock quality check, – SCMIF set. SCMIF generates self clock mode wakeup interrupt. – Exit wait mode in SCM using PLL clock (f_{SCM}) as system clock, – Continue to perform a additional clock quality checks until OSCCLK is o.k. again.

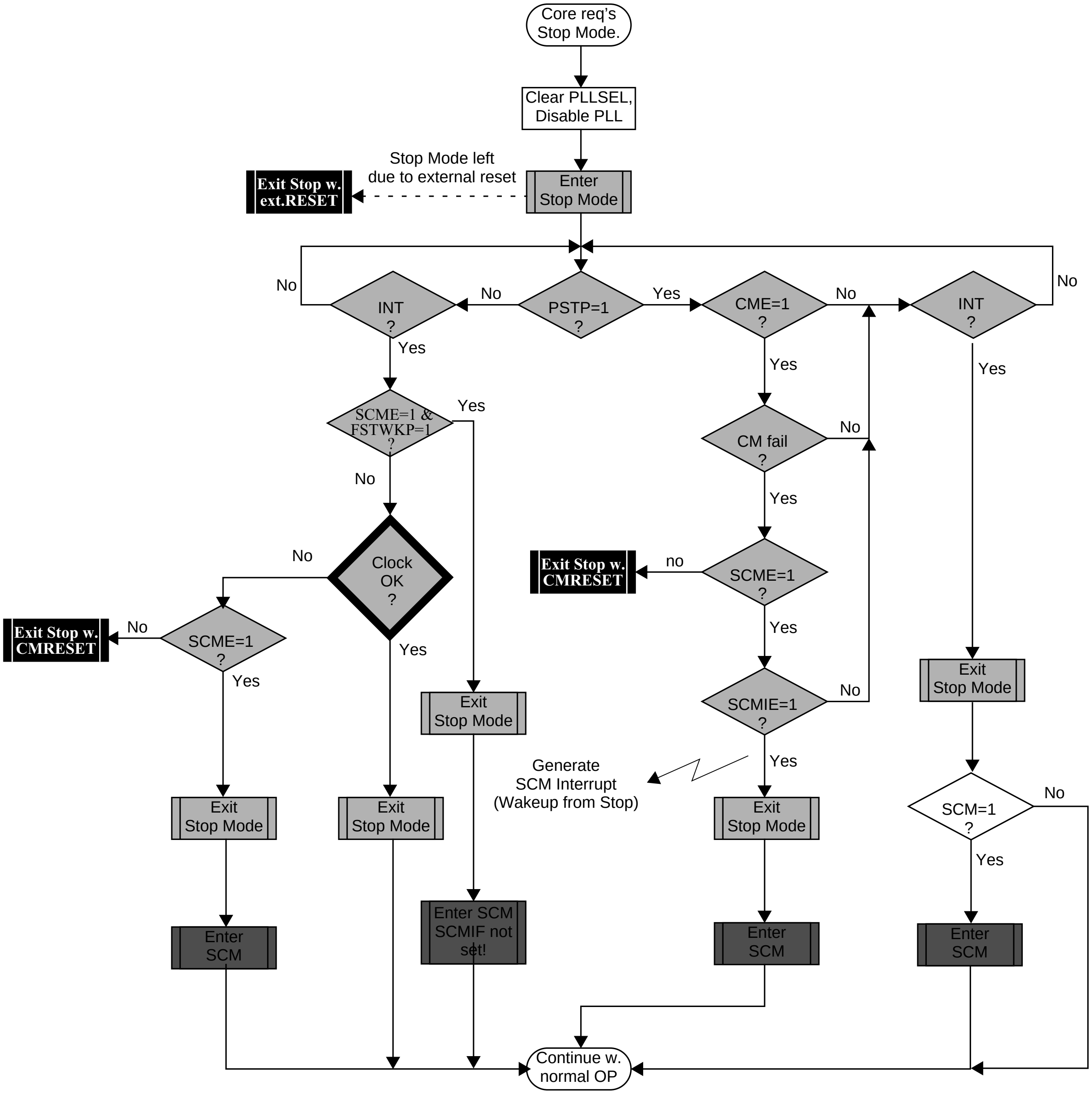


Figure 7-22. Stop Mode Entry/Exit Sequence

7.4.3.3.1 Wake-up from Pseudo Stop Mode (PSTP=1)

Wake-up from pseudo stop mode is the same as wake-up from wait mode. There are also four different scenarios for the CRG to restart the MCU from pseudo stop mode:

- External reset
- Clock monitor fail
- COP reset
- Wake-up interrupt

If the MCU gets an external reset or COP reset during pseudo stop mode active, the CRG asynchronously restores all configuration bits in the register space to its default settings and starts the reset generator. After completing the reset sequence processing begins by fetching the normal or COP reset vector. pseudo stop mode is left and the MCU is in run mode again.

If the clock monitor is enabled (CME = 1), the MCU is able to leave pseudo stop mode when loss of oscillator/external clock is detected by a clock monitor fail. If the SCME bit is not asserted the CRG generates a clock monitor fail reset (CMRESET). The CRG's behavior for CMRESET is the same compared to external reset, but another reset vector is fetched after completion of the reset sequence. If the SCME bit is asserted the CRG generates a SCM interrupt if enabled (SCMIE = 1). After generating the interrupt the CRG enters self-clock mode and starts the clock quality checker (Section 7.4.1.4, "Clock Quality Checker"). Then the MCU continues with normal operation. If the SCM interrupt is blocked by SCMIE=0, the SCMIF flag will be asserted but the CRG will not wake-up from pseudo stop mode.

If any other interrupt source (e.g., RTI) triggers exit from pseudo stop mode, the MCU immediately continues with normal operation. Because the PLL has been powered-down during stop mode, the PLLSEL bit is cleared and the MCU runs on OSCCLK after leaving stop mode. The software must set the PLLSEL bit again, in order to switch system and core clocks to the PLLCLK.

Table 7-13 summarizes the outcome of a clock loss while in pseudo stop mode.

Table 7-13. Outcome of Clock Loss in Pseudo Stop Mode

CME	SCME	SCMIE	CRG Actions
0	X	X	Clock failure --> No action, clock loss not detected.
1	0	X	Clock failure --> CRG performs Clock Monitor Reset immediately
1	1	0	Clock Monitor failure --> Scenario 1: OSCCLK **recovers** prior to exiting pseudo stop mode. – MCU remains in pseudo stop mode, – VREG enabled, – PLL enabled, – SCM activated, – Start clock quality check, – Set SCMIF interrupt flag. *Some time later OSCCLK recovers.* – CM no longer indicates a failure, – 4096 OSCCLK cycles later clock quality check indicates clock o.k., – SCM deactivated, – PLL disabled, – VREG disabled. – MCU remains in pseudo stop mode. *Some time later either a wakeup interrupt occurs (no SCM interrupt)* – Exit pseudo stop mode using OSCCLK as system clock (SYSCLK), – Continue normal operation. *or an External Reset is applied.* – Exit pseudo stop mode using OSCCLK as system clock, – Start reset sequence. Scenario 2: OSCCLK **does not recover** prior to exiting pseudo stop mode. – MCU remains in pseudo stop mode, – VREG enabled, – PLL enabled, – SCM activated, – Start clock quality check, – Set SCMIF interrupt flag, – Keep performing clock quality checks (could continue infinitely) while in pseudo stop mode. *Some time later either a wakeup interrupt occurs (no SCM interrupt)* – Exit pseudo stop mode in SCM using PLL clock (f_{SCM}) as system clock – Continue to perform additional clock quality checks until OSCCLK is o.k. again. or an External RESET is applied. – Exit pseudo stop mode in SCM using PLL clock (f_{SCM}) as system clock – Start reset sequence, – Continue to perform additional clock quality checks until OSCCLK is o.k.again.
1	1	1	Clock failure --> – VREG enabled, – PLL enabled, – SCM activated, – Start clock quality check, – SCMIF set. SCMIF generates self clock mode wakeup interrupt. – Exit pseudo stop mode in SCM using PLL clock (f_{SCM}) as system clock, – Continue to perform a additional clock quality checks until OSCCLK is o.k. again.

7.4.3.3.2 Wake-up from Full Stop (PSTP = 0)

The MCU requires an external interrupt or an external reset in order to wake-up from stop-mode.

If the MCU gets an external reset during full stop mode active, the CRG asynchronously restores all configuration bits in the register space to its default settings and will perform a maximum of 50 clock *check_windows* (see Section 7.4.1.4, “Clock Quality Checker”). After completing the clock quality check the CRG starts the reset generator. After completing the reset sequence processing begins by fetching the normal reset vector. Full stop-mode is left and the MCU is in run mode again.

If the MCU is woken-up by an interrupt and the fast wake-up feature is disabled (FSTWKP = 0 or SCME = 0), the CRG will also perform a maximum of 50 clock *check_window*s (see Section 7.4.1.4, “Clock Quality Checker”). If the clock quality check is successful, the CRG will release all system and core clocks and will continue with normal operation. If all clock checks within the Timeout-Window are failing, the CRG will switch to self-clock mode or generate a clock monitor reset (CMRESET) depending on the setting of the SCME bit.

If the MCU is woken-up by an interrupt and the fast wake-up feature is enabled (FSTWKP = 1 and SCME = 1), the system will immediately resume operation in self-clock mode (see Section 7.4.1.4, “Clock Quality Checker”). The SCMIF flag will not be set. The system will remain in self-clock mode with oscillator disabled until FSTWKP bit is cleared. The clearing of FSTWKP will start the oscillator and the clock quality check. If the clock quality check is successful, the CRG will switch all system clocks to oscillator clock. The SCMIF flag will be set. See application examples in Figure 7-23 and Figure 7-24.

Because the PLL has been powered-down during stop-mode the PLLSEL bit is cleared and the MCU runs on OSCCLK after leaving stop-mode. The software must manually set the PLLSEL bit again, in order to switch system and core clocks to the PLLCLK.

NOTE

In full stop mode or self-clock mode caused by the fast wake-up feature, the clock monitor and the oscillator are disabled.

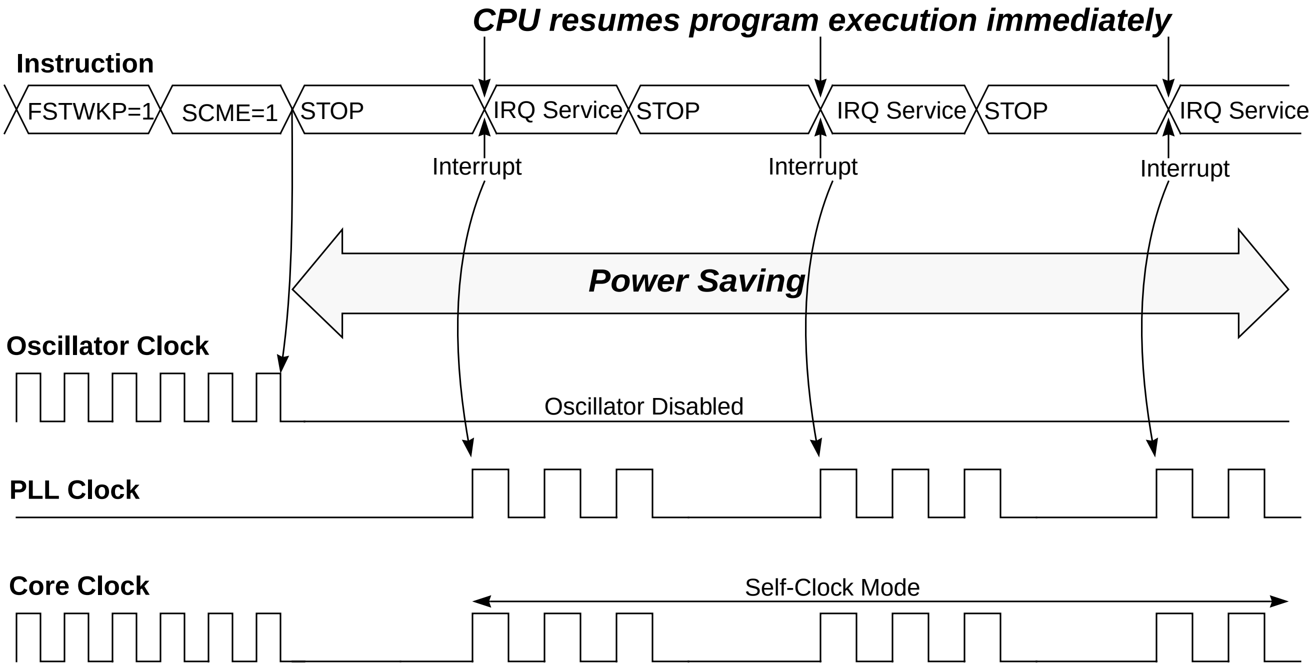


Figure 7-23. Fast Wake-up from Full Stop Mode: Example 1

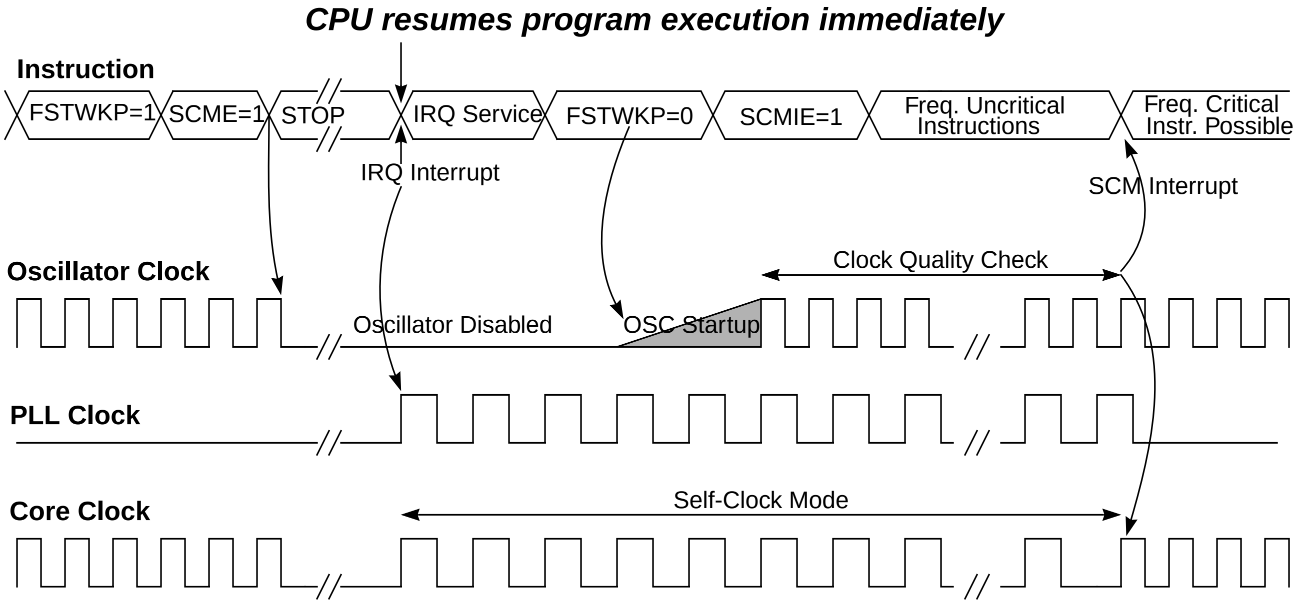


Figure 7-24. Fast Wake-up from Full Stop Mode: Example 2

7.5 Resets

This section describes how to reset the CRG, and how the CRG itself controls the reset of the MCU. It explains all special reset requirements. Since the reset generator for the MCU is part of the CRG, this section also describes all automatic actions that occur during or as a result of individual reset conditions. The reset values of registers and signals are provided in Section 7.3, “Memory Map and Register Definition”. All reset sources are listed in Table 7-14. Refer to MCU specification for related vector addresses and priorities.

Table 7-14. Reset Summary

Reset Source	Local Enable
Power on Reset	None
Low Voltage Reset	None
External Reset	None
Illegal Address Reset	None
Clock Monitor Reset	PLLCTL (CME = 1, SCME = 0)
COP Watchdog Reset	COPCTL (CR[2:0] nonzero)

7.5.1 Description of Reset Operation

The reset sequence is initiated by any of the following events:

- Low level is detected at the $\overline{\text{RESET}}$ pin (external reset)
- Power on is detected
- Low voltage is detected
- Illegal Address Reset is detected (see S12XMMC Block Guide for details)
- COP watchdog times out
- Clock monitor failure is detected and self-clock mode was disabled (SCME=0)

Upon detection of any reset event, an internal circuit drives the $\overline{\text{RESET}}$ pin low for 128 SYSCLK cycles (see Figure 7-25). Since entry into reset is asynchronous, it does not require a running SYSCLK. However, the internal reset circuit of the CRG cannot sequence out of current reset condition without a running SYSCLK. The number of 128 SYSCLK cycles might be increased by n = 3 to 6 additional SYSCLK cycles depending on the internal synchronization latency. After 128 + n SYSCLK cycles the $\overline{\text{RESET}}$ pin is released. The reset generator of the CRG waits for additional 64 SYSCLK cycles and then samples the RESET pin to determine the originating source. Table 7-15 shows which vector will be fetched.

Table 7-15. Reset Vector Selection

Sampled $\overline{\text{RESET}}$ Pin (64 cycles after release)	Clock Monitor Reset Pending	COP Reset Pending	Vector Fetch
1	0	0	POR / LVR / Illegal Address Reset / External Reset
1	1	X	Clock Monitor Reset
1	0	1	COP Reset
0	X	X	POR / LVR / Illegal Address Reset / External Reset with rise of $\overline{\text{RESET}}$ pin

NOTE

> External circuitry connected to the $\overline{\text{RESET}}$ pin should not include a large capacitance that would interfere with the ability of this signal to rise to a valid logic 1 within 64 SYSCLK cycles after the low drive is released.

The internal reset of the MCU remains asserted while the reset generator completes the 192 SYSCLK long reset sequence. The reset generator circuitry always makes sure the internal reset is deasserted synchronously after completion of the 192 SYSCLK cycles. In case the $\overline{\text{RESET}}$ pin is externally driven low for more than these 192 SYSCLK cycles (external reset), the internal reset remains asserted too.

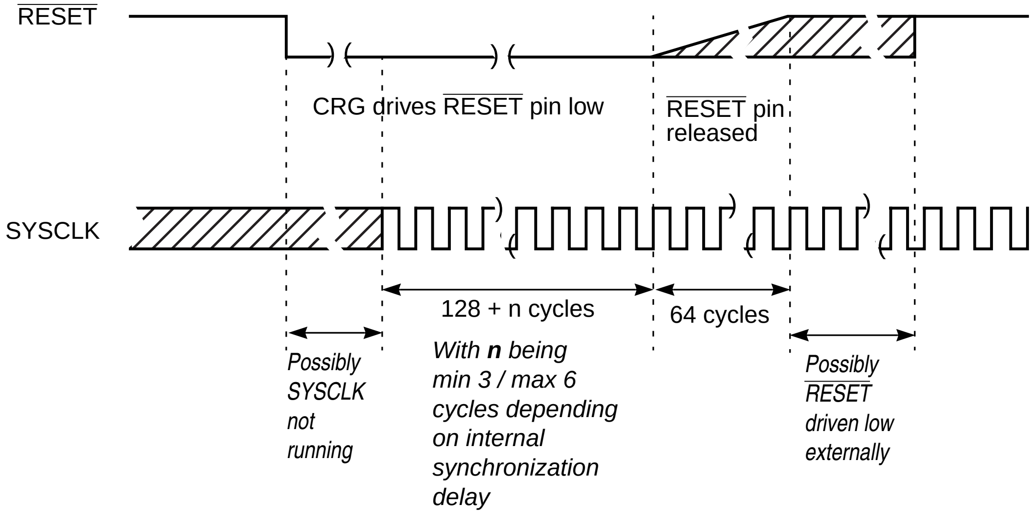


Figure 7-25. RESET Timing

7.5.2 Clock Monitor Reset

The CRG generates a clock monitor reset in case all of the following conditions are true:

- Clock monitor is enabled (CME = 1)
- Loss of clock is detected
- Self-clock mode is disabled (SCME = 0).

The reset event asynchronously forces the configuration registers to their default settings (see Section 7.3, “Memory Map and Register Definition”). In detail the CME and the SCME are reset to logical ‘1’ (which doesn’t change the state of the CME bit, because it has already been set). As a consequence the CRG immediately enters self clock mode and starts its internal reset sequence. In parallel the clock quality check starts. As soon as clock quality check indicates a valid oscillator clock the CRG switches to OSCCLK and leaves self clock mode. Since the clock quality checker is running in parallel to the reset generator, the CRG may leave self clock mode while still completing the internal reset sequence. When the reset sequence is finished, the CRG checks the internally latched state of the clock monitor fail circuit. If a clock monitor fail is indicated, processing begins by fetching the clock monitor reset vector.

7.5.3 Computer Operating Properly Watchdog (COP) Reset

When COP is enabled, the CRG expects sequential write of 0x_55 and 0x_AA (in this order) to the ARMCOP register during the selected time-out period. Once this is done, the COP time-out period restarts. If the program fails to do this the CRG will generate a reset. Also, if any value other than 0x_55 or 0x_AA is written, the CRG immediately generates a reset. In case windowed COP operation is enabled writes (0x_55 or 0x_AA) to the ARMCOP register must occur in the last 25% of the selected time-out period. A premature write the CRG will immediately generate a reset.

As soon as the reset sequence is completed the reset generator checks the reset condition. If no clock monitor failure is indicated and the latched state of the COP timeout is true, processing begins by fetching the COP vector.

7.5.4 Power On Reset, Low Voltage Reset

The on-chip voltage regulator detects when V_{DD} to the MCU has reached a certain level and asserts power on reset or low voltage reset or both. As soon as a power on reset or low voltage reset is triggered the CRG performs a quality check on the incoming clock signal. As soon as clock quality check indicates a valid oscillator clock signal, the reset sequence starts using the oscillator clock. If after 50 check windows the clock quality check indicated a non-valid oscillator clock, the reset sequence starts using self-clock mode.

Figure 7-26 and Figure 7-27 show the power-up sequence for cases when the $\overline{\text{RESET}}$ pin is tied to V_{DD} and when the $\overline{\text{RESET}}$ pin is held low.

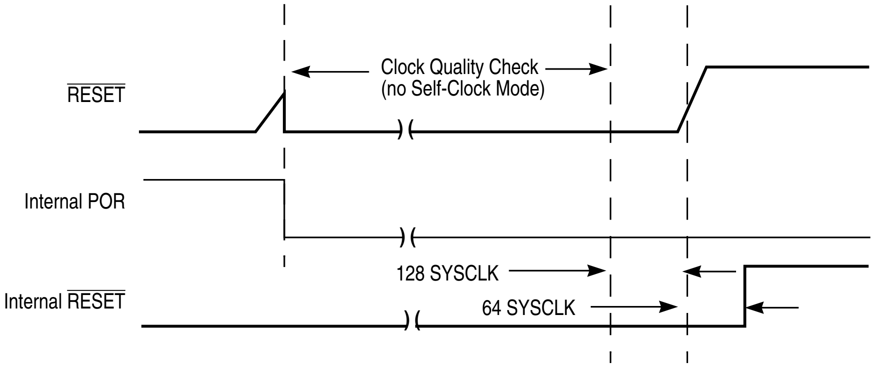


Figure 7-26. $\overline{\text{RESET}}$ Pin Tied to V_{DD} (by a pull-up resistor)

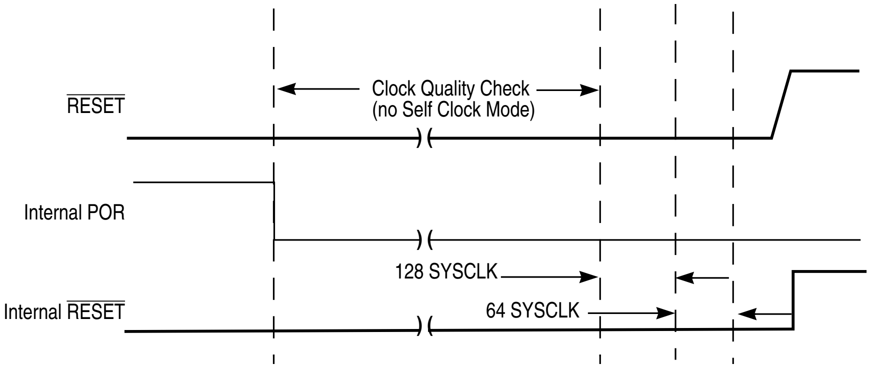


Figure 7-27. $\overline{\text{RESET}}$ Pin Held Low Externally

7.6 Interrupts

The interrupts/reset vectors requested by the CRG are listed in Table 7-16. Refer to MCU specification for related vector addresses and priorities.

Table 7-16. CRG Interrupt Vectors

Interrupt Source	CCR Mask	Local Enable
Real time interrupt	I bit	CRGINT (RTIE)
LOCK interrupt	I bit	CRGINT (LOCKIE)
SCM interrupt	I bit	CRGINT (SCMIE)

7.6.1 Real Time Interrupt

The CRG generates a real time interrupt when the selected interrupt time period elapses. RTI interrupts are locally disabled by setting the RTIE bit to 0. The real time interrupt flag (RTIF) is set to1 when a timeout occurs, and is cleared to 0 by writing a 1 to the RTIF bit.

The RTI continues to run during pseudo stop mode if the PRE bit is set to 1. This feature can be used for periodic wakeup from pseudo stop if the RTI interrupt is enabled.

7.6.2 PLL Lock Interrupt

The CRG generates a PLL Lock interrupt when the LOCK condition of the PLL has changed, either from a locked state to an unlocked state or vice versa. Lock interrupts are locally disabled by setting the LOCKIE bit to 0. The PLL Lock interrupt flag (LOCKIF) is set to1 when the LOCK condition has changed, and is cleared to 0 by writing a 1 to the LOCKIF bit.

7.6.3 Self Clock Mode Interrupt

The CRG generates a self clock mode interrupt when the SCM condition of the system has changed, either entered or exited self clock mode. SCM conditions can only change if the self clock mode enable bit (SCME) is set to 1. SCM conditions are caused by a failing clock quality check after power on reset (POR) or low voltage reset (LVR) or recovery from full stop mode (PSTP = 0) or clock monitor failure. For details on the clock quality check refer to Section 7.4.1.4, “Clock Quality Checker”. If the clock monitor is enabled (CME = 1) a loss of external clock will also cause a SCM condition (SCME = 1).

SCM interrupts are locally disabled by setting the SCMIE bit to 0. The SCM interrupt flag (SCMIF) is set to1 when the SCM condition has changed, and is cleared to 0 by writing a 1 to the SCMIF bit.

Chapter 8
Pierce Oscillator (S12XOSCLCPV1)

8.1 Introduction

The Pierce oscillator (XOSC) module provides a robust, low-noise and low-power clock source. The module will be operated from the V_{DDPLL} supply rail (2.5 V nominal) and require the minimum number of external components. It is designed for optimal start-up margin with typical crystal oscillators.

8.1.1 Features

The XOSC will contain circuitry to dynamically control current gain in the output amplitude. This ensures a signal with low harmonic distortion, low power and good noise immunity.

- High noise immunity due to input hysteresis
- Low RF emissions with peak-to-peak swing limited dynamically
- Transconductance (gm) sized for optimum start-up margin for typical oscillators
- Dynamic gain control eliminates the need for external current limiting resistor
- Integrated resistor eliminates the need for external bias resistor
- Low power consumption:
  - Operates from 2.5 V (nominal) supply
  - Amplitude control limits power
- Clock monitor

8.1.2 Modes of Operation

Two modes of operation exist:

1. Loop controlled Pierce oscillator
2. External square wave mode featuring also full swing Pierce without internal feedback resistor

8.1.3 Block Diagram

Figure 8-1 shows a block diagram of the XOSC.

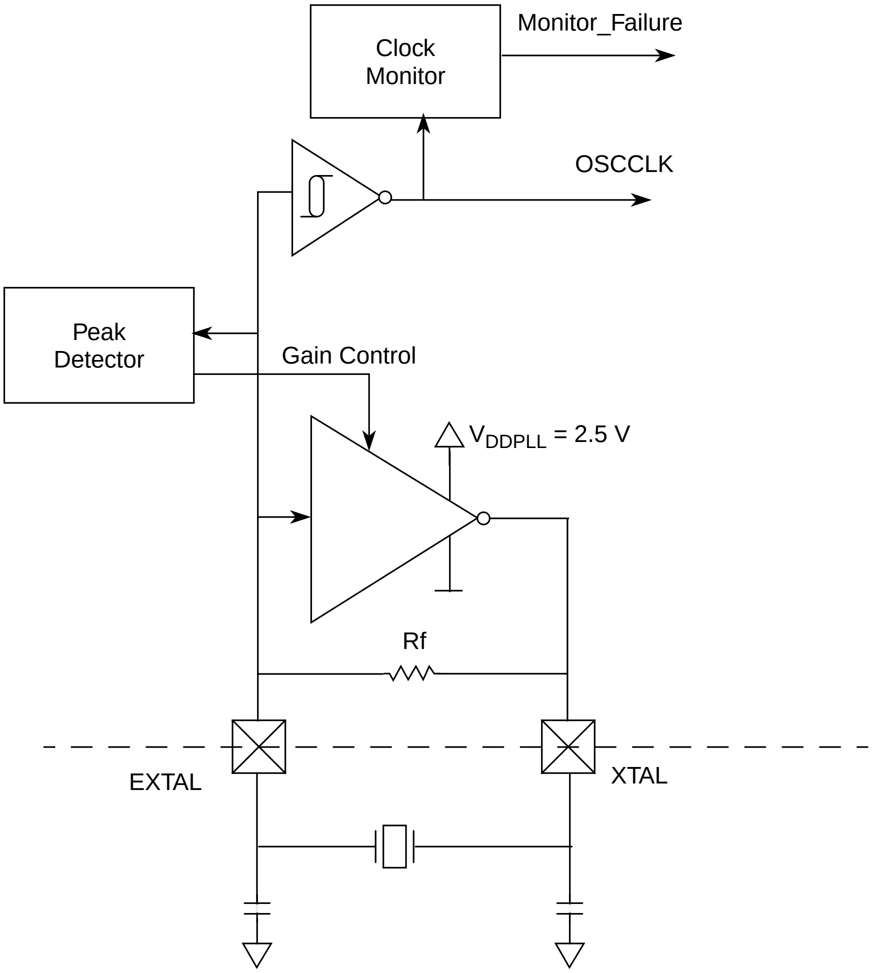


Figure 8-1. XOSC Block Diagram

8.2 External Signal Description

This section lists and describes the signals that connect off chip

8.2.1 V_{DDPLL} and V_{SSPLL} — Operating and Ground Voltage Pins

Theses pins provides operating voltage (V_{DDPLL}) and ground (V_{SSPLL}) for the XOSC circuitry. This allows the supply voltage to the XOSC to be independently bypassed.

8.2.2 EXTAL and XTAL — Input and Output Pins

These pins provide the interface for either a crystal or a CMOS compatible clock to control the internal clock generator circuitry. EXTAL is the external clock input or the input to the crystal oscillator amplifier. XTAL is the output of the crystal oscillator amplifier. The MCU internal system clock is derived from the

EXTAL input frequency. In full stop mode (PSTP = 0), the EXTAL pin is pulled down by an internal resistor of typical 200 kΩ.

NOTE

Freescale recommends an evaluation of the application board and chosen resonator or crystal by the resonator or crystal supplier.

Loop controlled circuit is not suited for overtone resonators and crystals.

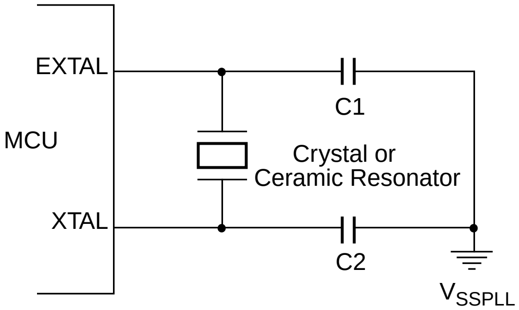
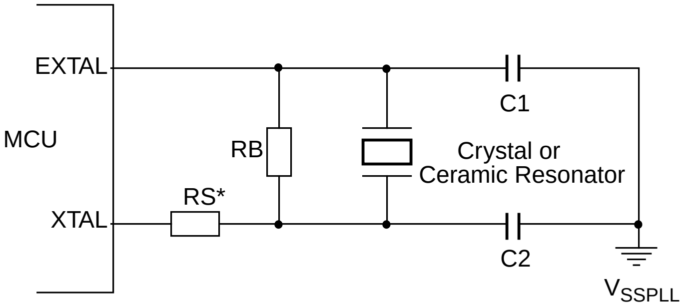


Figure 8-2. Loop Controlled Pierce Oscillator Connections (XCLKS = 0)

NOTE

Full swing Pierce circuit is not suited for overtone resonators and crystals without a careful component selection.



* R_S can be zero (shorted) when use with higher frequency crystals. Refer to manufacturer's data.

Figure 8-3. Full Swing Pierce Oscillator Connections (XCLKS = 1)

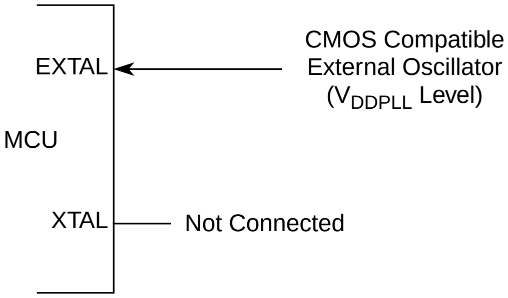


Figure 8-4. External Clock Connections (XCLKS = 1)

8.2.3 XCLKS — Input Signal

The XCLKS is an input signal which controls whether a crystal in combination with the internal loop controlled (low power) Pierce oscillator is used or whether full swing Pierce oscillator/external clock circuitry is used. Refer to the Device Overview chapter for polarity and sampling conditions of the XCLKS pin. Table 8-1 lists the state coding of the sampled XCLKS signal.

Table 8-1. Clock Selection Based on XCLKS

XCLKS	Description
0	Loop controlled Pierce oscillator selected
1	Full swing Pierce oscillator/external clock selected

8.3 Memory Map and Register Definition

The CRG contains the registers and associated bits for controlling and monitoring the oscillator module.

8.4 Functional Description

The XOSC module has control circuitry to maintain the crystal oscillator circuit voltage level to an optimal level which is determined by the amount of hysteresis being used and the maximum oscillation range.

The oscillator block has two external pins, EXTAL and XTAL. The oscillator input pin, EXTAL, is intended to be connected to either a crystal or an external clock source. The selection of loop controlled Pierce oscillator or full swing Pierce oscillator/external clock depends on the XCLKS signal which is sampled during reset. The XTAL pin is an output signal that provides crystal circuit feedback.

A buffered EXTAL signal becomes the internal clock. To improve noise immunity, the oscillator is powered by the V_{DDPLL} and V_{SSPLL} power supply pins.

8.4.1 Gain Control

A closed loop control system will be utilized whereby the amplifier is modulated to keep the output waveform sinusoidal and to limit the oscillation amplitude. The output peak to peak voltage will be kept above twice the maximum hysteresis level of the input buffer. Electrical specification details are provided in the Electrical Characteristics appendix.

8.4.2 Clock Monitor

The clock monitor circuit is based on an internal RC time delay so that it can operate without any MCU clocks. If no OSCCLK edges are detected within this RC time delay, the clock monitor indicates failure which asserts self-clock mode or generates a system reset depending on the state of SCME bit. If the clock monitor is disabled or the presence of clocks is detected no failure is indicated.The clock monitor function is enabled/disabled by the CME control bit, described in the CRG block description chapter.

8.4.3 Wait Mode Operation

During wait mode, XOSC is not impacted.

8.4.4 Stop Mode Operation

XOSC is placed in a static state when the part is in stop mode except when pseudo-stop mode is enabled. During pseudo-stop mode, XOSC is not impacted.

Chapter 9
Analog-to-Digital Converter (ATD10B16CV4) Block Description

9.1 Introduction

The ATD10B16C is a 16-channel, 10-bit, multiplexed input successive approximation analog-to-digital converter. Refer to the Electrical Specifications chapter for ATD accuracy.

9.1.1 Features

- 8-/10-bit resolution
- 7 µs, 10-bit single conversion time
- Sample buffer amplifier
- Programmable sample time
- Left/right justified, signed/unsigned result data
- External trigger control
- Conversion completion interrupt generation
- Analog input multiplexer for 16 analog input channels
- Analog/digital input pin multiplexing
- 1 to 16 conversion sequence lengths
- Continuous conversion mode
- Multiple channel scans
- Configurable external trigger functionality on any AD channel or any of four additional trigger inputs. The four additional trigger inputs can be chip external or internal. Refer to device specification for availability and connectivity
- Configurable location for channel wrap around (when converting multiple channels in a sequence)

9.1.2 Modes of Operation

There is software programmable selection between performing **single** or **continuous conversion** on a **single channel** or **multiple channels**.

9.1.3 Block Diagram

Refer to Figure 9-1 for a block diagram of the ATD0B16C block.

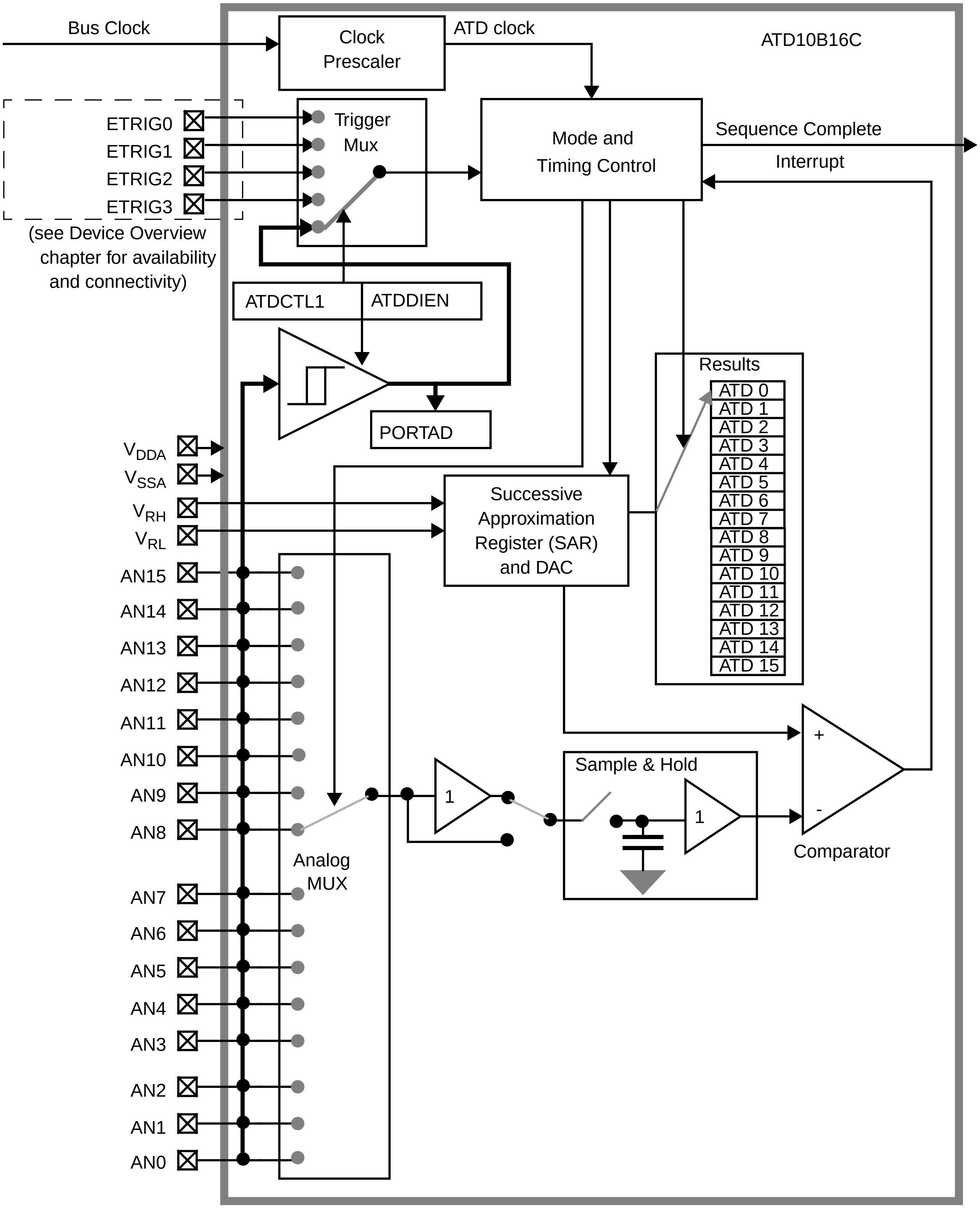


Figure 9-1. ATD10B16C Block Diagram

9.2 External Signal Description

This section lists all inputs to the ATD10B16C block.

9.2.1 AN*x* (*x* = 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0) — Analog Input Channel *x* Pins

This pin serves as the analog input channel *x*. It can also be configured as general-purpose digital input and/or external trigger for the ATD conversion.

9.2.2 ETRIG3, ETRIG2, ETRIG1, ETRIG0 — External Trigger Pins

These inputs can be configured to serve as an external trigger for the ATD conversion.

Refer to the Device Overview chapter for availability and connectivity of these inputs.

9.2.3 V_{RH}, V_{RL} — High Reference Voltage Pin, Low Reference Voltage Pin

V_{RH} is the high reference voltage, V_{RL} is the low reference voltage for ATD conversion.

9.2.4 V_{DDA}, V_{SSA} — Analog Circuitry Power Supply Pins

These pins are the power supplies for the analog circuitry of the ATD10B16CV4 block.

9.3 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in the ATD10B16C.

9.3.1 Module Memory Map

Table 9-1 gives an overview of all ATD10B16C registers

Table 9-1. ATD10B16CV4 Memory Map

Address Offset	Use	Access
0x0000	ATD Control Register 0 (ATDCTL0)	R/W
0x0001	ATD Control Register 1 (ATDCTL1)	R/W
0x0002	ATD Control Register 2 (ATDCTL2)	R/W
0x0003	ATD Control Register 3 (ATDCTL3)	R/W
0x0004	ATD Control Register 4 (ATDCTL4)	R/W
0x0005	ATD Control Register 5 (ATDCTL5)	R/W
0x0006	ATD Status Register 0 (ATDSTAT0)	R/W
0x0007	Unimplemented	
0x0008	ATD Test Register 0 (ATDTEST0)(1)	R
0x0009	ATD Test Register 1 (ATDTEST1)	R/W
0x000A	ATD Status Register 2 (ATDSTAT2)	R
0x000B	ATD Status Register 1 (ATDSTAT1)	R
0x000C	ATD Input Enable Register 0 (ATDDIEN0)	R/W
0x000D	ATD Input Enable Register 1 (ATDDIEN1)	R/W
0x000E	Port Data Register 0 (PORTAD0)	R
0x000F	Port Data Register 1 (PORTAD1)	R
0x0010, 0x0011	ATD Result Register 0 (ATDDR0H, ATDDR0L)	R/W
0x0012, 0x0013	ATD Result Register 1 (ATDDR1H, ATDDR1L)	R/W
0x0014, 0x0015	ATD Result Register 2 (ATDDR2H, ATDDR2L)	R/W
0x0016, 0x0017	ATD Result Register 3 (ATDDR3H, ATDDR3L)	R/W
0x0018, 0x0019	ATD Result Register 4 (ATDDR4H, ATDDR4L)	R/W
0x001A, 0x001B	ATD Result Register 5 (ATDDR5H, ATDDR5L)	R/W
0x001C, 0x001D	ATD Result Register 6 (ATDDR6H, ATDDR6L)	R/W
0x001E, 0x001F	ATD Result Register 7 (ATDDR7H, ATDDR7L)	R/W
0x0020, 0x0021	ATD Result Register 8 (ATDDR8H, ATDDR8L)	R/W
0x0022, 0x0023	ATD Result Register 9 (ATDDR9H, ATDDR9L)	R/W
0x0024, 0x0025	ATD Result Register 10 (ATDDR10H, ATDDR10L)	R/W
0x0026, 0x0027	ATD Result Register 11 (ATDDR11H, ATDDR11L)	R/W
0x0028, 0x0029	ATD Result Register 12 (ATDDR12H, ATDDR12L)	R/W
0x002A, 0x002B	ATD Result Register 13 (ATDDR13H, ATDDR13L)	R/W
0x002C, 0x002D	ATD Result Register 14 (ATDDR14H, ATDDR14L)	R/W
0x002E, 0x002F	ATD Result Register 15 (ATDDR15H, ATDDR15L)	R/W

1. ATDTEST0 is intended for factory test purposes only.

NOTE

Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level.

9.3.2 Register Descriptions

This section describes in address order all the ATD10B16C registers and their individual bits.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 ATDCTL0	R	0	0	0	0	WRAP3	WRAP2	WRAP1	WRAP0
	W	▨	▨	▨	▨				
0x0001 ATDCTL1	R	ETRIGSEL	0	0	0	ETRIGCH3	ETRIGCH2	ETRIGCH1	ETRIGCH0
	W		▨	▨	▨				
0x0002 ATDCTL2	R	ADPU	AFFC	AWAI	ETRIGLE	ETRIGP	ETRIGE	ASCIE	ASCIF
	W								▨
0x0003 ATDCTL3	R	0	S8C	S4C	S2C	S1C	FIFO	FRZ1	FRZ0
	W	▨							
0x0004 ATDCTL4	R	SRES8	SMP1	SMP0	PRS4	PRS3	PRS2	PRS1	PRS0
	W								
0x0005 ATDCTL5	R	DJM	DSGN	SCAN	MULT	CD	CC	CB	CA
	W								
0x0006 ATDSTAT0	R	SCF	0	ETORF	FIFOR	CC3	CC2	CC1	CC0
	W		▨			▨	▨	▨	▨
0x0007 Unimplemented	R	▨	▨	▨	▨	▨	▨	▨	▨
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0008 ATDTEST0	R	Unimplemented							
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0009 ATDTEST1	R	Unimplemented							SC
	W	▨	▨	▨	▨	▨	▨	▨	
0x000A ATDSTAT2	R	CCF15	CCF14	CCF13	CCF12	CCF11	CCF10	CCF9	CCF8
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x000B ATDSTAT1	R	CCF7	CCF6	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x000C ATDDIEN0	R	IEN15	IEN14	IEN13	IEN12	IEN11	IEN10	IEN9	IEN8
	W								

▨ = Unimplemented or Reserved u = Unaffected

Figure 9-2. ATD Register Summary

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x000D ATDDIEN1	R W	IEN7	IEN6	IEN5	IEN4	IEN3	IEN2	IEN1	IEN0
0x000E PORTAD0	R	PTAD15	PTAD14	PTAD13	PTAD12	PTAD11	PTAD10	PTAD9	PTAD8
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x000F PORTAD1	R	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0010–0x002F ATDDRxH– ATDDRxL	R	BIT 9 MSB BIT 7 MSB	BIT 8 BIT 6	BIT 7 BIT 5	BIT 6 BIT 4	BIT 5 BIT 3	BIT 4 BIT 2	BIT 3 BIT 1	BIT 2 BIT 0
	W	▨	▨	▨	▨	▨	▨	▨	▨
	R	BIT 1 u	BIT 0 u	0 0	0 0	0 0	0 0	0 0	0 0
	W	▨	▨	▨	▨	▨	▨	▨	▨

▨ = Unimplemented or Reserved u = Unaffected

Figure 9-2. ATD Register Summary (continued)

9.3.2.1 ATD Control Register 0 (ATDCTL0)

Writes to this register will abort current conversion sequence but will not start a new sequence.

Module Base + 0x0000

	7	6	5	4	3	2	1	0
R	0	0	0	0	WRAP3	WRAP2	WRAP1	WRAP0
W	▨	▨	▨	▨				
Reset	0	0	0	0	1	1	1	1

▨ = Unimplemented or Reserved

Figure 9-3. ATD Control Register 0 (ATDCTL0)

Read: Anytime

Write: Anytime

Table 9-2. ATDCTL0 Field Descriptions

Field	Description
3:0 WRAP[3:0]	**Wrap Around Channel Select Bits** — These bits determine the channel for wrap around when doing multi-channel conversions. The coding is summarized in Table 9-3.

Table 9-3. Multi-Channel Wrap Around Coding

WRAP3	WRAP2	WRAP1	WRAP0	Multiple Channel Conversions (MULT = 1) Wrap Around to AN0 after Converting
0	0	0	0	Reserved
0	0	0	1	AN1
0	0	1	0	AN2
0	0	1	1	AN3
0	1	0	0	AN4
0	1	0	1	AN5
0	1	1	0	AN6
0	1	1	1	AN7
1	0	0	0	AN8
1	0	0	1	AN9
1	0	1	0	AN10
1	0	1	1	AN11
1	1	0	0	AN12
1	1	0	1	AN13
1	1	1	0	AN14
1	1	1	1	AN15

9.3.2.2 ATD Control Register 1 (ATDCTL1)

Writes to this register will abort current conversion sequence but will not start a new sequence.

Module Base + 0x0001

	7	6	5	4	3	2	1	0
R	ETRIGSEL	0	0	0	ETRIGCH3	ETRIGCH2	ETRIGCH1	ETRIGCH0
W		▨	▨	▨				
Reset	0	0	0	0	1	1	1	1

▨ = Unimplemented or Reserved

Figure 9-4. ATD Control Register 1 (ATDCTL1)

Read: Anytime

Write: Anytime

Table 9-4. ATDCTL1 Field Descriptions

Field	Description
7 ETRIGSEL	**External Trigger Source Select** — This bit selects the external trigger source to be either one of the AD channels or one of the ETRIG[3:0] inputs. See device specification for availability and connectivity of ETRIG[3:0] inputs. If ETRIG[3:0] input option is not available, writing a 1 to ETRISEL only sets the bit but has no effect, that means one of the AD channels (selected by ETRIGCH[3:0]) remains the source for external trigger. The coding is summarized in Table 9-5.
3:0 ETRIGCH[3:0]	**External Trigger Channel Select** — These bits select one of the AD channels or one of the ETRIG[3:0] inputs as source for the external trigger. The coding is summarized in Table 9-5.

Table 9-5. External Trigger Channel Select Coding

ETRIGSEL	ETRIGCH3	ETRIGCH2	ETRIGCH1	ETRIGCH0	External Trigger Source
0	0	0	0	0	AN0
0	0	0	0	1	AN1
0	0	0	1	0	AN2
0	0	0	1	1	AN3
0	0	1	0	0	AN4
0	0	1	0	1	AN5
0	0	1	1	0	AN6
0	0	1	1	1	AN7
0	1	0	0	0	AN8
0	1	0	0	1	AN9
0	1	0	1	0	AN10
0	1	0	1	1	AN11
0	1	1	0	0	AN12
0	1	1	0	1	AN13
0	1	1	1	0	AN14
0	1	1	1	1	AN15
1	0	0	0	0	ETRIG0(1)
1	0	0	0	1	ETRIG1[1]
1	0	0	1	0	ETRIG2[1]
1	0	0	1	1	ETRIG3[1]
1	0	1	X	X	Reserved
1	1	X	X	X	Reserved

1. Only if ETRIG[3:0] input option is available (see device specification), else ETRISEL is ignored, that means external trigger source remains on one of the AD channels selected by ETRIGCH[3:0]

9.3.2.3 ATD Control Register 2 (ATDCTL2)

This register controls power down, interrupt and external trigger. Writes to this register will abort current conversion sequence but will not start a new sequence.

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R	ADPU	AFFC	AWAI	ETRIGLE	ETRIGP	ETRIGE	ASCIE	ASCIF
W								▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 9-5. ATD Control Register 2 (ATDCTL2)

Read: Anytime

Write: Anytime

Table 9-6. ATDCTL2 Field Descriptions

Field	Description
7 ADPU	**ATD Power Down** — This bit provides on/off control over the ATD10B16C block allowing reduced MCU power consumption. Because analog electronic is turned off when powered down, the ATD requires a recovery time period after ADPU bit is enabled. 0 Power down ATD 1 Normal ATD functionality
6 AFFC	**ATD Fast Flag Clear All** 0 ATD flag clearing operates normally (read the status register ATDSTAT1 before reading the result register to clear the associate CCF flag). 1 Changes all ATD conversion complete flags to a fast clear sequence. Any access to a result register will cause the associate CCF flag to clear automatically.
5 AWAI	**ATD Power Down in Wait Mode** — When entering Wait Mode this bit provides on/off control over the ATD10B16C block allowing reduced MCU power. Because analog electronic is turned off when powered down, the ATD requires a recovery time period after exit from Wait mode. 0 ATD continues to run in Wait mode 1 Halt conversion and power down ATD during Wait mode After exiting Wait mode with an interrupt conversion will resume. But due to the recovery time the result of this conversion should be ignored.
4 ETRIGLE	**External Trigger Level/Edge Control** — This bit controls the sensitivity of the external trigger signal. See Table 9-7 for details.
3 ETRIGP	**External Trigger Polarity** — This bit controls the polarity of the external trigger signal. See Table 9-7 for details.
2 ETRIGE	**External Trigger Mode Enable** — This bit enables the external trigger on one of the AD channels or one of the ETRIG[3:0] inputs as described in Table 9-5. If external trigger source is one of the AD channels, the digital input buffer of this channel is enabled. The external trigger allows to synchronize the start of conversion with external events. 0 Disable external trigger 1 Enable external trigger

Table 9-6. ATDCTL2 Field Descriptions (continued)

Field	Description
1 ASCIE	**ATD Sequence Complete Interrupt Enable** 0 ATD Sequence Complete interrupt requests are disabled. 1 ATD Interrupt will be requested whenever ASCIF = 1 is set.
0 ASCIF	**ATD Sequence Complete Interrupt Flag** — If ASCIE = 1 the ASCIF flag equals the SCF flag (see Section 9.3.2.7, “ATD Status Register 0 (ATDSTAT0)”), else ASCIF reads zero. Writes have no effect. 0 No ATD interrupt occurred 1 ATD sequence complete interrupt pending

Table 9-7. External Trigger Configurations

ETRIGLE	ETRIGP	External Trigger Sensitivity
0	0	Falling Edge
0	1	Ring Edge
1	0	Low Level
1	1	High Level

9.3.2.4 ATD Control Register 3 (ATDCTL3)

This register controls the conversion sequence length, FIFO for results registers and behavior in Freeze Mode. Writes to this register will abort current conversion sequence but will not start a new sequence.

Module Base + 0x0003

	7	6	5	4	3	2	1	0
R	0	S8C	S4C	S2C	S1C	FIFO	FRZ1	FRZ0
W	▨							
Reset	0	0	1	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 9-6. ATD Control Register 3 (ATDCTL3)

Read: Anytime

Write: Anytime

Table 9-8. ATDCTL3 Field Descriptions

Field	Description
6 S8C	**Conversion Sequence Length** — This bit controls the number of conversions per sequence. Table 9-9 shows all combinations. At reset, S4C is set to 1 (sequence length is 4). This is to maintain software continuity to HC12 Family.
5 S4C	**Conversion Sequence Length** — This bit controls the number of conversions per sequence. Table 9-9 shows all combinations. At reset, S4C is set to 1 (sequence length is 4). This is to maintain software continuity to HC12 Family.
4 S2C	**Conversion Sequence Length** — This bit controls the number of conversions per sequence. Table 9-9 shows all combinations. At reset, S4C is set to 1 (sequence length is 4). This is to maintain software continuity to HC12 Family.
3 S1C	**Conversion Sequence Length** — This bit controls the number of conversions per sequence. Table 9-9 shows all combinations. At reset, S4C is set to 1 (sequence length is 4). This is to maintain software continuity to HC12 Family.

Table 9-8. ATDCTL3 Field Descriptions (continued)

Field	Description
2 FIFO	**Result Register FIFO Mode** —If this bit is zero (non-FIFO mode), the A/D conversion results map into the result registers based on the conversion sequence; the result of the first conversion appears in the first result register, the second result in the second result register, and so on. If this bit is one (FIFO mode) the conversion counter is not reset at the beginning or ending of a conversion sequence; sequential conversion results are placed in consecutive result registers. In a continuously scanning conversion sequence, the result register counter will wrap around when it reaches the end of the result register file. The conversion counter value (CC3-0 in ATDSTAT0) can be used to determine where in the result register file, the current conversion result will be placed. Aborting a conversion or starting a new conversion by write to an ATDCTL register (ATDCTL5-0) clears the conversion counter even if FIFO=1. So the first result of a new conversion sequence, started by writing to ATDCTL5, will always be place in the first result register (ATDDDR0). Intended usage of FIFO mode is continuos conversion (SCAN=1) or triggered conversion (ETRIG=1). Finally, which result registers hold valid data can be tracked using the conversion complete flags. Fast flag clear mode may or may not be useful in a particular application to track valid data. 0 Conversion results are placed in the corresponding result register up to the selected sequence length. 1 Conversion results are placed in consecutive result registers (wrap around at end).
1:0 FRZ[1:0]	**Background Debug Freeze Enable** — When debugging an application, it is useful in many cases to have the ATD pause when a breakpoint (Freeze Mode) is encountered. These 2 bits determine how the ATD will respond to a breakpoint as shown in Table 9-10. Leakage onto the storage node and comparator reference capacitors may compromise the accuracy of an immediately frozen conversion depending on the length of the freeze period.

Table 9-9. Conversion Sequence Length Coding

S8C	S4C	S2C	S1C	Number of Conversions per Sequence
0	0	0	0	16
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Table 9-10. ATD Behavior in Freeze Mode (Breakpoint)

FRZ1	FRZ0	Behavior in Freeze Mode
0	0	Continue conversion
0	1	Reserved
1	0	Finish current conversion, then freeze
1	1	Freeze Immediately

9.3.2.5 ATD Control Register 4 (ATDCTL4)

This register selects the conversion clock frequency, the length of the second phase of the sample time and the resolution of the A/D conversion (i.e., 8-bits or 10-bits). Writes to this register will abort current conversion sequence but will not start a new sequence.

Module Base + 0x0004

	7	6	5	4	3	2	1	0
R W	SRES8	SMP1	SMP0	PRS4	PRS3	PRS2	PRS1	PRS0
Reset	0	0	0	0	0	1	0	1

Figure 9-7. ATD Control Register 4 (ATDCTL4)

Read: Anytime

Write: Anytime

Table 9-11. ATDCTL4 Field Descriptions

Field	Description
7 SRES8	**A/D Resolution Select** — This bit selects the resolution of A/D conversion results as either 8 or 10 bits. The A/D converter has an accuracy of 10 bits. However, if low resolution is required, the conversion can be speeded up by selecting 8-bit resolution. 0 10 bit resolution 1 8 bit resolution
6:5 SMP[1:0]	**Sample Time Select** —These two bits select the length of the second phase of the sample time in units of ATD conversion clock cycles. Note that the ATD conversion clock period is itself a function of the prescaler value (bits PRS4-0). The sample time consists of two phases. The first phase is two ATD conversion clock cycles long and transfers the sample quickly (via the buffer amplifier) onto the A/D machine's storage node. The second phase attaches the external analog signal directly to the storage node for final charging and high accuracy. Table 9-12 lists the lengths available for the second sample phase.
4:0 PRS[4:0]	**ATD Clock Prescaler** — These 5 bits are the binary value prescaler value PRS. The ATD conversion clock frequency is calculated as follows: $$\text{ATDclock} = \frac{[\text{BusClock}]}{[\text{PRS} + 1]} \times 0.5$$ **Note:** The maximum ATD conversion clock frequency is half the bus clock. The default (after reset) prescaler value is 5 which results in a default ATD conversion clock frequency that is bus clock divided by 12. Table 9-13 illustrates the divide-by operation and the appropriate range of the bus clock.

Table 9-12. Sample Time Select

SMP1	SMP0	Length of 2nd Phase of Sample Time
0	0	2 A/D conversion clock periods
0	1	4 A/D conversion clock periods
1	0	8 A/D conversion clock periods
1	1	16 A/D conversion clock periods

Table 9-13. Clock Prescaler Values

Prescale Value	Total Divisor Value	Max. Bus Clock[1]	Min. Bus Clock[2]
00000	Divide by 2	4 MHz	1 MHz
00001	Divide by 4	8 MHz	2 MHz
00010	Divide by 6	12 MHz	3 MHz
00011	Divide by 8	16 MHz	4 MHz
00100	Divide by 10	20 MHz	5 MHz
00101	Divide by 12	24 MHz	6 MHz
00110	Divide by 14	28 MHz	7 MHz
00111	Divide by 16	32 MHz	8 MHz
01000	Divide by 18	36 MHz	9 MHz
01001	Divide by 20	40 MHz	10 MHz
01010	Divide by 22	44 MHz	11 MHz
01011	Divide by 24	48 MHz	12 MHz
01100	Divide by 26	52 MHz	13 MHz
01101	Divide by 28	56 MHz	14 MHz
01110	Divide by 30	60 MHz	15 MHz
01111	Divide by 32	64 MHz	16 MHz
10000	Divide by 34	68 MHz	17 MHz
10001	Divide by 36	72 MHz	18 MHz
10010	Divide by 38	76 MHz	19 MHz
10011	Divide by 40	80 MHz	20 MHz
10100	Divide by 42	84 MHz	21 MHz
10101	Divide by 44	88 MHz	22 MHz
10110	Divide by 46	92 MHz	23 MHz
10111	Divide by 48	96 MHz	24 MHz
11000	Divide by 50	100 MHz	25 MHz
11001	Divide by 52	104 MHz	26 MHz
11010	Divide by 54	108 MHz	27 MHz
11011	Divide by 56	112 MHz	28 MHz
11100	Divide by 58	116 MHz	29 MHz
11101	Divide by 60	120 MHz	30 MHz
11110	Divide by 62	124 MHz	31 MHz
11111	Divide by 64	128 MHz	32 MHz

[1] Maximum ATD conversion clock frequency is 2 MHz. The maximum allowed bus clock frequency is shown in this column.

[2] Minimum ATD conversion clock frequency is 500 kHz. The minimum allowed bus clock frequency is shown in this column.

9.3.2.6 ATD Control Register 5 (ATDCTL5)

This register selects the type of conversion sequence and the analog input channels sampled. Writes to this register will abort current conversion sequence and start a new conversion sequence. If external trigger is enabled (ETRIGE = 1) an initial write to ATDCTL5 is required to allow starting of a conversion sequence which will then occur on each trigger event. Start of conversion means the beginning of the sampling phase.

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R W	DJM	DSGN	SCAN	MULT	CD	CC	CB	CA
Reset	0	0	0	0	0	0	0	0

Figure 9-8. ATD Control Register 5 (ATDCTL5)

Read: Anytime

Write: Anytime

Table 9-14. ATDCTL5 Field Descriptions

Field	Description
7 DJM	**Result Register Data Justification** — This bit controls justification of conversion data in the result registers. See Section 9.3.2.16, “ATD Conversion Result Registers (ATDDRx)” for details. 0 Left justified data in the result registers. 1 Right justified data in the result registers.
6 DSGN	**Result Register Data Signed or Unsigned Representation** — This bit selects between signed and unsigned conversion data representation in the result registers. Signed data is represented as 2’s complement. Signed data is not available in right justification. See <st-bold>9.3.2.16 ATD Conversion Result Registers (ATDDRx) for details. 0 Unsigned data representation in the result registers. 1 Signed data representation in the result registers. Table 9-15 summarizes the result data formats available and how they are set up using the control bits. Table 9-16 illustrates the difference between the signed and unsigned, left justified output codes for an input signal range between 0 and 5.12 Volts.
5 SCAN	**Continuous Conversion Sequence Mode** — This bit selects whether conversion sequences are performed continuously or only once. If external trigger is enabled (ETRIGE=1) setting this bit has no effect, that means each trigger event starts a single conversion sequence. 0 Single conversion sequence 1 Continuous conversion sequences (scan mode)
4 MULT	**Multi-Channel Sample Mode** — When MULT is 0, the ATD sequence controller samples only from the specified analog input channel for an entire conversion sequence. The analog channel is selected by channel selection code (control bits CD/CC/CB/CA located in ATDCTL5). When MULT is 1, the ATD sequence controller samples across channels. The number of channels sampled is determined by the sequence length value (S8C, S4C, S2C, S1C). The first analog channel examined is determined by channel selection code (CC, CB, CA control bits); subsequent channels sampled in the sequence are determined by incrementing the channel selection code or wrapping around to AN0 (channel 0. 0 Sample only one channel 1 Sample across several channels

Table 9-14. ATDCTL5 Field Descriptions (continued)

Field	Description
3:0 C[D:A}	**Analog Input Channel Select Code** — These bits select the analog input channel(s) whose signals are sampled and converted to digital codes. Table 9-17 lists the coding used to select the various analog input channels. In the case of single channel conversions (MULT = 0), this selection code specified the channel to be examined. In the case of multiple channel conversions (MULT = 1), this selection code represents the first channel to be examined in the conversion sequence. Subsequent channels are determined by incrementing the channel selection code or wrapping around to AN0 (after converting the channel defined by the Wrap Around Channel Select Bits WRAP[3:0] in ATDCTL0). In case starting with a channel number higher than the one defined by WRAP[3:0] the first wrap around will be AN15 to AN0.

Table 9-15. Available Result Data Formats.

SRES8	DJM	DSGN	Result Data Formats Description and Bus Bit Mapping
1	0	0	8-bit / left justified / unsigned — bits 15:8
1	0	1	8-bit / left justified / signed — bits 15:8
1	1	X	8-bit / right justified / unsigned — bits 7:0
0	0	0	10-bit / left justified / unsigned — bits 15:6
0	0	1	10-bit / left justified / signed -— bits 15:6
0	1	X	10-bit / right justified / unsigned — bits 9:0

Table 9-16. Left Justified, Signed and Unsigned ATD Output Codes.

Input Signal V_{RL} = 0 Volts V_{RH} = 5.12 Volts	Signed 8-Bit Codes	Unsigned 8-Bit Codes	Signed 10-Bit Codes	Unsigned 10-Bit Codes
5.120 Volts	7F	FF	7FC0	FFC0
5.100	7F	FF	7F00	FF00
5.080	7E	FE	7E00	FE00
2.580	01	81	0100	8100
2.560	00	80	0000	8000
2.540	FF	7F	FF00	7F00
0.020	81	01	8100	0100
0.000	80	00	8000	0000

Table 9-17. Analog Input Channel Select Coding

CD	CC	CB	CA	Analog Input Channel
0	0	0	0	AN0
0	0	0	1	AN1
0	0	1	0	AN2
0	0	1	1	AN3
0	1	0	0	AN4
0	1	0	1	AN5
0	1	1	0	AN6
0	1	1	1	AN7
1	0	0	0	AN8
1	0	0	1	AN9
1	0	1	0	AN10
1	0	1	1	AN11
1	1	0	0	AN12
1	1	0	1	AN13
1	1	1	0	AN14
1	1	1	1	AN15

9.3.2.7 ATD Status Register 0 (ATDSTAT0)

This read-only register contains the Sequence Complete Flag, overrun flags for external trigger and FIFO mode, and the conversion counter.

Module Base + 0x0006

	7	6	5	4	3	2	1	0
R	SCF	0	ETORF	FIFOR	CC3	CC2	CC1	CC0
W		▨			▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 9-9. ATD Status Register 0 (ATDSTAT0)

Read: Anytime

Write: Anytime (No effect on CC[3:0])

Table 9-18. ATDSTAT0 Field Descriptions

Field	Description
7 SCF	**Sequence Complete Flag** — This flag is set upon completion of a conversion sequence. If conversion sequences are continuously performed (SCAN = 1), the flag is set after each one is completed. This flag is cleared when one of the following occurs: • Write “1” to SCF • Write to ATDCTL5 (a new conversion sequence is started) • If AFFC = 1 and read of a result register 0 Conversion sequence not completed 1 Conversion sequence has completed
5 ETORF	**External Trigger Overrun Flag** —While in edge trigger mode (ETRIGLE = 0), if additional active edges are detected while a conversion sequence is in process the overrun flag is set. This flag is cleared when one of the following occurs: • Write “1” to ETORF • Write to ATDCTL0,1,2,3,4 (a conversion sequence is aborted) • Write to ATDCTL5 (a new conversion sequence is started) 0 No External trigger over run error has occurred 1 External trigger over run error has occurred

Table 9-18. ATDSTAT0 Field Descriptions (continued)

Field	Description
4 FIFOR	**FIFO Over Run Flag** — This bit indicates that a result register has been written to before its associated conversion complete flag (CCF) has been cleared. This flag is most useful when using the FIFO mode because the flag potentially indicates that result registers are out of sync with the input channels. However, it is also practical for non-FIFO modes, and indicates that a result register has been over written before it has been read (i.e., the old data has been lost). This flag is cleared when one of the following occurs: • Write “1” to FIFOR • Start a new conversion sequence (write to ATDCTL5 or external trigger) 0 No over run has occurred 1 Overrun condition exists (result register has been written while associated CCFx flag remained set)
3:0 CC[3:0}	**Conversion Counter** — These 4 read-only bits are the binary value of the conversion counter. The conversion counter points to the result register that will receive the result of the current conversion. For example, CC3 = 0, CC2 = 1, CC1 = 1, CC0 = 0 indicates that the result of the current conversion will be in ATD Result Register 6. If in non-FIFO mode (FIFO = 0) the conversion counter is initialized to zero at the begin and end of the conversion sequence. If in FIFO mode (FIFO = 1) the register counter is not initialized. The conversion counters wraps around when its maximum value is reached. Aborting a conversion or starting a new conversion by write to an ATDCTL register (ATDCTL5-0) clears the conversion counter even if FIFO=1.

9.3.2.8 Reserved Register 0 (ATDTEST0)

Module Base + 0x0008

	7	6	5	4	3	2	1	0
R	u	u	u	u	u	u	u	u
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	1	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved u = Unaffected

Figure 9-10. Reserved Register 0 (ATDTEST0)

Read: Anytime, returns unpredictable values

Write: Anytime in special modes, unimplemented in normal modes

NOTE

Writing to this register when in special modes can alter functionality.

9.3.2.9 ATD Test Register 1 (ATDTEST1)

This register contains the SC bit used to enable special channel conversions.

Module Base + 0x0009

	7	6	5	4	3	2	1	0
R	u	u	u	u	u	u	u	SC
W	▨	▨	▨	▨	▨	▨	▨	
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved u = Unaffected

Figure 9-11. Reserved Register 1 (ATDTEST1)

Read: Anytime, returns unpredictable values for bit 7 and bit 6

Write: Anytime

NOTE

Writing to this register when in special modes can alter functionality.

Table 9-19. ATDTEST1 Field Descriptions

Field	Description
0 SC	**Special Channel Conversion Bit** — If this bit is set, then special channel conversion can be selected using CC, CB, and CA of ATDCTL5. Table 9-20 lists the coding. 0 Special channel conversions disabled 1 Special channel conversions enabled

Table 9-20. Special Channel Select Coding

SC	CD	CC	CB	CA	Analog Input Channel
1	0	0	X	X	Reserved
1	0	1	0	0	V_{RH}
1	0	1	0	1	V_{RL}
1	0	1	1	0	$(V_{RH}+V_{RL}) / 2$
1	0	1	1	1	Reserved
1	1	X	X	X	Reserved

9.3.2.10 ATD Status Register 2 (ATDSTAT2)

This read-only register contains the Conversion Complete Flags CCF15 to CCF8.

Module Base + 0x000A

	7	6	5	4	3	2	1	0
R	CCF15	CCF14	CCF13	CCF12	CCF11	CCF10	CCF9	CCF8
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 9-12. ATD Status Register 2 (ATDSTAT2)

Read: Anytime

Write: Anytime, no effect

Table 9-21. ATDSTAT2 Field Descriptions

Field	Description
7:0 CCF[15:8]	**Conversion Complete Flag Bits** — A conversion complete flag is set at the end of each conversion in a conversion sequence. The flags are associated with the conversion position in a sequence (and also the result register number). Therefore, CCF8 is set when the ninth conversion in a sequence is complete and the result is available in result register ATDDR8; CCF9 is set when the tenth conversion in a sequence is complete and the result is available in ATDDR9, and so forth. A flag CCFx (x = 15, 14, 13, 12, 11, 10, 9, 8) is cleared when one of the following occurs: • Write to ATDCTL5 (a new conversion sequence is started) • If AFFC = 0 and read of ATDSTAT2 followed by read of result register ATDDRx • If AFFC = 1 and read of result register ATDDRx In case of a concurrent set and clear on CCFx: The clearing by method A) will overwrite the set. The clearing by methods B) or C) will be overwritten by the set. 0 Conversion number x not completed 1 Conversion number x has completed, result ready in ATDDRx

9.3.2.11 ATD Status Register 1 (ATDSTAT1)

This read-only register contains the Conversion Complete Flags CCF7 to CCF0

Module Base + 0x000B

	7	6	5	4	3	2	1	0
R	CCF7	CCF6	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 9-13. ATD Status Register 1 (ATDSTAT1)

Read: Anytime

Write: Anytime, no effect

Table 9-22. ATDSTAT1 Field Descriptions

Field	Description
7:0 CCF[7:0]	**Conversion Complete Flag Bits** — A conversion complete flag is set at the end of each conversion in a conversion sequence. The flags are associated with the conversion position in a sequence (and also the result register number). Therefore, CCF0 is set when the first conversion in a sequence is complete and the result is available in result register ATDDR0; CCF1 is set when the second conversion in a sequence is complete and the result is available in ATDDR1, and so forth. A CCF flag is cleared when one of the following occurs: • Write to ATDCTL5 (a new conversion sequence is started) • If AFFC = 0 and read of ATDSTAT1 followed by read of result register ATDDRx • If AFFC = 1 and read of result register ATDDRx In case of a concurrent set and clear on CCFx: The clearing by method A) will overwrite the set. The clearing by methods B) or C) will be overwritten by the set. Conversion number x not completed Conversion number x has completed, result ready in ATDDRx

9.3.2.12 ATD Input Enable Register 0 (ATDDIEN0)

Module Base + 0x000C

	7	6	5	4	3	2	1	0
R W	IEN15	IEN14	IEN13	IEN12	IEN11	IEN10	IEN9	IEN8
Reset	0	0	0	0	0	0	0	0

Figure 9-14. ATD Input Enable Register 0 (ATDDIEN0)

Read: Anytime

Write: anytime

Table 9-23. ATDDIEN0 Field Descriptions

Field	Description
7:0 IEN[15:8]	**ATD Digital Input Enable on Channel Bits** — This bit controls the digital input buffer from the analog input pin (ANx) to PTADx data register. 0 Disable digital input buffer to PTADx 1 Enable digital input buffer to PTADx. **Note:** Setting this bit will enable the corresponding digital input buffer continuously. If this bit is set while simultaneously using it as an analog port, there is potentially increased power consumption because the digital input buffer maybe in the linear region.

9.3.2.13 ATD Input Enable Register 1 (ATDDIEN1)

Module Base + 0x000D

	7	6	5	4	3	2	1	0
R W	IEN7	IEN6	IEN5	IEN4	IEN3	IEN2	IEN1	IEN0
Reset	0	0	0	0	0	0	0	0

Figure 9-15. ATD Input Enable Register 1 (ATDDIEN1)

Read: Anytime

Write: Anytime

Table 9-24. ATDDIEN1 Field Descriptions

Field	Description
7:0 IEN[7:0]	**ATD Digital Input Enable on Channel Bits** — This bit controls the digital input buffer from the analog input pin (ANx) to PTADx data register. 0 Disable digital input buffer to PTADx 1 Enable digital input buffer to PTADx. **Note:** Setting this bit will enable the corresponding digital input buffer continuously. If this bit is set while simultaneously using it as an analog port, there is potentially increased power consumption because the digital input buffer maybe in the linear region.

9.3.2.14 Port Data Register 0 (PORTAD0)

The data port associated with the ATD is input-only. The port pins are shared with the analog A/D inputs AN[15:8].

Module Base + 0x000E

	7	6	5	4	3	2	1	0
R	PTAD15	PTAD14	PTAD13	PTAD12	PTAD11	PTAD10	PTAD9	PTAD8
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	1	1	1	1	1	1	1	1
Pin Function	AN15	AN14	AN13	AN12	AN11	AN10	AN9	AN8

▨ = Unimplemented or Reserved

Figure 9-16. Port Data Register 0 (PORTAD0)

Read: Anytime

Write: Anytime, no effect

The A/D input channels may be used for general-purpose digital input.

Table 9-25. PORTAD0 Field Descriptions

Field	Description
7:0 PTAD[15:8]	**A/D Channel x (ANx) Digital Input Bits**— If the digital input buffer on the ANx pin is enabled (IENx = 1) or channel x is enabled as external trigger (ETRIGE = 1, ETRIGCH[3-0] = x, ETRIGSEL = 0) read returns the logic level on ANx pin (signal potentials not meeting V_{IL} or V_{IH} specifications will have an indeterminate value)). If the digital input buffers are disabled (IENx = 0) and channel x is not enabled as external trigger, read returns a “1”. Reset sets all PORTAD0 bits to “1”.

9.3.2.15 Port Data Register 1 (PORTAD1)

The data port associated with the ATD is input-only. The port pins are shared with the analog A/D inputs AN7-0.

Module Base + 0x000F

	7	6	5	4	3	2	1	0
R	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	1	1	1	1	1	1	1	1
Pin Function	AN 7	AN6	AN5	AN4	AN3	AN2	AN1	AN0

▨ = Unimplemented or Reserved

Figure 9-17. Port Data Register 1 (PORTAD1)

Read: Anytime

Write: Anytime, no effect

The A/D input channels may be used for general-purpose digital input.

Table 9-26. PORTAD1 Field Descriptions

Field	Description
7:0 PTAD[7:8]	**A/D Channel x (ANx) Digital Input Bits** — If the digital input buffer on the ANx pin is enabled (IENx=1) or channel x is enabled as external trigger (ETRIGE = 1, ETRIGCH[3-0] = x, ETRIGSEL = 0) read returns the logic level on ANx pin (signal potentials not meeting V_{IL} or V_{IH} specifications will have an indeterminate value)). If the digital input buffers are disabled (IENx = 0) and channel x is not enabled as external trigger, read returns a “1”. Reset sets all PORTAD1 bits to “1”.

9.3.2.16 ATD Conversion Result Registers (ATDDRx)

The A/D conversion results are stored in 16 read-only result registers. The result data is formatted in the result registers bases on two criteria. First there is left and right justification; this selection is made using the DJM control bit in ATDCTL5. Second there is signed and unsigned data; this selection is made using the DSGN control bit in ATDCTL5. Signed data is stored in 2's complement format and only exists in left justified format. Signed data selected for right justified format is ignored.

Read: Anytime

Write: Anytime in special mode, unimplemented in normal modes

9.3.2.16.1 Left Justified Result Data

Module Base +
0x0010 = ATDDR0H
0x0012 = ATDDR1H
0x0014 = ATDDR2H
0x0016 = ATDDR3H
0x0018 = ATDDR4H
0x001A = ATDDR5H
0x001C = ATDDR6H
0x001E = ATDDR7H
0x0020 = ATDDR8H
0x0022 = ATDDR9H
0x0024 = ATDDR10H
0x0026 = ATDDR11H
0x0028 = ATDDR12H
0x002A = ATDDR13H
0x002C = ATDDR14H
0x002E = ATDDR15H

	7	6	5	4	3	2	1	0
R (10-BIT) R (8-BIT)	BIT 9 MSB BIT 7 MSB	BIT 8 BIT 6	BIT 7 BIT 5	BIT 6 BIT 4	BIT 5 BIT 3	BIT 4 BIT 2	BIT 3 BIT 1	BIT 2 BIT 0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 9-18. Left Justified, ATD Conversion Result Register x, High Byte (ATDDRxH)

Module Base +
0x0011 = ATDDR0L
0x0013 = ATDDR1L
0x0015 = ATDDR2L
0x0017 = ATDDR3L
0x0019 = ATDDR4L
0x001B = ATDDR5L
0x001D = ATDDR6L
0x001F = ATDDR7L
0x0021 = ATDDR8L
0x0023 = ATDDR9L
0x0025 = ATDDR10L
0x0027 = ATDDR11L
0x0029 = ATDDR12L
0x002B = ATDDR13L
0x002D = ATDDR14L
0x002F = ATDDR15L

	7	6	5	4	3	2	1	0
R (10-BIT) R (8-BIT)	BIT 1 u	BIT 0 u	0 0	0 0	0 0	0 0	0 0	0 0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

u = Unaffected

Figure 9-19. Left Justified, ATD Conversion Result Register x, Low Byte (ATDDRxL)

9.3.2.16.2 Right Justified Result Data

Module Base + 0x0010 = ATDDR0H,
0x0012 = ATDDR1H,
0x0014 = ATDDR2H,
0x0016 = ATDDR3H
0x0018 = ATDDR4H,
0x001A = ATDDR5H,
0x001C = ATDDR6H,
0x001E = ATDDR7H
0x0020 = ATDDR8H,
0x0022 = ATDDR9H,
0x0024 = ATDDR10H,
0x0026 = ATDDR11H
0x0028 = ATDDR12H,
0x002A = ATDDR13H,
0x002C = ATDDR14H,
0x002E = ATDDR15H

	7	6	5	4	3	2	1	0
R (10-BIT) R (8-BIT)	0 0	0 0	0 0	0 0	0 0	0 0	BIT 9 MSB 0	BIT 8 0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 9-20. Right Justified, ATD Conversion Result Register x, High Byte (ATDDRxH)

Module Base + 0x0011 = ATDDR0L,
0x0013 = ATDDR1L,
0x0015 = ATDDR2L,
0x0017 = ATDDR3L
0x0019 = ATDDR4L,
0x001B = ATDDR5L,
0x001D = ATDDR6L,
0x001F = ATDDR7L
0x0021 = ATDDR8L,
0x0023 = ATDDR9L,
0x0025 = ATDDR10L,
0x0027 = ATDDR11L
0x0029 = ATDDR12L,
0x002B = ATDDR13L,
0x002D = ATDDR14L,
0x002F = ATDDR15L

	7	6	5	4	3	2	1	0
R (10-BIT) R (8-BIT)	BIT 7 BIT 7 MSB	BIT 6 BIT 6	BIT 5 BIT 5	BIT 4 BIT 4	BIT 3 BIT 3	BIT 2 BIT 2	BIT 1 BIT 1	BIT 0 BIT 0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 9-21. Right Justified, ATD Conversion Result Register x, Low Byte (ATDDRxL)

9.4 Functional Description

The ATD10B16C is structured in an analog and a digital sub-block.

9.4.1 Analog Sub-block

The analog sub-block contains all analog electronics required to perform a single conversion. Separate power supplies V_{DDA} and V_{SSA} allow to isolate noise of other MCU circuitry from the analog sub-block.

9.4.1.1 Sample and Hold Machine

The sample and hold (S/H) machine accepts analog signals from the external world and stores them as capacitor charge on a storage node.

The sample process uses a two stage approach. During the first stage, the sample amplifier is used to quickly charge the storage node.The second stage connects the input directly to the storage node to complete the sample for high accuracy.

When not sampling, the sample and hold machine disables its own clocks. The analog electronics continue drawing their quiescent current. The power down (ADPU) bit must be set to disable both the digital clocks and the analog power consumption.

The input analog signals are unipolar and must fall within the potential range of V_{SSA} to VDDA.

9.4.1.2 Analog Input Multiplexer

The analog input multiplexer connects one of the 16 external analog input channels to the sample and hold machine.

9.4.1.3 Sample Buffer Amplifier

The sample amplifier is used to buffer the input analog signal so that the storage node can be quickly charged to the sample potential.

9.4.1.4 Analog-to-Digital (A/D) Machine

The A/D machine performs analog to digital conversions. The resolution is program selectable at either 8 or 10 bits. The A/D machine uses a successive approximation architecture. It functions by comparing the stored analog sample potential with a series of digitally generated analog potentials. By following a binary search algorithm, the A/D machine locates the approximating potential that is nearest to the sampled potential.

When not converting the A/D machine disables its own clocks. The analog electronics continue drawing quiescent current. The power down (ADPU) bit must be set to disable both the digital clocks and the analog power consumption.

Only analog input signals within the potential range of V_{RL} to V_{RH} (A/D reference potentials) will result in a non-railed digital output codes.

9.4.2 Digital Sub-Block

This subsection explains some of the digital features in more detail. See register descriptions for all details.

9.4.2.1 External Trigger Input

The external trigger feature allows the user to synchronize ATD conversions to the external environment events rather than relying on software to signal the ATD module when ATD conversions are to take place. The external trigger signal (out of reset ATD channel 15, configurable in ATDCTL1) is programmable to be edge or level sensitive with polarity control. Table 9-27 gives a brief description of the different combinations of control bits and their effect on the external trigger function.

Table 9-27. External Trigger Control Bits

ETRIGLE	ETRIGP	ETRIGE	SCAN	Description
X	X	0	0	Ignores external trigger. Performs one conversion sequence and stops.
X	X	0	1	Ignores external trigger. Performs continuous conversion sequences.
0	0	1	X	Falling edge triggered. Performs one conversion sequence per trigger.
0	1	1	X	Rising edge triggered. Performs one conversion sequence per trigger.
1	0	1	X	Trigger active low. Performs continuous conversions while trigger is active.
1	1	1	X	Trigger active high. Performs continuous conversions while trigger is active.

During a conversion, if additional active edges are detected the overrun error flag ETORF is set.

In either level or edge triggered modes, the first conversion begins when the trigger is received. In both cases, the maximum latency time is one bus clock cycle plus any skew or delay introduced by the trigger circuitry.

After ETRIGE is enabled, conversions cannot be started by a write to ATDCTL5, but rather must be triggered externally.

If the level mode is active and the external trigger both de-asserts and re-asserts itself during a conversion sequence, this does not constitute an overrun. Therefore, the flag is not set. If the trigger remains asserted in level mode while a sequence is completing, another sequence will be triggered immediately.

9.4.2.2 General-Purpose Digital Input Port Operation

The input channel pins can be multiplexed between analog and digital data. As analog inputs, they are multiplexed and sampled to supply signals to the A/D converter. As digital inputs, they supply external input data that can be accessed through the digital port registers (PORTAD0 & PORTAD1) (input-only).

The analog/digital multiplex operation is performed in the input pads. The input pad is always connected to the analog inputs of the ATD10B16C. The input pad signal is buffered to the digital port registers. This buffer can be turned on or off with the ATDDIEN0 & ATDDIEN1 register. This is important so that the buffer does not draw excess current when analog potentials are presented at its input.

9.4.3 Operation in Low Power Modes

The ATD10B16C can be configured for lower MCU power consumption in three different ways:

- **Stop Mode**
  Stop Mode: This halts A/D conversion. Exit from Stop mode will resume A/D conversion, But due to the recovery time the result of this conversion should be ignored.

  Entering stop mode causes all clocks to halt and thus the system is placed in a minimum power standby mode. This halts any conversion sequence in progress. During recovery from stop mode, there must be a minimum delay for the stop recovery time t_{SR} before initiating a new ATD conversion sequence.
- **Wait Mode**
  Wait Mode with AWAI = 1: This halts A/D conversion. Exit from Wait mode will resume A/D conversion, but due to the recovery time the result of this conversion should be ignored.

Entering wait mode, the ATD conversion either continues or halts for low power depending on the logical value of the AWAIT bit.

- **Freeze Mode**
  Writing ADPU = 0 (Note that all ATD registers remain accessible.): This aborts any A/D conversion in progress.

  In freeze mode, the ATD10B16C will behave according to the logical values of the FRZ1 and FRZ0 bits. This is useful for debugging and emulation.

NOTE

The reset value for the ADPU bit is zero. Therefore, when this module is reset, it is reset into the power down state.

9.5 Initialization/Application Information

9.5.1 Setting up and starting an A/D conversion

The following describes a typical setup procedure for starting A/D conversions. It is highly recommended to follow this procedure to avoid common mistakes.

Each step of the procedure will have a general remark and a typical example

9.5.1.1 Step 1

Power up the ATD and concurrently define other settings in ATDCTL2
Example: Write to ATDCTL2: ADPU=1 -> powers up the ATD, ASCIE=1 enable interrupt on finish of a conversion sequence.

9.5.1.2 Step 2

Wait for the ATD Recovery Time t_{REC} before you proceed with Step 3.

Example: Use the CPU in a branch loop to wait for a defined number of bus clocks.

9.5.1.3 Step 3

Configure how many conversions you want to perform in one sequence and define other settings in ATDCTL3.

Example: Write S4C=1 to do 4 conversions per sequence.

9.5.1.4 Step 4

Configure resolution, sampling time and ATD clock speed in ATDCTL4.

Example: Use default for resolution and sampling time by leaving SRES8, SMP1 and SMP0 clear. For a bus clock of 40MHz write 9 to PR4-0, this gives an ATD clock of 0.5*40MHz/(9+1) = 2MHz which is within the allowed range for f_{ATDCLK}.

9.5.1.5 Step 5

Configure starting channel, single/multiple channel, continuous or single sequence and result data format in ATDCTL5. Writing ATDCTL5 will start the conversion, so make sure your write ATDCTL5 in the last step.

Example: Leave CD, CC,CB,CA clear to start on channel AN0. Write MULT=1 to convert channel AN0 to AN3 in a sequence (4 conversion per sequence selected in ATDCTL3).

9.5.2 Aborting an A/D conversion

9.5.2.1 Step 1

Write to ATDCTL4. This will abort any ongoing conversion sequence.

(Do not use write to other ATDCTL registers to abort, as this under certain circumstances might not work correctly.)

9.5.2.2 Step 2

Disable the ATD Interrupt by writing ASCIE=0 in ATDCTL2.

It is important to clear the interrupt enable at this point, prior to step 3, as depending on the device clock gating it may not always be possible to clear it or the SCF flag once the module is disabled (ADPU=0).

9.5.2.3 Step 3

Clear the SCF flag by writing a 1 in ATDSTAT0.

(Remaining flags will be cleared with the next start of a conversions, but SCF flag should be cleared to avoid SCF interrupt.)

9.5.2.4 Step 4

Power down ATD by writing ADPU=0 in ATDCTL2.

9.6 Resets

At reset the ATD10B16C is in a power down state. The reset state of each individual bit is listed within Section 9.3, “Memory Map and Register Definition,” which details the registers and their bit fields.

9.7 Interrupts

The interrupt requested by the ATD10B16C is listed in Table 9-28. Refer to MCU specification for related vector address and priority.

Table 9-28. ATD Interrupt Vectors

Interrupt Source	CCR Mask	Local Enable
Sequence Complete Interrupt	I bit	ASCIE in ATDCTL2

See Section 9.3.2, “Register Descriptions,” for further details.

Chapter 10
Liquid Crystal Display (LCD32F4BV1) Block Description

10.1 Introduction

The LCD32F4BV1 driver module has 32 frontplane drivers and 4 backplane drivers so that a maximum of 128 LCD segments are controllable. Each segment is controlled by a corresponding bit in the LCD RAM. Four multiplex modes (1/1, 1/2, 1/3, 1/4 duty), and three bias (1/1, 1/2, 1/3) methods are available. The V_0 voltage is the lowest level of the output waveform and V_3 becomes the highest level. All frontplane and backplane pins can be multiplexed with other port functions.

The LCD32F4BV1 driver system consists of five major sub-modules:

- Timing and Control – consists of registers and control logic for frame clock generation, bias voltage level select, frame duty select, backplane select, and frontplane select/enable to produce the required frame frequency and voltage waveforms.
- LCD RAM – contains the data to be displayed on the LCD. Data can be read from or written to the display RAM at any time.
- Frontplane Drivers – consists of 32 frontplane drivers.
- Backplane Drivers – consists of 4 backplane drivers.
- Voltage Generator – Based on voltage applied to VLCD, it generates the voltage levels for the timing and control logic to produce the frontplane and backplane waveforms.

10.1.1 Features

The LCD32F4BV1 includes these distinctive features:

- Supports five LCD operation modes
- 32 frontplane drivers
- 4 backplane drivers
  - Each frontplane has an enable bit respectively
- Programmable frame clock generator
- Programmable bias voltage level selector
- On-chip generation of 4 different output voltage levels

10.1.2 Modes of Operation

The LCD32F4BV1 module supports five operation modes with different numbers of backplanes and different biasing levels. During pseudo stop mode and wait mode the LCD operation can be suspended under software control. Depending on the state of internal bits, the LCD can operate normally or the LCD clock generation can be turned off and the LCD32F4BV1 module enters a power conservation state.

This is a high level description only, detailed descriptions of operating modes are contained in Section 10.4.2, “Operation in Wait Mode”, Section 10.4.3, “Operation in Pseudo Stop Mode”, and Section 10.4.4, “Operation in Stop Mode”.

10.1.3 Block Diagram

Figure 10-1 is a block diagram of the LCD32F4BV1 module.

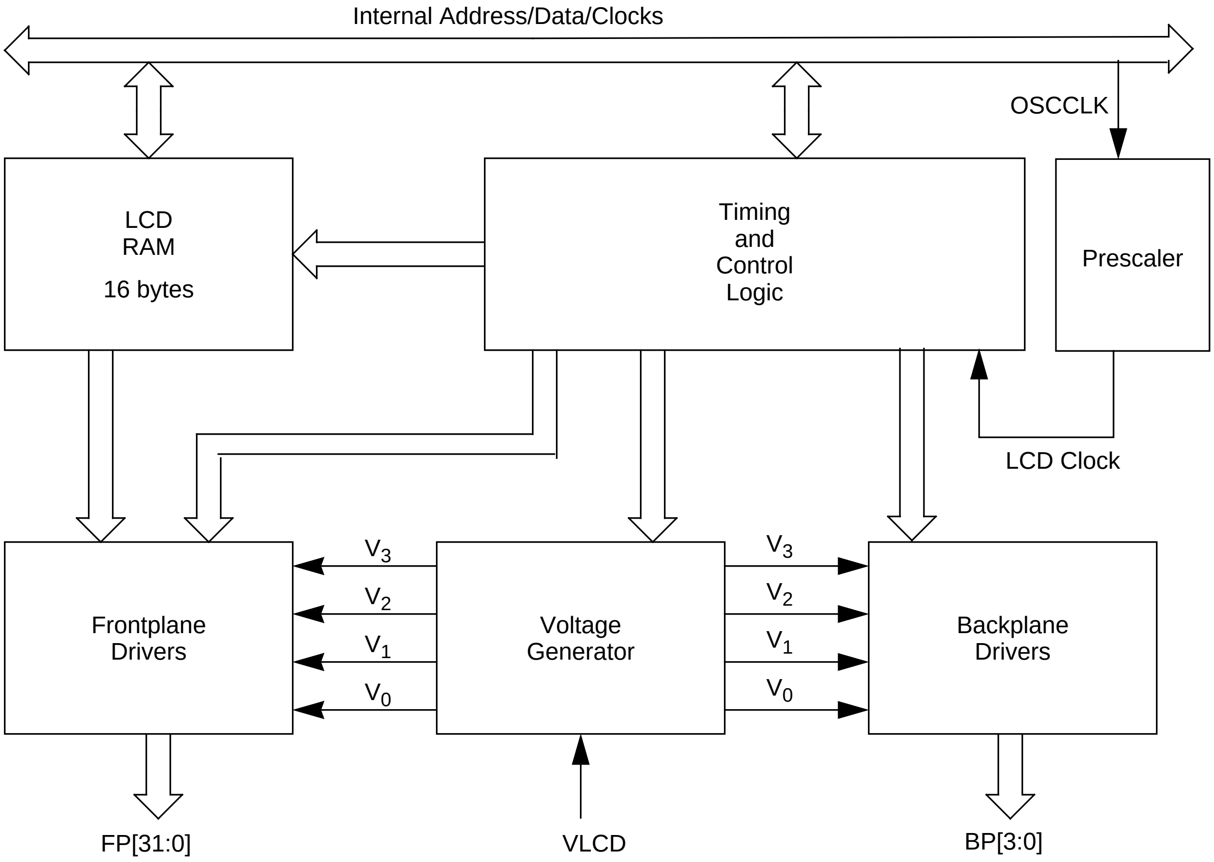


Figure 10-1. LCD32F4BV1 Block Diagram

10.2 External Signal Description

The LCD32F4BV1 module has a total of 37 external pins.

Table 10-1. Signal Properties

Name	Port	Function	Reset State
4 backplane waveforms	BP[3:0]	Backplane waveform signals that connect directly to the pads	High impedance
32 frontplane waveforms	FP[31:0]	Frontplane waveform signals that connect directly to the pads	High impedance
LCD voltage	VLCD	LCD supply voltage	—

10.2.1 BP[3:0] — Analog Backplane Pins

This output signal vector represents the analog backplane waveforms of the LCD32F4BV1 module and is connected directly to the corresponding pads.

10.2.2 FP[31:0] — Analog Frontplane Pins

This output signal vector represents the analog frontplane waveforms of the LCD32F4BV1 module and is connected directly to the corresponding pads.

10.2.3 VLCD — LCD Supply Voltage Pin

Positive supply voltage for the LCD waveform generation.

10.3 Memory Map and Register Definition

This section provides a detailed description of all memory and registers.

10.3.1 Module Memory Map

The memory map for the LCD32F4BV1 module is given in Table 10-2. The address listed for each register is the address offset. The total address for each register is the sum of the base address for the LCD32F4BV1 module and the address offset for each register.

Table 10-2. LCD32F4BV1 Memory Map

Address Offset	Use	Access
0x0000	LCD Control Register 0 (LCDCR0)	Read/Write
0x0001	LCD Control Register 1 (LCDCR1)	Read/Write
0x0002	LCD Frontplane Enable Register 0 (FPENR0)	Read/Write
0x0003	LCD Frontplane Enable Register 1 (FPENR1)	Read/Write
0x0004	LCD Frontplane Enable Register 2 (FPENR2)	Read/Write
0x0005	LCD Frontplane Enable Register 3 (FPENR3)	Read/Write
0x0006	Unimplemented	
0x0007	Unimplemented	
0x0008	LCDRAM (Location 0)	Read/Write
0x0009	LCDRAM (Location 1)	Read/Write
0x000A	LCDRAM (Location 2)	Read/Write
0x000B	LCDRAM (Location 3)	Read/Write
0x000C	LCDRAM (Location 4)	Read/Write
0x000D	LCDRAM (Location 5)	Read/Write
0x000E	LCDRAM (Location 6)	Read/Write
0x000F	LCDRAM (Location 7)	Read/Write
0x0010	LCDRAM (Location 8)	Read/Write
0x0011	LCDRAM (Location 9)	Read/Write
0x0012	LCDRAM (Location 10)	Read/Write
0x0013	LCDRAM (Location 11)	Read/Write
0x0014	LCDRAM (Location 12)	Read/Write
0x0015	LCDRAM (Location 13)	Read/Write
0x0016	LCDRAM (Location 14)	Read/Write
0x0017	LCDRAM (Location 15)	Read/Write

10.3.2 Register Descriptions

This section consists of register descriptions. Each description includes a standard register diagram. Details of register bit and field function follow the register diagrams, in bit order.

10.3.2.1 LCD Control Register 0 (LCDCR0)

Module Base + 0x0000

	7	6	5	4	3	2	1	0
R	LCDEN	0	LCLK2	LCLK1	LCLK0	BIAS	DUTY1	DUTY0
W		▨						
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 10-2. LCD Control Register 0 (LCDCR0)

Read: anytime

Write: LCDEN anytime. To avoid segment flicker the clock prescaler bits, the bias select bit and the duty select bits must not be changed when the LCD is enabled.

Table 10-3. LCDCR0 Field Descriptions

Field	Description
7 LCDEN	**LCD32F4BV1 Driver System Enable** — The LCDEN bit starts the LCD waveform generator. 0 All frontplane and backplane pins are disabled. In addition, the LCD32F4BV1 system is disabled and all LCD waveform generation clocks are stopped. 1 LCD driver system is enabled. All FP[31:0] pins with FP[31:0]EN set, will output an LCD driver waveform The BP[3:0] pins will output an LCD32F4BV1 driver waveform based on the settings of DUTY0 and DUTY1.
5:3 LCLK[2:0]	**LCD Clock Prescaler** — The LCD clock prescaler bits determine the OSCCLK divider value to produce the LCD clock frequency. For detailed description of the correlation between LCD clock prescaler bits and the divider value please refer to Table 10-7.
2 BIAS	**BIAS Voltage Level Select** — This bit selects the bias voltage levels during various LCD operating modes, as shown in Table 10-8.
1:0 DUTY[1:0]	**LCD Duty Select** — The DUTY1 and DUTY0 bits select the duty (multiplex mode) of the LCD32F4BV1 driver system, as shown in Table 10-8.

10.3.2.2 LCD Control Register 1 (LCDCR1)

Module Base + 0x0001

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	LCDSWAI	LCDRPSTP
W	▨	▨	▨	▨	▨	▨		
Reset	0	0	0	0	0	0	0	0

▨ Unimplemented or Reserved

Figure 10-3. LCD Control Register 1 (LCDCR1)

Read: anytime

Write: anytime

Table 10-4. LCDCR1 Field Descriptions

Field	Description
1 LCDSWAI	**LCD Stop in Wait Mode** — This bit controls the LCD operation while in wait mode. 0 LCD operates normally in wait mode. 1 Stop LCD32F4BV1 driver system when in wait mode.
0 LCDRPSTP	**LCD Run in Pseudo Stop Mode** — This bit controls the LCD operation while in pseudo stop mode. 0 Stop LCD32F4BV1 driver system when in pseudo stop mode. 1 LCD operates normally in pseudo stop mode.

10.3.2.3 LCD Frontplane Enable Register 0–3 (FPENR0–FPENR3)

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R W	FP7EN	FP6EN	FP5EN	FP4EN	FP3EN	FP2EN	FP1EN	FP0EN
Reset	0	0	0	0	0	0	0	0

Figure 10-4. LCD Frontplane Enable Register 0 (FPENR0)

Module Base + 0x0003

	7	6	5	4	3	2	1	0
R W	FP15EN	FP14EN	FP13EN	FP12EN	FP11EN	FP10EN	FP9EN	FP8EN
Reset	0	0	0	0	0	0	0	0

Figure 10-5. LCD Frontplane Enable Register 1 (FPENR1)

Module Base + 0x0004

	7	6	5	4	3	2	1	0
R W	FP23EN	FP22EN	FP21EN	FP20EN	FP19EN	FP18EN	FP17EN	FP16EN
Reset	0	0	0	0	0	0	0	0

Figure 10-6. LCD Frontplane Enable Register 2 (FPENR2)

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R W	FP31EN	FP30EN	FP29EN	FP28EN	FP27EN	FP26EN	FP25EN	FP24EN
Reset	0	0	0	0	0	0	0	0

Figure 10-7. LCD Frontplane Enable Register 3 (FPENR3)

These bits enable the frontplane output waveform on the corresponding frontplane pin when LCDEN = 1.

Read: anytime

Write: anytime

Table 10-5. FPENR0–FPENR3 Field Descriptions

Field	Description
31:0 FP[31:0]EN	**Frontplane Output Enable** — The FP[31:0]EN bit enables the frontplane driver outputs. If LCDEN = 0, these bits have no effect on the state of the I/O pins. It is recommended to set FP[31:0]EN bits before LCDEN is set. 0 Frontplane driver output disabled on FP[31:0]. 1 Frontplane driver output enabled on FP[31:0].

10.3.2.4 LCD RAM (LCDRAM)

The LCD RAM consists of 16 bytes. After reset the LCD RAM contents will be indeterminate (I), as indicated by Figure 10-8.

		7	6	5	4	3	2	1	0
0x0008 LCDRAM	R W	FP1BP3	FP1BP2	FP1BP1	FP1BP0	FP0BP3	FP0BP2	FP0BP1	FP0BP0
	Reset	I	I	I	I	I	I	I	I
0x0009 LCDRAM	R W	FP3BP3	FP3BP2	FP3BP1	FP3BP0	FP2BP3	FP2BP2	FP2BP1	FP2BP0
	Reset	I	I	I	I	I	I	I	I
0x000A LCDRAM	R W	FP5BP3	FP5BP2	FP5BP1	FP5BP0	FP4BP3	FP4BP2	FP4BP1	FP4BP0
	Reset	I	I	I	I	I	I	I	I

I = Value is indeterminate

Figure 10-8. LCD RAM (LCDRAM)

0x000B	R	FP7BP3	FP7BP2	FP7BP1	FP7BP0	FP6BP3	FP6BP2	FP6BP1	FP6BP0
LCDRAM	W								
	Reset	I	I	I	I	I	I	I	I
0x000C	R	FP9BP3	FP9BP2	FP9BP1	FP9BP0	FP8BP3	FP8BP2	FP8BP1	FP8BP0
LCDRAM	W								
	Reset	I	I	I	I	I	I	I	I
0x000D	R	FP11BP3	FP11BP2	FP11BP1	FP11BP0	FP10BP3	FP10BP2	FP10BP1	FP10BP0
LCDRAM	W								
	Reset	I	I	I	I	I	I	I	I
0x000E	R	FP13BP3	FP13BP2	FP13BP1	FP13BP0	FP12BP3	FP12BP2	FP12BP1	FP12BP0
LCDRAM	W								
	Reset	I	I	I	I	I	I	I	I
0x000F	R	FP15BP3	FP15BP2	FP15BP1	FP15BP0	FP14BP3	FP14BP2	FP14BP1	FP14BP0
LCDRAM	W								
	Reset	I	I	I	I	I	I	I	I
0x0010	R	FP17BP3	FP17BP2	FP17BP1	FP17BP0	FP16BP3	FP16BP2	FP16BP1	FP16BP0
LCDRAM	W								
	Reset	I	I	I	I	I	I	I	I
0x0011	R	FP19BP3	FP19BP2	FP19BP1	FP19BP0	FP18BP3	FP18BP2	FP18BP1	FP18BP0
LCDRAM	W								
	Reset	I	I	I	I	I	I	I	I
0x0012	R	FP21BP3	FP21BP2	FP21BP1	FP21BP0	FP20BP3	FP20BP2	FP20BP1	FP20BP0
LCDRAM	W								
	Reset	I	I	I	I	I	I	I	I
0x0013	R	FP23BP3	FP23BP2	FP23BP1	FP23BP0	FP22BP3	FP22BP2	FP22BP1	FP22BP0
LCDRAM	W								
	Reset	I	I	I	I	I	I	I	I
0x0014	R	FP25BP3	FP25BP2	FP25BP1	FP25BP0	FP24BP3	FP24BP2	FP24BP1	FP24BP0
LCDRAM	W								
	Reset	I	I	I	I	I	I	I	I
0x0015	R	FP27BP3	FP27BP2	FP27BP1	FP27BP0	FP26BP3	FP26BP2	FP26BP1	FP26BP0
LCDRAM	W								
	Reset	I	I	I	I	I	I	I	I
0x0016	R	FP29BP3	FP29BP2	FP29BP1	FP29BP0	FP28BP3	FP28BP2	FP28BP1	FP28BP0
LCDRAM	W								
	Reset	I	I	I	I	I	I	I	I
0x0017	R	FP31BP3	FP31BP2	FP31BP1	FP31BP0	FP30BP3	FP30BP2	FP30BP1	FP30BP0
LCDRAM	W								
	Reset	I	I	I	I	I	I	I	I

I = Value is indeterminate

Figure 10-8. LCD RAM (LCDRAM) (continued)

Read: anytime

Write: anytime

Table 10-6. LCD RAM Field Descriptions

Field	Description
31:0 3:0 FP[31:0] BP[3:0]	**LCD Segment ON** — The FP[31:0]BP[3:0] bit displays (turns on) the LCD segment connected between FP[31:0] and BP[3:0]. 0 LCD segment OFF 1 LCD segment ON

10.4 Functional Description

This section provides a complete functional description of the LCD32F4BV1 block, detailing the operation of the design from the end user perspective in a number of subsections.

10.4.1 LCD Driver Description

10.4.1.1 Frontplane, Backplane, and LCD System During Reset

During a reset the following conditions exist:

- The LCD32F4BV1 system is configured in the default mode, 1/4 duty and 1/3 bias, that means all backplanes are used.
- All frontplane enable bits, FP[31:0]EN are cleared and the ON/OFF control for the display, the LCDEN bit is cleared, thereby forcing all frontplane and backplane driver outputs to the high impedance state. The MCU pin state during reset is defined by the port integration module (PIM).

10.4.1.2 LCD Clock and Frame Frequency

The frequency of the oscillator clock (OSCCLK) and divider determine the LCD clock frequency. The divider is set by the LCD clock prescaler bits, LCLK[2:0], in the LCD control register 0 (LCDCR0). Table 10-7 shows the LCD clock and frame frequency for some multiplexed mode at OSCCLK = 16 MHz, 8 MHz, 4 MHz, 2 MHz, 1 MHz, and 0.5 MHz.

Table 10-7. LCD Clock and Frame Frequency

Oscillator Frequency in MHz	LCD Clock Prescaler			Divider	LCD Clock Frequency [Hz]	Frame Frequency [Hz]			
	LCLK2	LCLK1	LCLK0			1/1 Duty	1/2 Duty	1/3 Duty	1/4 Duty
OSCCLK = 0.5	0 0	0 0	0 1	1024 2048	488 244	488 244	244 122	163 81	122 61
OSCCLK = 1.0	0 0	0 1	1 0	2048 4096	488 244	488 244	244 122	163 81	122 61
OSCCLK = 2.0	0 0	1 1	0 1	4096 8192	488 244	488 244	244 122	163 81	122 61

Table 10-7. LCD Clock and Frame Frequency

Oscillator Frequency in MHz	LCD Clock Prescaler			Divider	LCD Clock Frequency [Hz]	Frame Frequency [Hz]			
	LCLK2	LCLK1	LCLK0			1/1 Duty	1/2 Duty	1/3 Duty	1/4 Duty
OSCCLK = 4.0	0 1	1 0	1 0	8192 16384	488 244	488 244	244 122	163 81	122 61
OSCCLK = 8.0	1 1	0 0	0 1	16384 32768	488 244	488 244	244 122	163 81	122 61
OSCCLK = 16.0	1 1	1 1	0 1	65536 131072	244 122	244 122	122 61	81 40	61 31

For other combinations of OSCCLK and divider not shown in Table 10-7, the following formula may be used to calculate the LCD frame frequency for each multiplex mode:

$$\text{LCD Frame Frequency (Hz)} = \left[\frac{\text{OSCCLK (Hz)}}{\text{Divider}}\right] \cdot \text{Duty}$$

The possible divider values are shown in Table 10-7.

10.4.1.3 LCD RAM

For a segment on the LCD to be displayed, data must be written to the LCD RAM which is shown in Section 10.3, "Memory Map and Register Definition". The 128 bits in the LCD RAM correspond to the 128 segments that are driven by the frontplane and backplane drivers. Writing a 1 to a given location will result in the corresponding display segment being driven with a differential RMS voltage necessary to turn the segment ON when the LCDEN bit is set and the corresponding FP[31:0]EN bit is set. Writing a 0 to a given location will result in the corresponding display segment being driven with a differential RMS voltage necessary to turn the segment OFF. The LCD RAM is a dual port RAM that interfaces with the internal address and data buses of the MCU. It is possible to read from LCD RAM locations for scrolling purposes. When LCDEN = 0, the LCD RAM can be used as on-chip RAM. Writing or reading of the LCDEN bit does not change the contents of the LCD RAM. After a reset, the LCD RAM contents will be indeterminate.

10.4.1.4 LCD Driver System Enable and Frontplane Enable Sequencing

If LCDEN = 0 (LCD32F4BV1 driver system disabled) and the frontplane enable bit, FP[31:0]EN, is set, the frontplane driver waveform will not appear on the output until LCDEN is set. If LCDEN = 1 (LCD32F4BV1 driver system enabled), the frontplane driver waveform will appear on the output as soon as the corresponding frontplane enable bit, FP[31:0]EN, in the registers FPENR0–FPENR3 is set.

10.4.1.5 LCD Bias and Modes of Operation

The LCD32F4BV1 driver has five modes of operation:

- 1/1 duty (1 backplane), 1/1 bias (2 voltage levels)
- 1/2 duty (2 backplanes), 1/2 bias (3 voltage levels)
- 1/2 duty (2 backplanes), 1/3 bias (4 voltage levels)

- 1/3 duty (3 backplanes), 1/3 bias (4 voltage levels)
- 1/4 duty (4 backplanes), 1/3 bias (4 voltage levels)

The voltage levels required for the different operating modes are generated internally based on VLCD. Changing VLCD alters the differential RMS voltage across the segments in the ON and OFF states, thereby setting the display contrast.

The backplane waveforms are continuous and repetitive every frame. They are fixed within each operating mode and are not affected by the data in the LCD RAM.

The frontplane waveforms generated are dependent on the state (ON or OFF) of the LCD segments as defined in the LCD RAM. The LCD32F4BV1 driver hardware uses the data in the LCD RAM to construct the frontplane waveform to create a differential RMS voltage necessary to turn the segment ON or OFF.

The LCD duty is decided by the DUTY1 and DUTY0 bits in the LCD control register 0 (LCDCR0). The number of bias voltage levels is determined by the BIAS bit in LCDCR0. Table 10-8 summarizes the multiplex modes (duties) and the bias voltage levels that can be selected for each multiplex mode (duty). The backplane pins have their corresponding backplane waveform output BP[3:0] in high impedance state when in the OFF state as indicated in Table 10-8. In the OFF state the corresponding pins BP[3:0]can be used for other functionality, for example as general purpose I/O ports.

Table 10-8. LCD Duty and Bias

Duty	LCDCR0 Register		Backplanes				Bias (BIAS = 0)			Bias (BIAS = 1)		
	DUTY1	DUTY0	BP3	BP2	BP1	BP0	1/1	1/2	1/3	1/1	1/2	1/3
1/1	0	1	OFF	OFF	OFF	BP0	YES	NA	NA	YES	NA	NA
1/2	1	0	OFF	OFF	BP1	BP0	NA	YES	NA	NA	NA	YES
1/3	1	1	OFF	BP2	BP1	BP0	NA	NA	YES	NA	NA	YES
1/4	0	0	BP3	BP2	BP1	BP0	NA	NA	YES	NA	NA	YES

10.4.2 Operation in Wait Mode

The LCD32F4BV1 driver system operation during wait mode is controlled by the LCD stop in wait (LCDSWAI) bit in the LCD control register 1 (LCDCR1). If LCDSWAI is reset, the LCD32F4BV1 driver system continues to operate during wait mode. If LCDSWAI is set, the LCD32F4BV1 driver system is turned off during wait mode. In this case, the LCD waveform generation clocks are stopped and the LCD32F4BV1 drivers pull down to VSSX those frontplane and backplane pins that were enabled before entering wait mode. The contents of the LCD RAM and the LCD registers retain the values they had prior to entering wait mode.

10.4.3 Operation in Pseudo Stop Mode

The LCD32F4BV1 driver system operation during pseudo stop mode is controlled by the LCD run in pseudo stop (LCDRPSTP) bit in the LCD control register 1 (LCDCR1). If LCDRPSTP is reset, the LCD32F4BV1 driver system is turned off during pseudo stop mode. In this case, the LCD waveform generation clocks are stopped and the LCD32F4BV1 drivers pull down to VSSX those frontplane and backplane pins that were enabled before entering pseudo stop mode. If LCDRPSTP is set, the

LCD32F4BV1 driver system continues to operate during pseudo stop mode. The contents of the LCD RAM and the LCD registers retain the values they had prior to entering pseudo stop mode.

10.4.4 Operation in Stop Mode

All LCD32F4BV1 driver system clocks are stopped, the LCD32F4BV1 driver system pulls down to VSSX those frontplane and backplane pins that were enabled before entering stop mode. Also, during stop mode, the contents of the LCD RAM and the LCD registers retain the values they had prior to entering stop mode. As a result, after exiting from stop mode, the LCD32F4BV1 driver system clocks will run (if LCDEN = 1) and the frontplane and backplane pins retain the functionality they had prior to entering stop mode.

10.4.5 LCD Waveform Examples

Figure 10-9 through Figure 10-13 show the timing examples of the LCD output waveforms for the available modes of operation.

10.4.5.1 1/1 Duty Multiplexed with 1/1 Bias Mode

Duty = 1/1:DUTY1 = 0, DUTY0 = 1

Bias = 1/1:BIAS = 0 or BIAS = 1

$V_0 = V_1 = \text{VSSX}, V_2 = V_3 = \text{VLCD}$

- BP1, BP2, and BP3 are not used, a maximum of 32 segments are displayed.

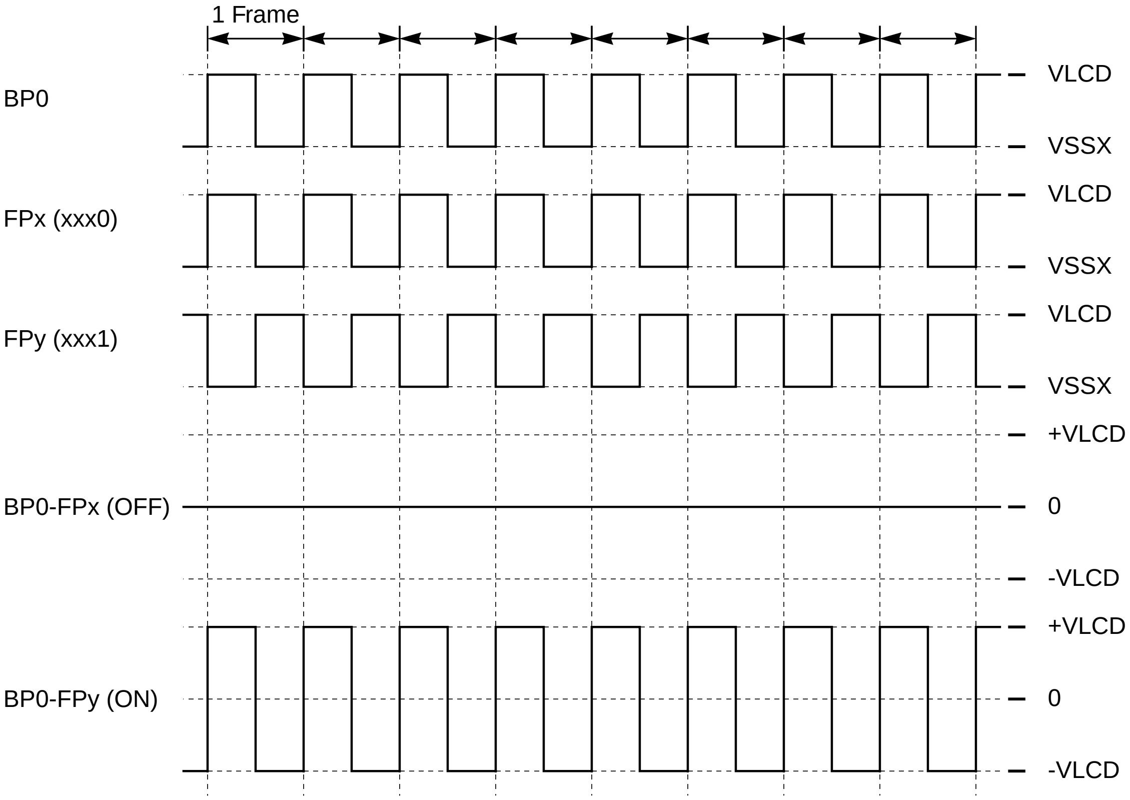


Figure 10-9. 1/1 Duty and 1/1 Bias

10.4.5.2 1/2 Duty Multiplexed with 1/2 Bias Mode

Duty = 1/2:DUTY1 = 1, DUTY0 = 0

Bias = 1/2:BIAS = 0

V_0 = VSSX, V_1 = V_2 = VLCD * 1/2, V_3 = VLCD

- BP2 and BP3 are not used, a maximum of 64 segments are displayed.

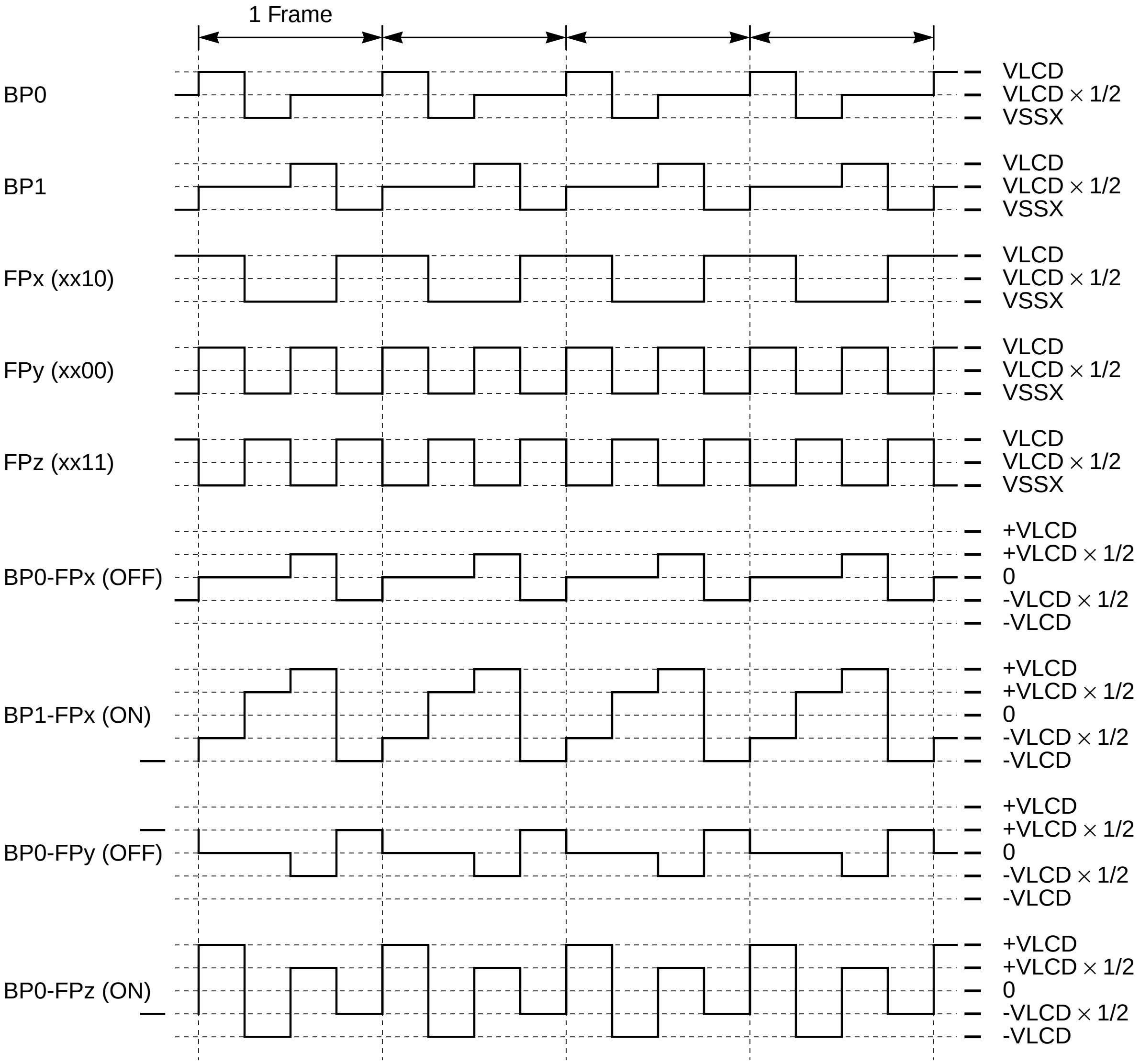


Figure 10-10. 1/2 Duty and 1/2 Bias

10.4.5.3 1/2 Duty Multiplexed with 1/3 Bias Mode

Duty = 1/2:DUTY1 = 1, DUTY0 = 0

Bias = 1/3:BIAS = 1

V_0 = VSSX, V_1 = VLCD * 1/3, V_2 = VLCD * 2/3, V_3 = VLCD

- BP2 and BP3 are not used, a maximum of 64 segments are displayed.

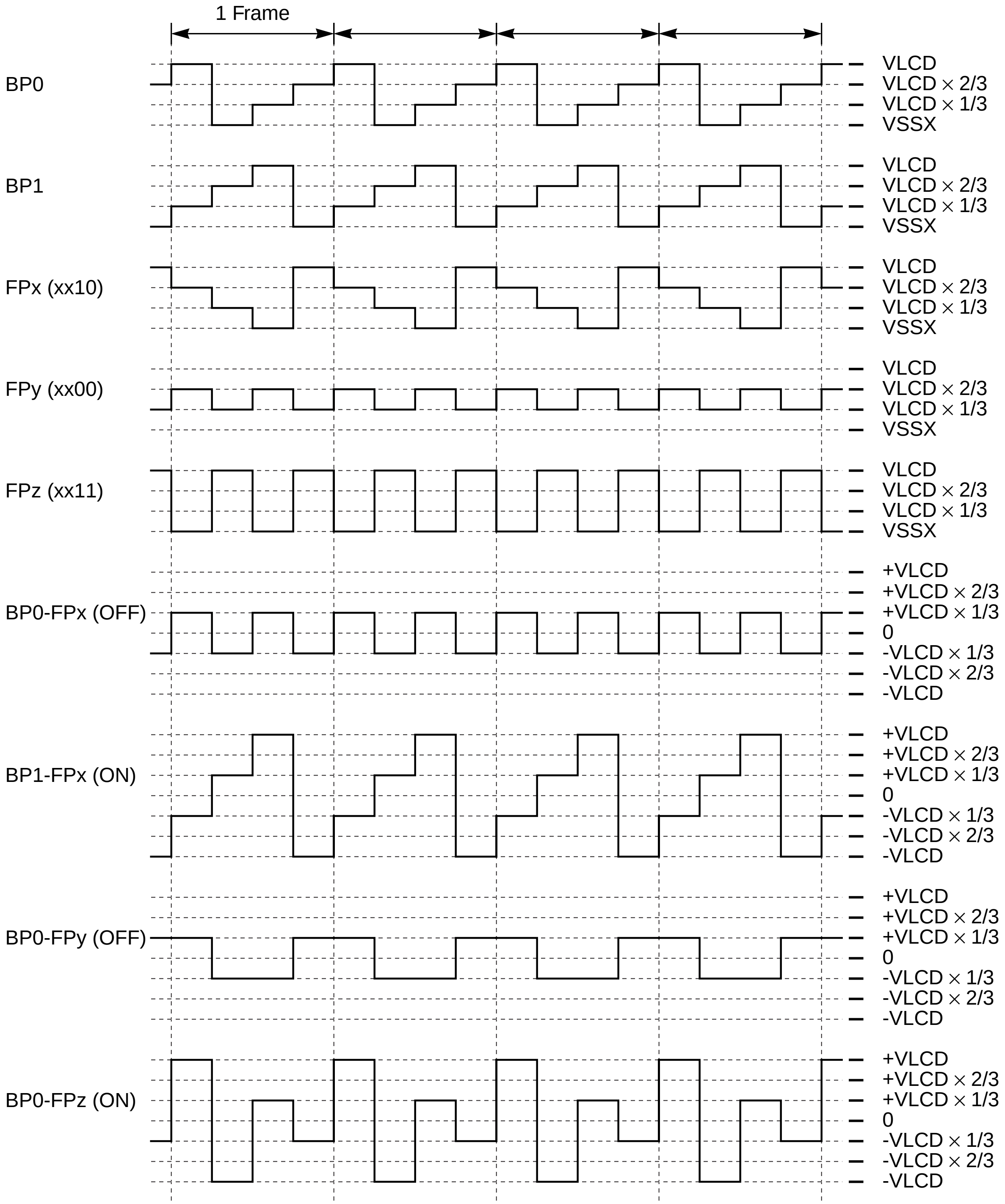


Figure 10-11. 1/2 Duty and 1/3 Bias

10.4.5.4 1/3 Duty Multiplexed with 1/3 Bias Mode

Duty = 1/3:DUTY1 = 1, DUTY0 = 1

Bias = 1/3:BIAS = 0 or BIAS = 1

V_0 = VSSX, V_1 = VLCD * 1/3, V_2 = VLCD * 2/3, V_3 = VLCD

- BP3 is not used, a maximum of 96 segments are displayed.

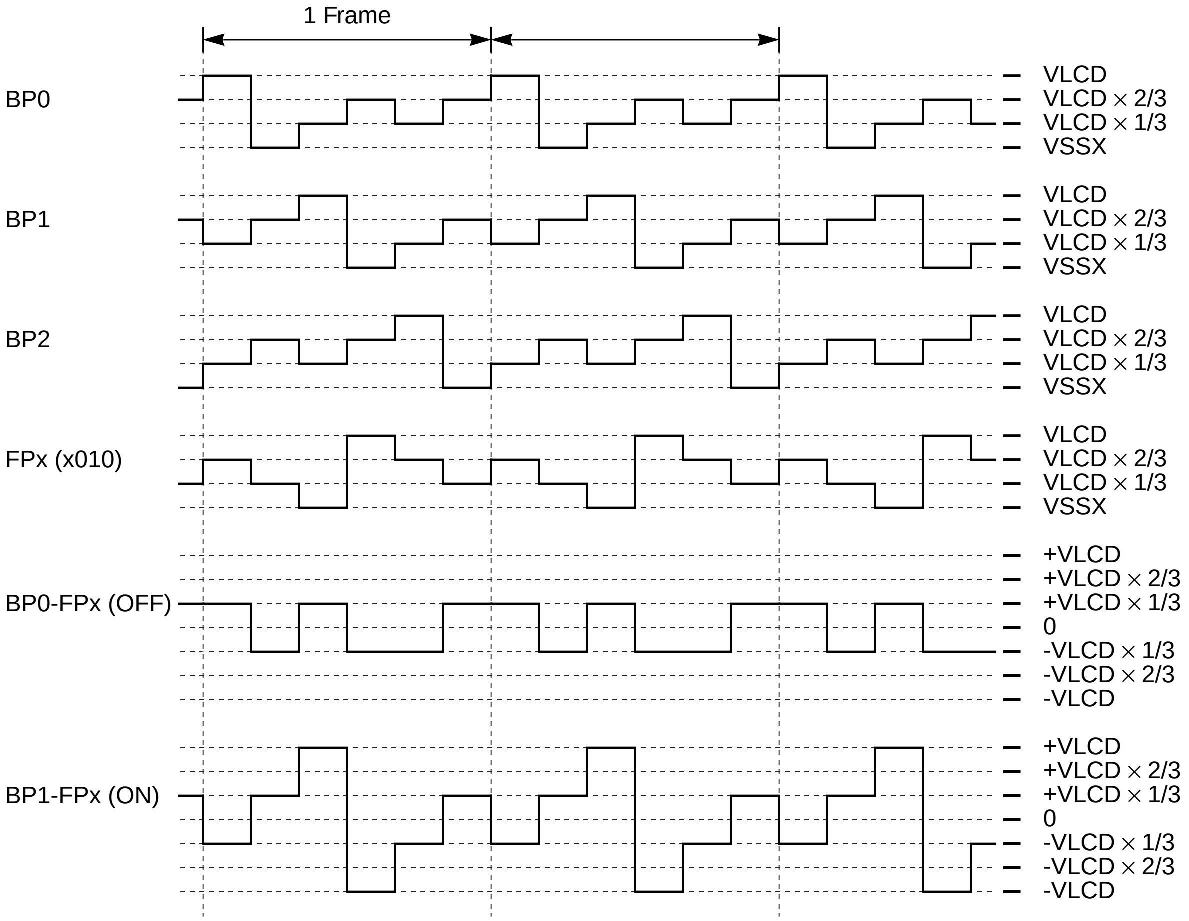


Figure 10-12. 1/3 Duty and 1/3 Bias

10.4.5.5 1/4 Duty Multiplexed with 1/3 Bias Mode

Duty = 1/4:DUTY1 = 0, DUTY0 = 0

Bias = 1/3:BIAS = 0 or BIAS = 1

V_0 = VSSX, V_1 = VLCD * 1/3, V_2 = VLCD * 2/3, V_3 = VLCD

- A maximum of 128 segments are displayed.

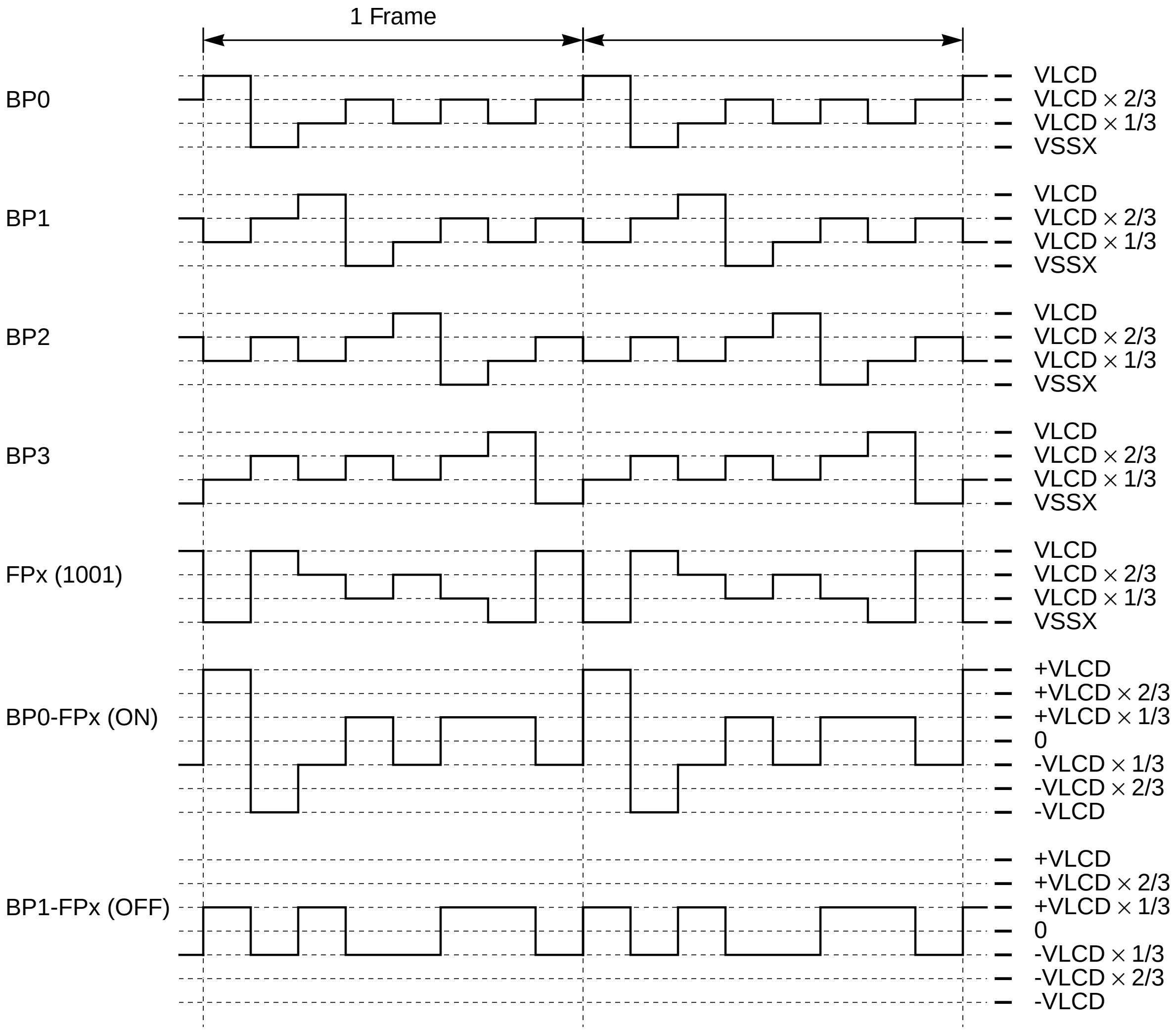


Figure 10-13. 1/4 Duty and 1/3 Bias

10.5 Resets

The reset values of registers and signals are described in Section 10.3, “Memory Map and Register Definition”. The behavior of the LCD32F4BV1 system during reset is described in Section 10.4.1, “LCD Driver Description”.

10.6 Interrupts

This module does not generate any interrupts.

Chapter 11
Motor Controller (MC10B12CV2) Block Description

11.1 Introduction

The block MC10B12C is a PWM motor controller suitable to drive instruments in a cluster configuration or any other loads requiring a PWM signal. The motor controller has twelve PWM channels associated with two pins each (24 pins in total).

11.1.1 Features

The MC_10B12C includes the following features:

- 10/11-bit PWM counter
- 11-bit resolution with selectable PWM dithering function
- 7-bit resolution mode (fast mode): duty cycle can be changed by accessing only 1 byte/output
- Left, right, or center aligned PWM
- Output slew rate control
- This module is suited for, but not limited to, driving small stepper and air core motors used in instrumentation applications. This module can be used for other motor control or PWM applications that match the frequency, resolution, and output drive capabilities of the module.

11.1.2 Modes of Operation

11.1.2.1 Functional Modes

11.1.2.1.1 PWM Resolution

The motor controller can be configured to either 11- or 7-bits resolution mode by clearing or setting the FAST bit. This bit influences all PWM channels. For details, please refer to Section 11.3.2.5, “Motor Controller Duty Cycle Registers”.

11.1.2.1.2 Dither Function

Dither function can be selected or deselected by setting or clearing the DITH bit. This bit influences all PWM channels. For details, please refer to Section 11.4.1.3.5, “Dither Bit (DITH)”.

11.1.2.2 PWM Channel Configuration Modes

The twelve PWM channels can operate in three functional modes. Those modes are, with some restrictions, selectable for each channel independently.

11.1.2.2.1 Dual Full H-Bridge Mode

This mode is suitable to drive a stepper motor or a 360^{o} air gauge instrument. For details, please refer to Section 11.4.1.1.1, “Dual Full H-Bridge Mode (MCOM = 11)”. In this mode two adjacent PWM channels are combined, and two PWM channels drive four pins.

11.1.2.2.2 Full H-Bridge Mode

This mode is suitable to drive any load requiring a PWM signal in a H-bridge configuration using two pins. For details please refer to Section 11.4.1.1.2, “Full H-Bridge Mode (MCOM = 10)”.

11.1.2.2.3 Half H-Bridge Mode

This mode is suitable to drive a 90^{o} instrument driven by one pin. For details, please refer to Section 11.4.1.1.3, “Half H-Bridge Mode (MCOM = 00 or 01)”.

11.1.2.3 PWM Alignment Modes

Each PWM channel can operate independently in three different alignment modes. For details, please refer to Section 11.4.1.3.1, “PWM Alignment Modes”.

11.1.2.4 Low-Power Modes

The behavior of the motor controller in low-power modes is programmable. For details, please refer to Section 11.4.5, “Operation in Wait Mode” and Section 11.4.6, “Operation in Stop and Pseudo-Stop Modes”.

11.1.3 Block Diagram

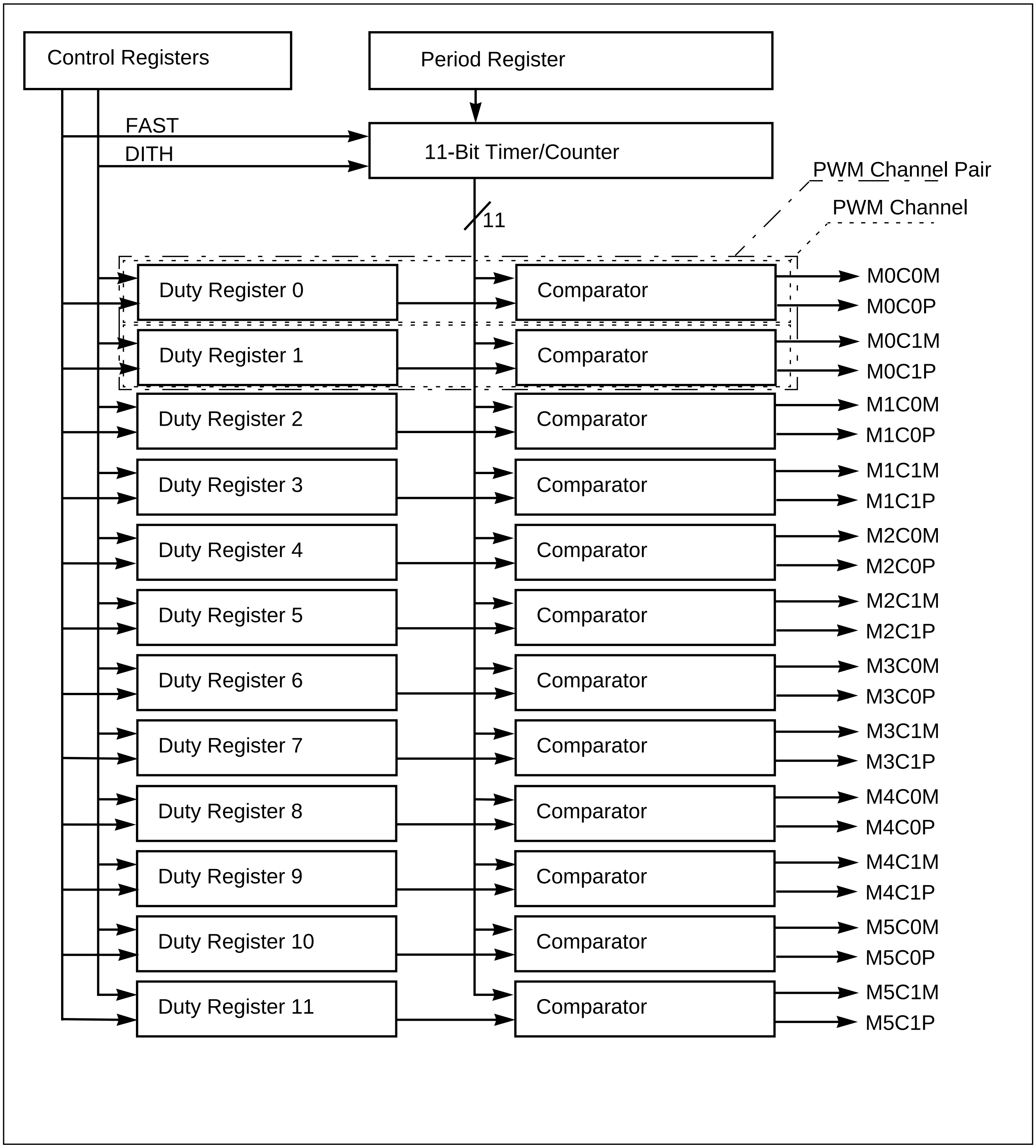


Figure 11-1. MC10B12C Block Diagram

11.2 External Signal Description

The motor controller is associated with 24 pins. Table 11-1 lists the relationship between the PWM channels and signal pins as well as PWM channel pair (motor number), coils, and nodes they are supposed to drive if all channels are set to dual full H-bridge configuration.

Table 11-1. PWM Channel and Pin Assignment

Pin Name	PWM Channel	PWM Channel Pair[1]	Coil	Node
M0C0M	0	0	0	Minus
M0C0P				Plus
M0C1M	1		1	Minus
M0C1P				Plus
M1C0M	2	1	0	Minus
M1C0P				Plus
M1C1M	3		1	Minus
M1C1P				Plus
M2C0M	4	2	0	Minus
M2C0P				Plus
M2C1M	5		1	Minus
M2C1P				Plus
M3C0M	6	3	0	Minus
M3C0P				Plus
M3C1M	7		1	Minus
M3C1P				Plus
M4C0M	8	4	0	Minus
M4C0P				Plus
M4C1M	9		1	Minus
M4C1P				Plus
M5C0M	10	5	0	Minus
M5C0P				Plus
M5C1M	11		1	Minus
M5C1P				Plus

[1] A PWM Channel Pair always consists of PWM channel x and PWM channel x+1 (x = 2·n). The term “PWM Channel Pair” is equivalent to the term “Motor”. E.g. Channel Pair 0 is equivalent to Motor 0

11.2.1 M0C0M/M0C0P/M0C1M/M0C1P — PWM Output Pins for Motor 0

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 0. PWM output on M0C0M results in a positive current flow through coil 0 when M0C0P is driven to a logic high state. PWM output on M0C1M results in a positive current flow through coil 1 when M0C1P is driven to a logic high state.

11.2.2 M1C0M/M1C0P/M1C1M/M1C1P — PWM Output Pins for Motor 1

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 1. PWM output on M1C0M results in a positive current flow through coil 0 when M1C0P is driven to a logic high state. PWM output on M1C1M results in a positive current flow through coil 1 when M1C1P is driven to a logic high state.

11.2.3 M2C0M/M2C0P/M2C1M/M2C1P — PWM Output Pins for Motor 2

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 2. PWM output on M2C0M results in a positive current flow through coil 0 when M2C0P is driven to a logic high state. PWM output on M2C1M results in a positive current flow through coil 1 when M2C1P is driven to a logic high state.

11.2.4 M3C0M/M3C0P/M3C1M/M3C1P — PWM Output Pins for Motor 3

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 3. PWM output on M3C0M results in a positive current flow through coil 0 when M3C0P is driven to a logic high state. PWM output on M3C1M results in a positive current flow through coil 1 when M3C1P is driven to a logic high state.

11.2.5 M4C0M/M4C0P/M4C1M/M4C1P — PWM Output Pins for Motor 4

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 4. PWM output on M4C0M results in a positive current flow through coil 0 when M4C0P is driven to a logic high state. PWM output on M4C1M results in a positive current flow through coil 1 when M4C1P is driven to a logic high state.

11.2.6 M5C0M/M5C0P/M5C1M/M5C1P — PWM Output Pins for Motor 5

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 5. PWM output on M5C0M results in a positive current flow through coil 0 when M5C0P is driven to a logic high state. PWM output on M5C1M results in a positive current flow through coil 1 when M5C1P is driven to a logic high state.

11.3 Memory Map and Register Definition

This section provides a detailed description of all registers of the 10-bit 12-channel motor controller module.

11.3.1 Module Memory Map

Table 11-2 shows the memory map of the 10-bit 12-channel motor controller module.

Table 11-2. MC10B12C - Memory Map

Address offset	Use	Access
$00	MCCTL0	RW
$01	MCCTL1	RW
$02	MCPER (high byte)	RW
$03	MCPER (low byte)	RW
$04	Reserved	-
$05	Reserved	-
$06	Reserved	-
$07	Reserved	-
$08	Reserved	-
$09	Reserved	-
$0A	Reserved	-
$0B	Reserved	-
$0C	Reserved	-
$0D	Reserved	-
$0E	Reserved	-
$0F	Reserved	-
$10	MCCC0	RW
$11	MCCC1	RW
$12	MCCC2	RW
$13	MCCC3	RW
$14	MCCC4	RW
$15	MCCC5	RW
$16	MCCC6	RW
$17	MCCC7	RW
$18	MCCC8	RW
$19	MCCC9	RW
$1A	MCCC10	RW
$1B	MCCC11	RW
$1C	Reserved	-
$1D	Reserved	-
$1E	Reserved	-
$1F	Reserved	-
$20	MCDC0 (high byte)	RW
$21	MCDC0 (low byte)	RW
$22	MCDC1 (high byte)	RW

Table 11-2. MC10B12C - Memory Map

$23	MCDC1 (low byte)	RW
$24	MCDC2 (high byte)	RW
$25	MCDC2 (low byte)	RW
$26	MCDC3 (high byte)	RW
$27	MCDC3 (low byte)	RW
$28	MCDC4 (high byte)	RW
$29	MCDC4 (low byte)	RW
$2A	MCDC5 (high byte)	RW
$2B	MCDC5 (low byte)	RW
$2C	MCDC6 (high byte)	RW
$2D	MCDC6 (low byte)	RW
$2E	MCDC7 (high byte)	RW
$2F	MCDC7 (low byte)	RW
$30	MCDC8 (high byte)	RW
$31	MCDC8 (low byte)	RW
$32	MCDC9 (high byte)	RW
$33	MCDC9 (low byte)	RW
$34	MCDC10 (high byte)	RW
$35	MCDC10 (low byte)	RW
$36	MCDC11 (high byte)	RW
$37	MCDC11 (low byte)	RW
$38	Reserved	-
$39	Reserved	-
$3A	Reserved	-
$3B	Reserved	-
$3C	Reserved	-
$3D	Reserved	-
$3E	Reserved	-
$3F	Reserved	-

11.3.2 Register Descriptions

11.3.2.1 Motor Controller Control Register 0

This register controls the operating mode of the motor controller module.

Offset Module Base + 0x0000

	7	6	5	4	3	2	1	0
R	0	MCPRE[1:0]		MCSWAI	FAST	DITH	0	MCTOIF
W	▨						▨	
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 11-3. Motor Controller Control Register 0 (MCCTL0)

Table 11-3. MCCTL0 Field Descriptions

Field	Description
6:5 MCPRE[1:0]	**Motor Controller Prescaler Select** — MCPRE1 and MCPRE0 determine the prescaler value that sets the motor controller timer counter clock frequency (f_{TC}). The clock source for the prescaler is the peripheral bus clock (f_{BUS}) as shown in Figure 11-22. Writes to MCPRE1 or MCPRE0 will not affect the timer counter clock frequency f_{TC} until the start of the next PWM period. Table 11-4 shows the prescaler values that result from the possible combinations of MCPRE1 and MCPRE0
4 MCSWAI	**Motor Controller Module Stop in Wait Mode** 0 Entering wait mode has no effect on the motor controller module and the associated port pins maintain the functionality they had prior to entering wait mode both during wait mode and after exiting wait mode. 1 Entering wait mode will stop the clock of the module and debias the analog circuitry. The module will release the pins.
3 FAST	**Motor Controller PWM Resolution Mode** 0 PWM operates in 11-bit resolution mode, duty cycle registers of all channels are switched to word mode. 1 PWM operates in 7-bit resolution (fast) mode, duty cycle registers of all channels are switched to byte mode.
2 DITH	**Motor Control/Driver Dither Feature Enable** (refer to Section 11.4.1.3.5, “Dither Bit (DITH)”) 0 Dither feature is disabled. 1 Dither feature is enabled.
0 MCTOIF	**Motor Controller Timer Counter Overflow Interrupt Flag** — This bit is set when a motor controller timer counter overflow occurs. The bit is cleared by writing a 1 to the bit. 0 A motor controller timer counter overflow has not occurred since the last reset or since the bit was cleared. 1 A motor controller timer counter overflow has occurred.

Table 11-4. Prescaler Values

MCPRE[1:0]	f_{TC}
00	f_{Bus}
01	$f_{Bus}/2$
10	$f_{Bus}/4$
11	$f_{Bus}/8$

11.3.2.2 Motor Controller Control Register 1

This register controls the behavior of the analog section of the motor controller as well as the interrupt enables.

Offset Module Base + 0x0001

	7	6	5	4	3	2	1	0
R	RECIRC	0	0	0	0	0	0	MCTOIE
W		▨	▨	▨	▨	▨	▨	
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 11-4. Motor Controller Control Register 1 (MCCTL1)

Table 11-5. MCCTL1 Field Descriptions

Field	Description
7 RECIRC	**Recirculation in (Dual) Full H-Bridge Mode** (refer to Section 11.4.1.3.3, “RECIRC Bit”)— RECIRC only affects the outputs in (dual) full H-bridge modes. In half H-bridge mode, the PWM output is always active low. RECIRC = 1 will also invert the effect of the S bits (refer to Section 11.4.1.3.2, “Sign Bit (S)”) in (dual) full H-bridge modes. RECIRC must be changed only while no PWM channel is operating in (dual) full H-bridge mode; otherwise, erroneous output pattern may occur. 0 Recirculation on the high side transistors. Active state for PWM output is logic low, the static channel will output logic high. 1 Recirculation on the low side transistors. Active state for PWM output is logic high, the static channel will output logic low.
0 MCTOIE	**Motor Controller Timer Counter Overflow Interrupt Enable** 0 Interrupt disabled. 1 Interrupt enabled. An interrupt will be generated when the motor controller timer counter overflow interrupt flag (MCTOIF) is set.

11.3.2.3 Motor Controller Period Register

The period register defines PER, the number of motor controller timer counter clocks a PWM period lasts. The motor controller timer counter is clocked with the frequency f_{TC}. If dither mode is enabled (DITH = 1, refer to Section 11.4.1.3.5, "Dither Bit (DITH)"), P0 is ignored and reads as a 0. In this case PER = 2 * D[10:1].

Offset Module Base + 0x0002, 0x0003

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
W	▨	▨	▨	▨	▨											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 11-5. Motor Controller Period Register (MCPER) with DITH = 0

Offset Module Base + 0x0002, 0x0003

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	0
W	▨	▨	▨	▨	▨											▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 11-6. Motor Controller Period Register (MCPER) with DITH = 1

For example, programming MCPER to 0x0022 (PER = 34 decimal) will result in 34 counts for each complete PWM period. Setting MCPER to 0 will shut off all PWM channels as if MCAM[1:0] is set to 0 in all channel control registers after the next period timer counter overflow. In this case, the motor controller releases all pins.

NOTE

> Programming MCPER to 0x0001 and setting the DITH bit will be managed as if MCPER is programmed to 0x0000. All PWM channels will be shut off after the next period timer counter overflow.

11.3.2.4 Motor Controller Channel Control Registers

Each PWM channel has one associated control register to control output delay, PWM alignment, and output mode. The registers are named MCCC0... MCCC11. In the following, MCCC0 is described as a reference for all twelve registers.

Offset Module Base + 0x0010 . . . 0x001B

	7	6	5	4	3	2	1	0
R	MCOM1	MCOM0	MCAM1	MCAM0	0	0	CD1	CD0
W					▨	▨		
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 11-7. Motor Controller Control Register Channel0 .. 11 (MCCC0 .. MCCC11)

Table 11-6. MCCC0–MCCC11 Field Descriptions

Field	Description
7:6 MCOM[1:0]	**Output Mode** — MCOM1, MCOM0 control the PWM channel's output mode. See Table 11-7.
5:4 MCAM[1:0]	**PWM Channel Alignment Mode** — MCAM1, MCAM0 control the PWM channel's PWM alignment mode and operation. See Table 11-8. MCAM[1:0] and MCOM[1:0] are double buffered. The values used for the generation of the output waveform will be copied to the working registers either at once (if all PWM channels are disabled or MCPER is set to 0) or if a timer counter overflow occurs. Reads of the register return the most recent written value, which are not necessarily the currently active values.
1:0 CD[1:0]	**PWM Channel Delay** — Each PWM channel can be individually delayed by a programmable number of PWM timer counter clocks. The delay will be n/f_{TC}. See Table 11-9.

Table 11-7. Output Mode

MCOM[1:0]	Output Mode
00	Half H-bridge mode, PWM on MnCxM, MnCxP is released
01	Half H-bridge mode, PWM on MnCxP, MnCxM is released
10	Full H-bridge mode
11	Dual full H-bridge mode

Table 11-8. PWM Alignment Mode

MCAM[1:0]	PWM Alignment Mode
00	Channel disabled
01	Left aligned
10	Right aligned
11	Center aligned

Table 11-9. Channel Delay

CD[1:0]	n [# of PWM Clocks]
00	0
01	1
10	2
11	3

NOTE

> The PWM motor controller will release the pins after the next PWM timer counter overflow without accommodating any channel delay if a single channel has been disabled or if the period register has been cleared or all channels have been disabled. Program one or more inactive PWM frames (duty cycle = 0) before writing a configuration that disables a single channel or the entire PWM motor controller.

11.3.2.5 Motor Controller Duty Cycle Registers

Each duty cycle register sets the sign and duty functionality for the respective PWM channel.

The contents of the duty cycle registers define DUTY, the number of motor controller timer counter clocks the corresponding output is driven low (RECIRC = 0) or is driven high (RECIRC = 1). Setting all bits to 0 will give a static high output in case of RECIRC = 0; otherwise, a static low output. Values greater than or equal to the contents of the period register will generate a static low output in case of RECIRC = 0, or a static high output if RECIRC = 1. The layout of the duty cycle registers differ dependent upon the state of the FAST bit in the control register 0.

Offset Module Base + 0x0020 . . . 0x0037 — Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	S	S	S	S	S	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
W		▨	▨	▨	▨											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 11-8. Motor Controller Duty Cycle Register x (MCDCx) with FAST = 0

Offset Module Base + 0x0020 . . . 0x0037 Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	S	D8	D7	D6	D5	D4	D3	D2	0	0	0	0	0	0	0	0
W									▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 11-9. Motor Controller Duty Cycle Register x (MCDCx) with FAST = 1

Table 11-10. MCDCx Field Descriptions

Field	Description
0 S	SIGN — The SIGN bit is used to define which output will drive the PWM signal in (dual) full-H-bridge modes. The SIGN bit has no effect in half-bridge modes. See Section 11.4.1.3.2, “Sign Bit (S)”, and table Table 11-12 for detailed information about the impact of RECIRC and SIGN bit on the PWM output.

Whenever FAST = 1, the bits D10, D9, D1, and D0 will be set to 0 if the duty cycle register is written.

For example setting MCDCx = 0x0158 with FAST = 0 gives the same output waveform as setting MCDCx = 0x5600 with FAST = 1 (with FAST = 1, the low byte of MCDCx needs not to be written).

The state of the FAST bit has impact only during write and read operations. A change of the FAST bit (set or clear) without writing a new value does not impact the internal interpretation of the duty cycle values.

To prevent the output from inconsistent signals, the duty cycle registers are double buffered. The motor controller module will use working registers to generate the output signals. The working registers are copied from the bus accessible registers at the following conditions:

- MCPER is set to 0 (all channels are disabled in this case)
- MCAM[1:0] of the respective channel is set to 0 (channel is disabled)
- A PWM timer counter overflow occurs while in half H-bridge or full H-bridge mode
- A PWM channel pair is configured to work in Dual Full H-Bridge mode and a PWM timer counter overflow occurs after the odd[1] duty cycle register of the channel pair has been written.

In this way, the output of the PWM will always be either the old PWM waveform or the new PWM waveform, not some variation in between.

Reads of this register return the most recent value written. Reads do not necessarily return the value of the currently active sign, duty cycle, and dither functionality due to the double buffering scheme.

---

[1] Odd duty cycle register: MCDCx+1, x = 2·n

11.4 Functional Description

11.4.1 Modes of Operation

11.4.1.1 PWM Output Modes

The motor controller is configurable between three output modes.

- Dual full H-bridge mode can be used to control either a stepper motor or a 360° air core instrument. In this case two PWM channels are combined.
- In full H-bridge mode, each PWM channel is updated independently.
- In half H-bridge mode, one pin of the PWM channel can generate a PWM signal to control a 90° air core instrument (or other load requiring a PWM signal) and the other pin is unused.

The mode of operation for each PWM channel is determined by the corresponding MCOM[1:0] bits in channel control registers. After a reset occurs, each PWM channel will be disabled, the corresponding pins are released.

Each PWM channel consists of two pins. One output pin will generate a PWM signal. The other will operate as logic high or low output depending on the state of the RECIRC bit (refer to Section 11.4.1.3.3, “RECIRC Bit”), while in (dual) full H-bridge mode, or will be released, while in half H-bridge mode. The state of the S bit in the duty cycle register determines the pin where the PWM signal is driven in full H-bridge mode. While in half H-bridge mode, the state of the released pin is determined by other modules associated with this pin.

Associated with each PWM channel pair n are two PWM channels, x and x + 1, where x = 2 * n and n (0,1,2... 5) is the PWM channel pair number. Duty cycle register x controls the sign of the PWM signal (which pin drives the PWM signal) and the duty cycle of the PWM signal for motor controller channel x. The pins associated with PWM channel x are MnC0P and MnC0M. Similarly, duty cycle register x + 1 controls the sign of the PWM signal and the duty cycle of the PWM signal for channel x + 1. The pins associated with PWM channel x + 1 are MnC1P and MnC1M. This is summarized in Table 11-11.

Table 11-11. Corresponding Registers and Pin Names for each PWM Channel Pair

PWM Channel Pair Number	PWM Channel Control Register	Duty Cycle Register	Channel Number	Pin Names
n	MCMCx	MCDCx	PWM Channel x, x = 2·n	MnC0M
				MnC0P
	MCMCx+1	MCDCx+1	PWM Channel x+1, x = 2·n	MnC1M
				MnC1P
0	MCMC0	MCDC0	PWM Channel 0	M0C0M
				M0C0P
	MCMC1	MCDC1	PWM Channel 1	M0C1M
				M0C1P

Table 11-11. Corresponding Registers and Pin Names for each PWM Channel Pair

PWM Channel Pair Number	PWM Channel Control Register	Duty Cycle Register	Channel Number	Pin Names
1	MCMC2	MCDC2	PWM Channel 2	M1C0M
				M1C0P
	MCMC3	MCDC3	PWM Channel 3	M1C1M
				M1C1P
2	MCMC4	MCDC4	PWM Channel 4	M2C0M
				M2C0P
	MCMC5	MCDC5	PWM Channel 5	M2C1M
				M2C1P
3	MCMC6	MCDC6	PWM Channel 6	M3C0M
				M3C0P
	MCMC7	MCDC7	PWM Channel 7	M3C1M
				M3C1P
4	MCMC8	MCDC8	PWM Channel 8	M4C0M
				M4C0P
	MCMC9	MCDC9	PWM Channel 9	M4C1M
				M4C1P
5	MCMC10	MCDC10	PWM Channel 10	M5C0M
				M5C0P
	MCMC11	MCDC11	PWM Channel 11	M5C1M
				M5C1P

11.4.1.1.1 Dual Full H-Bridge Mode (MCOM = 11)

PWM channel pairs x and x + 1 operate in dual full H-bridge mode if both channels have been enabled (MCAM[1:0]=01, 10, or 11) and both of the corresponding output mode bits MCOM[1:0] in both PWM channel control registers are set.

A typical configuration in dual full H-bridge mode is shown in Figure 11-10. PWM channel x drives the PWM output signal on either MnC0P or MnC0M. If MnC0P drives the PWM signal, MnC0M will be output either high or low depending on the RECIRC bit. If MnC0M drives the PWM signal, MnC0P will be an output high or low. PWM channel x + 1 drives the PWM output signal on either MnC1P or MnC1M. If MnC1P drives the PWM signal, MnC1M will be an output high or low. If MnC1M drives the PWM signal, MnC1P will be an output high or low. This results in motor recirculation currents on the high side drivers (RECIRC = 0) while the PWM signal is at a logic high level, or motor recirculation currents on the low side drivers (RECIRC = 1) while the PWM signal is at a logic low level. The pin driving the PWM signal is determined by the S (sign) bit in the corresponding duty cycle register and the state of the RECIRC bit. The value of the PWM duty cycle is determined by the value of the D[10:0] or D[8:2] bits respectively in the duty cycle register depending on the state of the FAST bit.

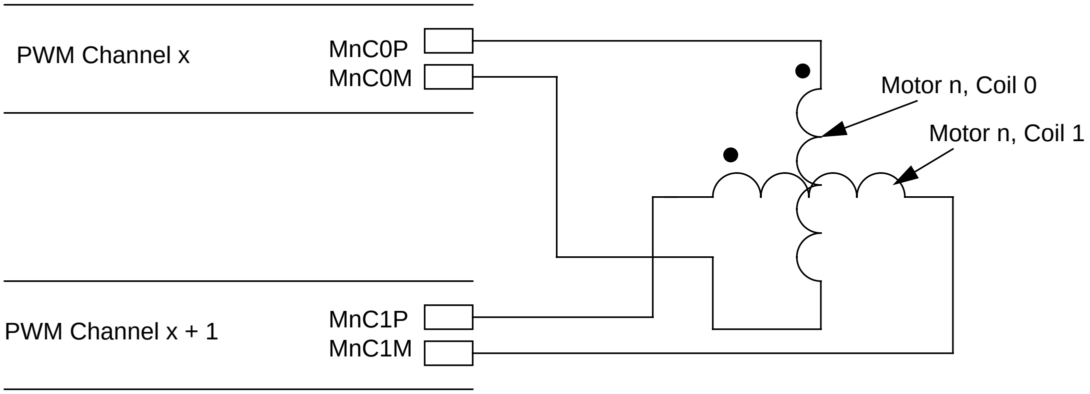


Figure 11-10. Typical Dual Full H-Bridge Mode Configuration

Whenever FAST = 0 only 16-bit write accesses to the duty cycle registers are allowed, 8-bit write accesses can lead to unpredictable duty cycles.

While fast mode is enabled (FAST = 1), 8-bit write accesses to the high byte of the duty cycle registers are allowed, because only the high byte of the duty cycle register is used to determine the duty cycle.

The following sequence should be used to update the current magnitude and direction for coil 0 and coil 1 of the motor to achieve consistent PWM output:

1. Write to duty cycle register x
2. Write to duty cycle register x + 1.

At the next timer counter overflow, the duty cycle registers will be copied to the working duty cycle registers. Sequential writes to the duty cycle register x will result in the previous data being overwritten.

11.4.1.1.2 Full H-Bridge Mode (MCOM = 10)

In full H-bridge mode, the PWM channels x and x + 1 operate independently. The duty cycle working registers are updated whenever a timer counter overflow occurs.

11.4.1.1.3 Half H-Bridge Mode (MCOM = 00 or 01)

In half H-bridge mode, the PWM channels x and x + 1 operate independently. In this mode, each PWM channel can be configured such that one pin is released and the other pin is a PWM output. Figure 11-11 shows a typical configuration in half H-bridge mode.

The two pins associated with each channel are switchable between released mode and PWM output dependent upon the state of the MCOM[1:0] bits in the MCCCx (channel control) register. See register description in Section 11.3.2.4, “Motor Controller Channel Control Registers”. In half H-bridge mode, the state of the S bit has no effect.

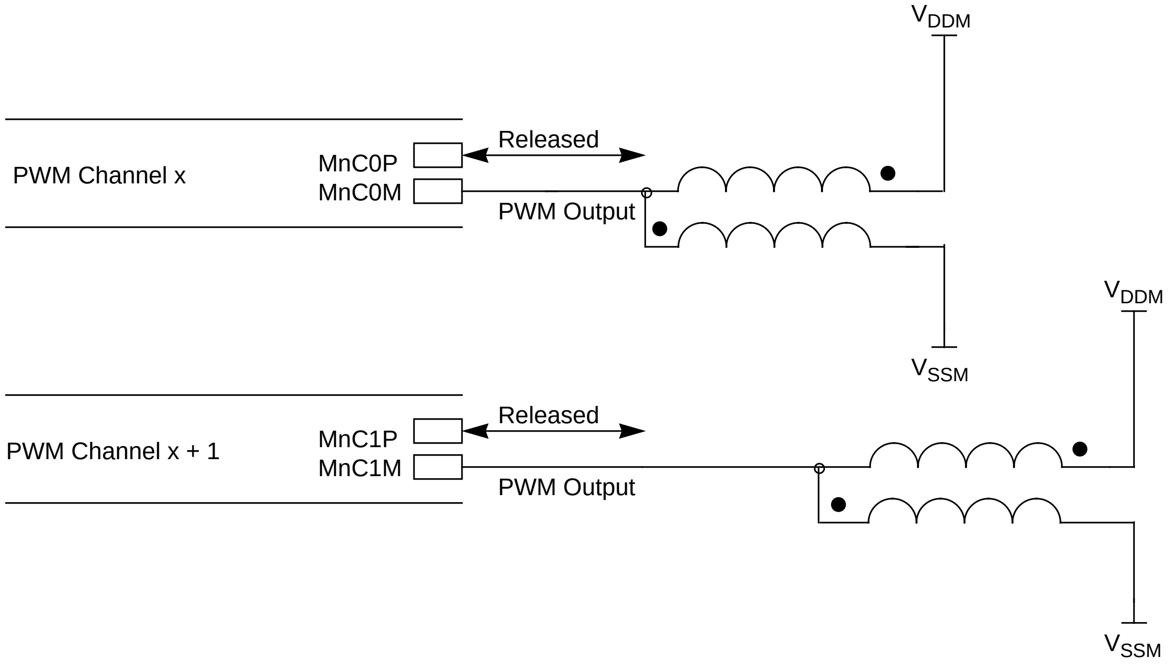


Figure 11-11. Typical Quad Half H-Bridge Mode Configuration

11.4.1.2 Relationship Between PWM Mode and PWM Channel Enable

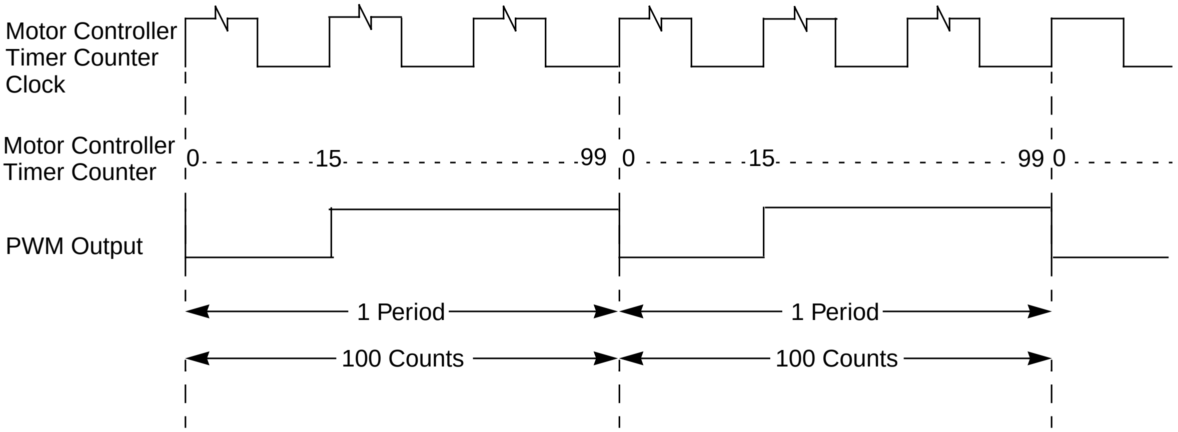
The pair of motor controller channels cannot be placed into dual full H-bridge mode unless both motor controller channels have been enabled (MCAM[1:0] not equal to 00) and dual full H-bridge mode is selected for both PWM channels (MCOM[1:0] = 11). If only one channel is set to dual full H-bridge mode, this channel will operate in full H-bridge mode, the other as programmed.

11.4.1.3 Relationship Between Sign, Duty, Dither, RECIRC, Period, and PWM Mode Functions

11.4.1.3.1 PWM Alignment Modes

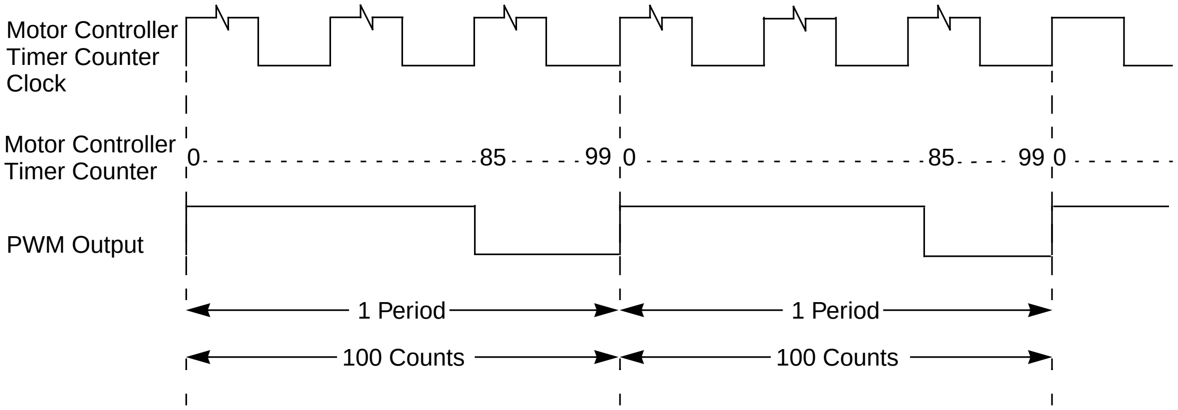
Each PWM channel can be programmed individually to three different alignment modes. The mode is determined by the MCAM[1:0] bits in the corresponding channel control register.

Left aligned (MCAM[1:0] = 01): The output will start active (low if RECIRC = 0 or high if RECIRC = 1) and will turn inactive (high if RECIRC = 0 or low if RECIRC = 1) after the number of counts specified by the corresponding duty cycle register.



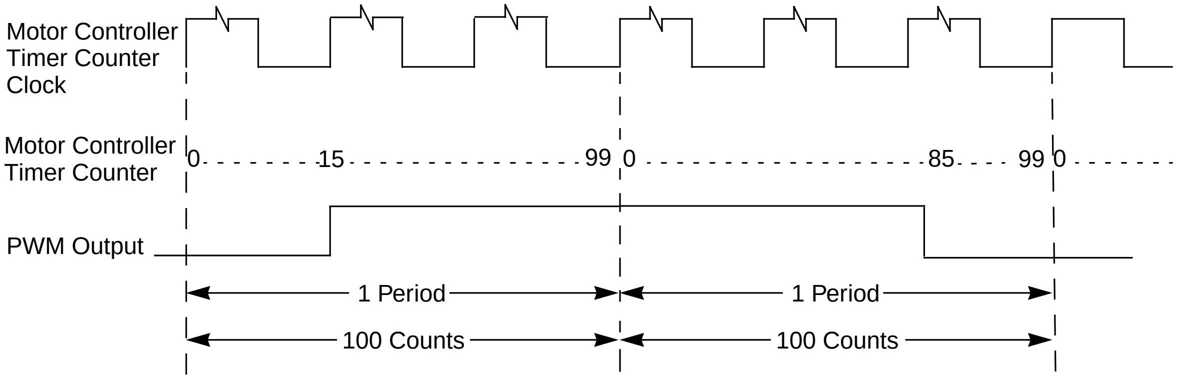
DITH = 0, MCAM[1:0] = 01, MCDCx = 15, MCPER = 100, RECIRC = 0

Right aligned (MCAM[1:0] = 10): The output will start inactive (high if RECIRC = 0 and low if RECIRC = 1) and will turn active after the number of counts specified by the difference of the contents of period register and the corresponding duty cycle register.



DITH = 0, MCAM[1:0] = 10, MCDCx = 15, MCPER = 100, RECIRC = 0

Center aligned (MCAM[1:0] = 11): Even periods will be output left aligned, odd periods will be output right aligned. PWM operation starts with the even period after the channel has been enabled. PWM operation in center aligned mode might start with the odd period if the channel has not been disabled before changing the alignment mode to center aligned.



DITH = 0, MCAM[1:0] = 11, MCDCx = 15, MCPER = 100, RECIRC = 0

11.4.1.3.2 Sign Bit (S)

Assuming RECIRC = 0 (the active state of the PWM signal is low), when the S bit for the corresponding channel is cleared, MnC0P (if the PWM channel number is even, n = 0, 1, 2...5, see Table 11-11) or MnC1P (if the PWM channel number is odd, ,n = 0, 1, 2...5 see Table 11-11), outputs a logic high while in (dual) full H-bridge mode. In half H-bridge mode the state of the S bit has no effect. The PWM output signal is generated on MnC0M (if the PWM channel number is even, n = 0, 1, 2...5, see Table 11-11) or MnC1M (if the PWM channel number is odd, n = 0, 1, 2...5).

Assuming RECIRC = 0 (the active state of the PWM signal is low), when the S bit for the corresponding channel is set, MnC0M (if the PWM channel number is even, n = 0, 1, 2...5, see Table 11-11) or MnC1M (if the PWM channel number is odd, n = 0, 1, 2...5, see Table 11-11), outputs a logic high while in (dual) full H-bridge mode. In half H-bridge mode the state of the S bit has no effect. The PWM output signal is generated on MnC0P (if the PWM channel number is even, n = 0, 1, 2...5, see Table 11-11) or MnC1P (if the PWM channel number is odd, n = 0, 1, 2...5).

Setting RECIRC = 1 will also invert the effect of the S bit such that while S = 0, MnC0P or MnC1P will generate the PWM signal and MnC0M or MnC1M will be a static low output. While S = 1, MnC0M or MnC1M will generate the PWM signal and MnC0P or MnC1P will be a static low output. In this case the active state of the PWM signal will be high.

See Table 11-12 for detailed information about the impact of SIGN and RECIRC bit on the PWM output.

Table 11-12. Impact of RECIRC and SIGN Bit on the PWM Output

Output Mode	RECIRC	SIGN	MnCyM	MnCyP
(Dual) Full H-Bridge	0	0	$\overline{\text{PWM}}$[1]	1
(Dual) Full H-Bridge	0	1	1	PWM
(Dual) Full H-Bridge	1	0	0	PWM[2]
(Dual) Full H-Bridge	1	1	PWM	1
Half H-Bridge: PWM on MnCyM	Don’t care	Don’t care	PWM	—[3]
Half H-Bridge: PWM on MnCyP	Don’t care	Don’t care	—	PWM

[1] $\overline{\text{PWM}}$: The PWM signal is low active. e.g., the waveform starts with 0 in left aligned mode. Output M generates the PWM signal. Output P is static high.

[2] PWM: The PWM signal is high active. e.g., the waveform starts with 1 in left aligned mode. output P generates the PWM signal. Output M is static low.

[3] The state of the output transistors is not controlled by the motor controller.

11.4.1.3.3 RECIRC Bit

The RECIRC bit controls the flow of the recirculation current of the load. Setting RECIRC = 0 will cause recirculation current to flow through the high side transistors, and RECIRC = 1 will cause the recirculation current to flow through the low side transistors. The RECIRC bit is only active in (dual) full H-bridge modes.

Effectively, RECIRC = 0 will cause a static high output on the output terminal not driven by the PWM, RECIRC = 1 will cause a static low output on the output terminals not driven by the PWM. To achieve the same current direction, the S bit behavior is inverted if RECIRC = 1. Figure 11-12, Figure 11-13, Figure 11-14, and Figure 11-15 illustrate the effect of the RECIRC bit in (dual) full H-bridge modes.

RECIRC bit must be changed only while no PWM channel is operated in (dual) full H-bridge mode.

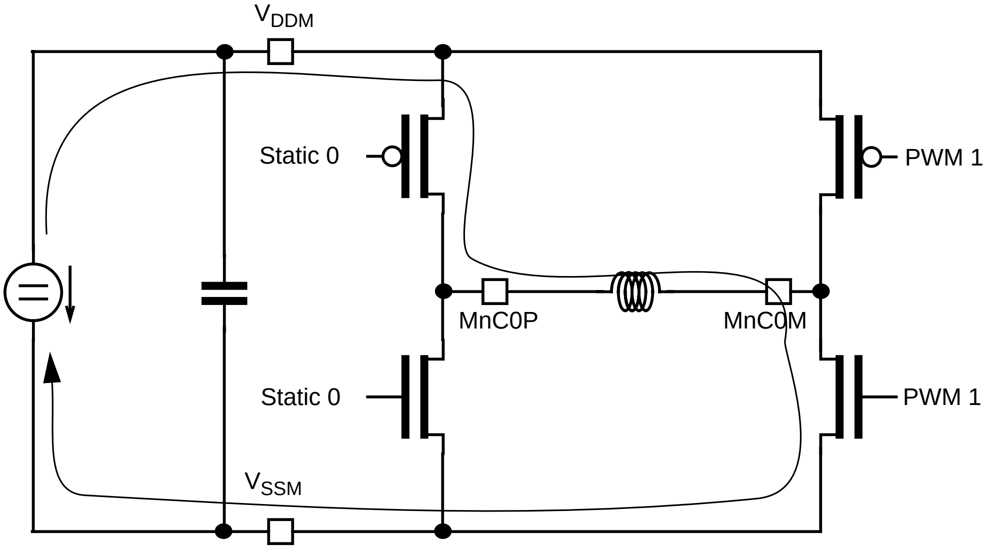


Figure 11-12. PWM Active Phase, RECIRC = 0, S = 0

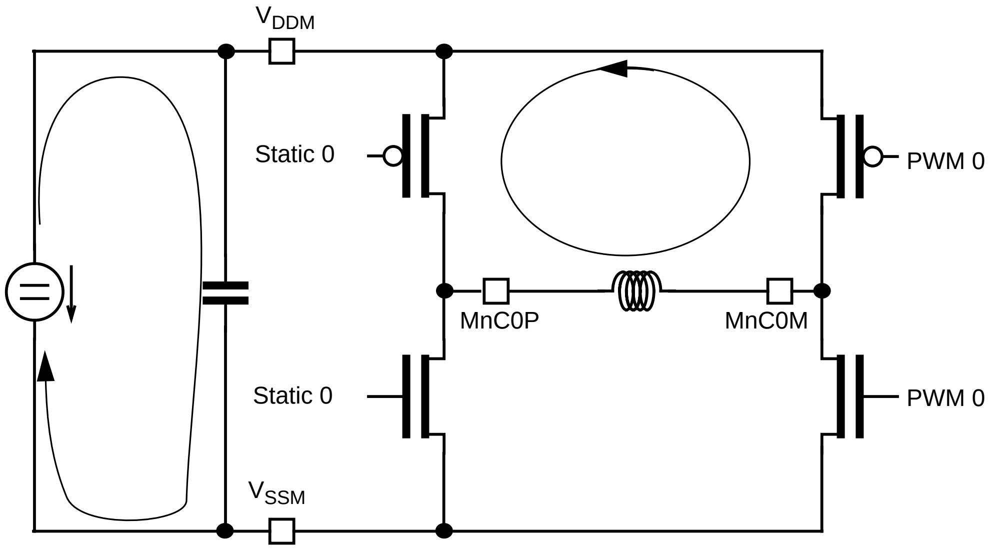


Figure 11-13. PWM Passive Phase, RECIRC = 0, S = 0

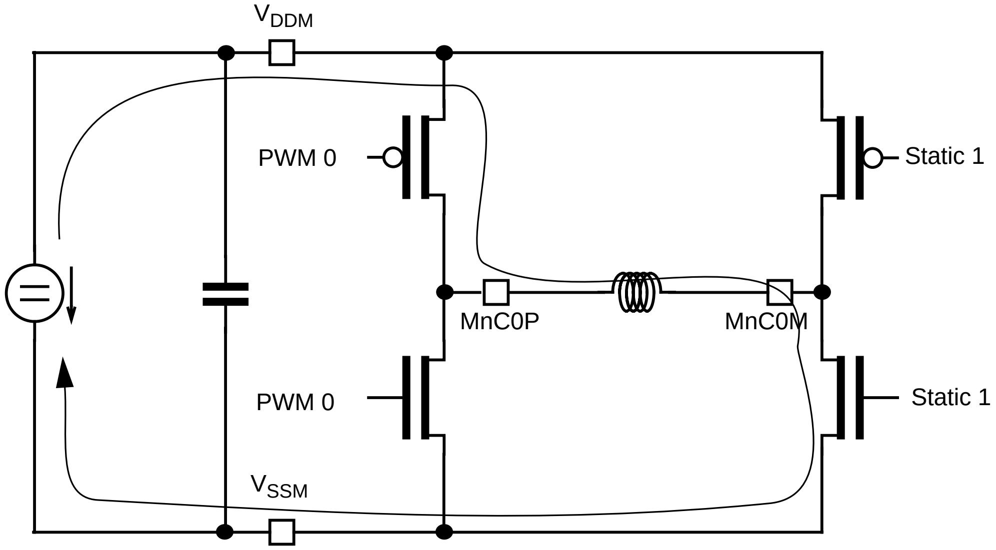


Figure 11-14. PWM Active Phase, RECIRC = 1, S = 0

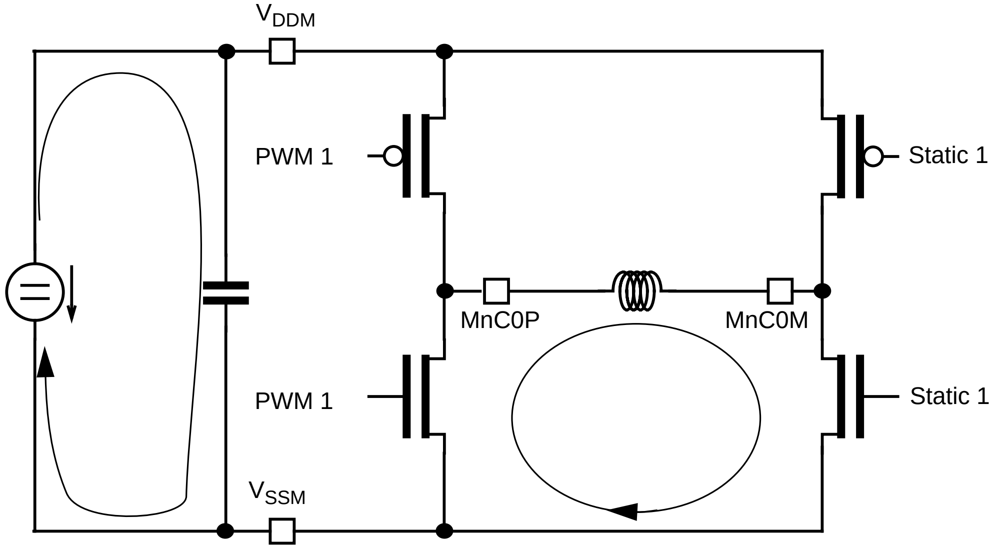


Figure 11-15. PWM Passive Phase, RECIRC = 1, S = 0

11.4.1.3.4 Relationship Between RECIRC Bit, S Bit, MCOM Bits, PWM State, and Output Transistors

Please refer to Figure 11-16 for the output transistor assignment.

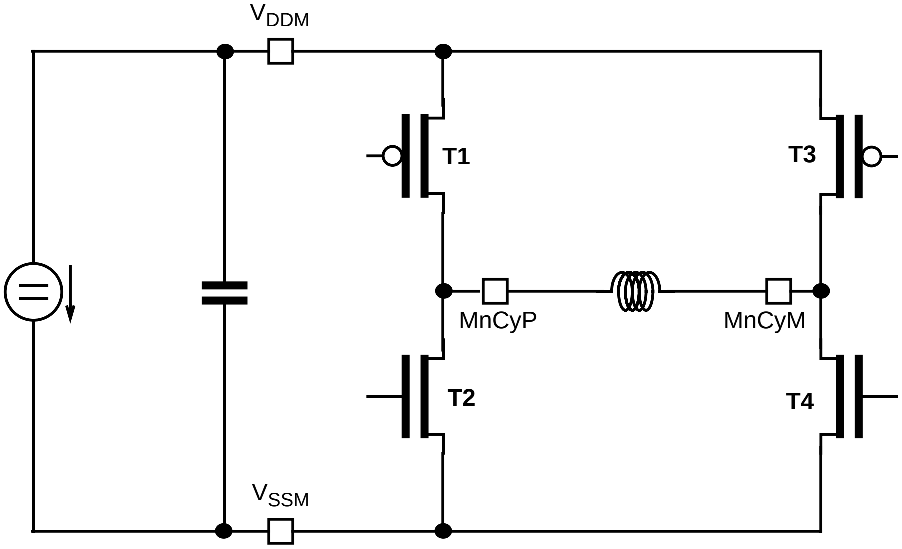


Figure 11-16. Output Transistor Assignment

Table 11-13 illustrates the state of the output transistors in different states of the PWM motor controller module. '—' means that the state of the output transistor is not controlled by the motor controller.

Table 11-13. State of Output Transistors in Various Modes

Mode	MCOM[1:0]	PWM Duty	RECIRC	S	T1	T2	T3	T4
Off	Don't care	—	Don't care	Don't care	—	—	—	—
Half H-Bridge	00	Active	Don't care	Don't care	—	—	OFF	ON
Half H-Bridge	00	Passive	Don't care	Don't care	—	—	ON	OFF
Half H-Bridge	01	Active	Don't care	Don't care	OFF	ON	—	—
Half H-Bridge	01	Passive	Don't care	Don't care	ON	OFF	—	—
(Dual) Full	10 or 11	Active	0	0	ON	OFF	OFF	ON
(Dual) Full	10 or 11	Passive	0	0	ON	OFF	ON	OFF
(Dual) Full	10 or 11	Active	0	1	OFF	ON	ON	OFF
(Dual) Full	10 or 11	Passive	0	1	ON	OFF	ON	OFF
(Dual) Full	10 or 11	Active	1	0	ON	OFF	OFF	ON
(Dual) Full	10 or 11	Passive	1	0	OFF	ON	OFF	ON
(Dual) Full	10 or 11	Active	1	1	OFF	ON	ON	OFF
(Dual) Full	10 or 11	Passive	1	1	OFF	ON	OFF	ON

11.4.1.3.5 Dither Bit (DITH)

The purpose of the dither mode is to increase the minimum length of output pulses without decreasing the PWM resolution, in order to limit the pulse distortion introduced by the slew rate control of the outputs. If dither mode is selected the output pattern will repeat after two timer counter overflows. For the same output frequency, the shortest output pulse will have twice the length while dither feature is selected. To achieve the same output frame frequency, the prescaler of the MC10B12C module has to be set to twice the division rate if dither mode is selected; e.g., with the same prescaler division rate the repeat rate of the output pattern is the same as well as the shortest output pulse with or without dither mode selected.

The DITH bit in control register 0 enables or disables the dither function.

DITH = 0: dither function is disabled.

When DITH is cleared and assuming left aligned operation and RECIRC = 0, the PWM output will start at a logic low level at the beginning of the PWM period (motor controller timer counter = 0x000). The PWM output remains low until the motor controller timer counter matches the 11-bit PWM duty cycle value, DUTY, contained in D[10:0] in MCDCx. When a match (output compare between motor controller timer counter and DUTY) occurs, the PWM output will toggle to a logic high level and will remain at a logic high level until the motor controller timer counter overflows (reaches the contents of MCPER – 1). After the motor controller timer counter resets to 0x000, the PWM output will return to a logic low level. This completes one PWM period. The PWM period repeats every P counts (as defined by the bits P[10:0] in the motor controller period register) of the motor controller timer counter. If DUTY >= P, the output will be static low. If DUTY = 0x0000, the output will be continuously at a logic high level. The relationship between the motor controller timer counter clock, motor controller timer counter value, and PWM output while DITH = 0 is shown in Figure 11-17.

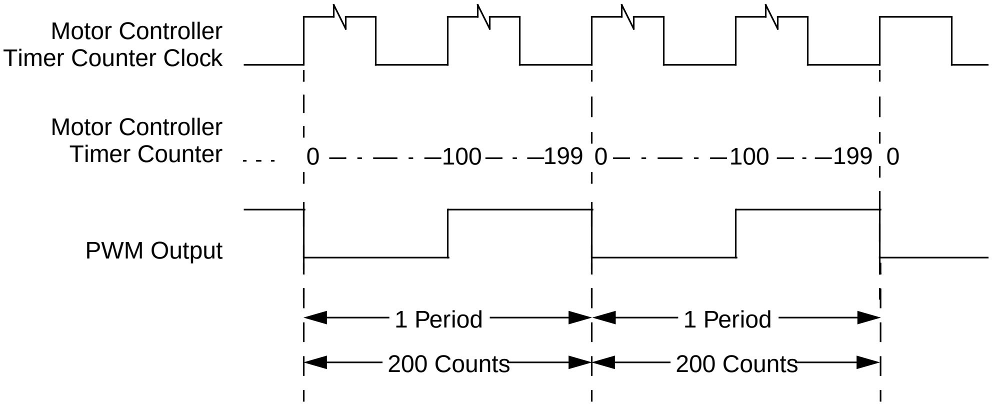


Figure 11-17. PWM Output: DITH = 0, MCAM[1:0] = 01, MCDC = 100, MCPER = 200, RECIRC = 0

DITH = 1: dither function is enabled

Please note if DITH = 1, the bit P0 in the motor controller period register will be internally forced to 0 and read always as 0.

When DITH is set and assuming left aligned operation and RECIRC = 0, the PWM output will start at a logic low level at the beginning of the PWM period (when the motor controller timer counter = 0x000). The PWM output remains low until the motor controller timer counter matches the 10-bit PWM duty cycle value, DUTY, contained in D[10:1] in MCDCx. When a match (output compare between motor controller timer counter and DUTY) occurs, the PWM output will toggle to a logic high level and will remain at a logic high level until the motor controller timer counter overflows (reaches the value defined by P[10:1] – 1 in MCPER). After the motor controller timer counter resets to 0x000, the PWM output will return to a logic low level. This completes the first half of the PWM period. During the second half of the PWM period, the PWM output will remain at a logic low level until either the motor controller timer counter matches the 10-bit PWM duty cycle value, DUTY, contained in D[10:1] in MCDCx if D0 = 0, or the motor controller timer counter matches the 10-bit PWM duty cycle value + 1 (the value of D[10:1] in MCDCx is increment by 1 and is compared with the motor controller timer counter value) if D0 = 1 in the corresponding duty cycle register. When a match occurs, the PWM output will toggle to a logic high level and will remain at a logic high level until the motor controller timer counter overflows (reaches the value defined by P[10:1] – 1 in MCPER). After the motor controller timer counter resets to 0x000, the PWM output will return to a logic low level.

This process will repeat every number of counts of the motor controller timer counter defined by the period register contents (P[10:0]). If the output is neither set to 0% nor to 100% there will be four edges on the PWM output per PWM period in this case. Therefore, the PWM output compare function will alternate between DUTY and DUTY + 1 every half PWM period if D0 in the corresponding duty cycle register is set to 1. The relationship between the motor controller timer counter clock (f_{TC}), motor controller timer counter value, and left aligned PWM output if DITH = 1 is shown in Figure 11-18 and Figure 11-19. Figure 11-20 and Figure 11-21 show right aligned and center aligned PWM operation respectively, with dither feature enabled and D0 = 1. Please note: In the following examples, the MCPER value is defined by the bits P[10:0], which is, if DITH = 1, always an even number.

NOTE

The DITH bit must be changed only if the motor controller is disabled (all channels disabled or period register cleared) to avoid erroneous waveforms.

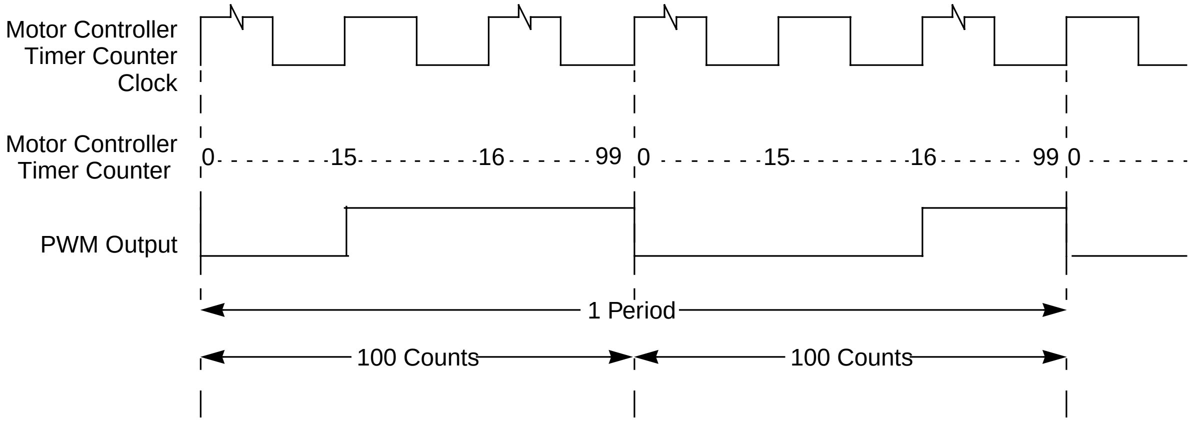


Figure 11-18. PWM Output: DITH = 1, MCAM[1:0] = 01, MCDC = 31, MCPER = 200, RECIRC = 0

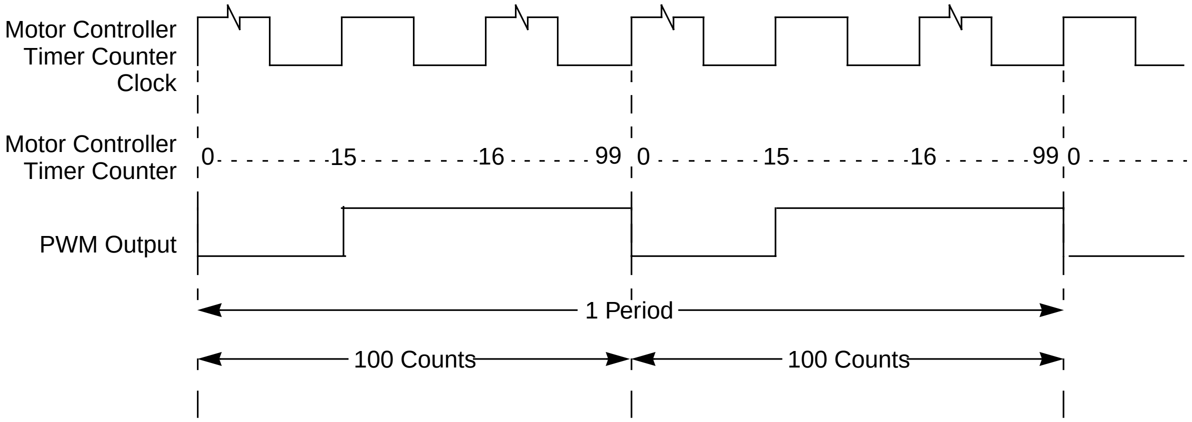


Figure 11-19. PWM Output: DITH = 1, MCAM[1:0] = 01, MCDC = 30, MCPER = 200, RECIRC = 0

.

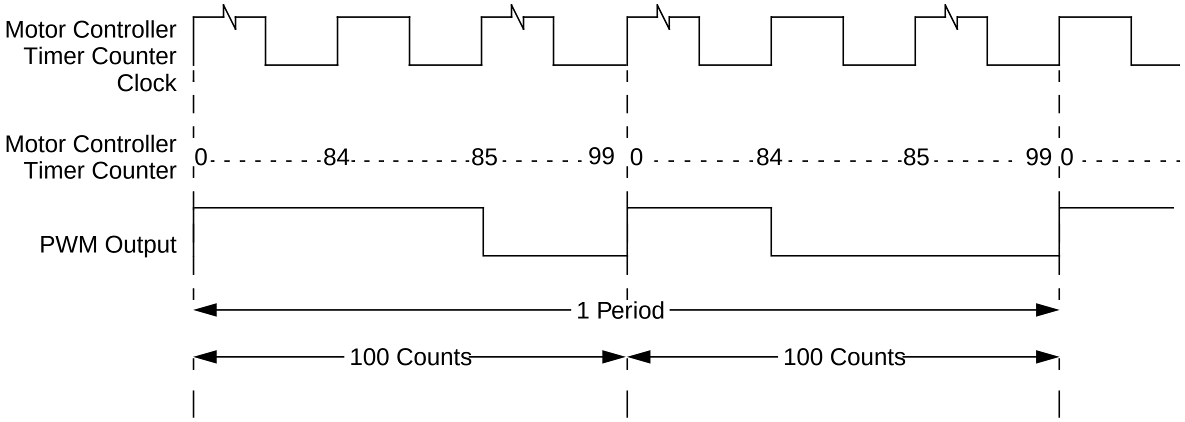


Figure 11-20. PWM Output: DITH = 1, MCAM[1:0] = 10, MCDC = 31, MCPER = 200, RECIRC = 0

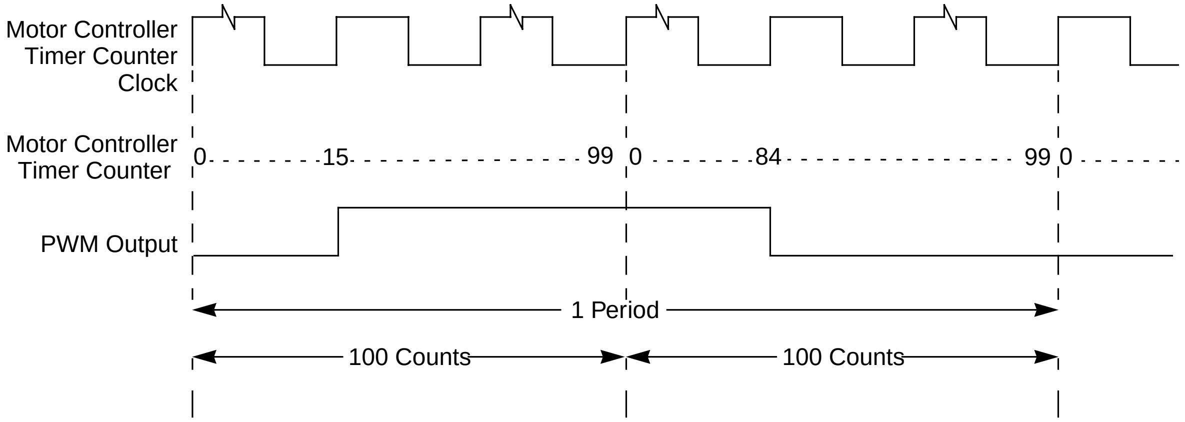


Figure 11-21. PWM Output: DITH = 1, MCAM[1:0] = 11, MCDC = 31, MCPER = 200, RECIRC = 0

11.4.2 PWM Duty Cycle

The PWM duty cycle for the motor controller channel x can be determined by dividing the decimal representation of bits D[10:0] in MCDCx by the decimal representation of the bits P[10:0] in MCPER and multiplying the result by 100% as shown in the equation below:

$$\text{Effective PWM Channel X \% Duty Cycle} = \frac{\text{DUTY}}{\text{MCPER}} \cdot 100\%$$

NOTE

x = PWM Channel Number = 0, 1, 2, 3 ... 11. This equation is only valid if DUTY <= MCPER and MCPER is not equal to 0.

Whenever D[10:0] >= P[10:0], a constant low level (RECIRC = 0) or high level (RECIRC = 1) will be output.

11.4.3 Motor Controller Counter Clock Source

Figure 11-22 shows how the PWM motor controller timer counter clock source is selected.

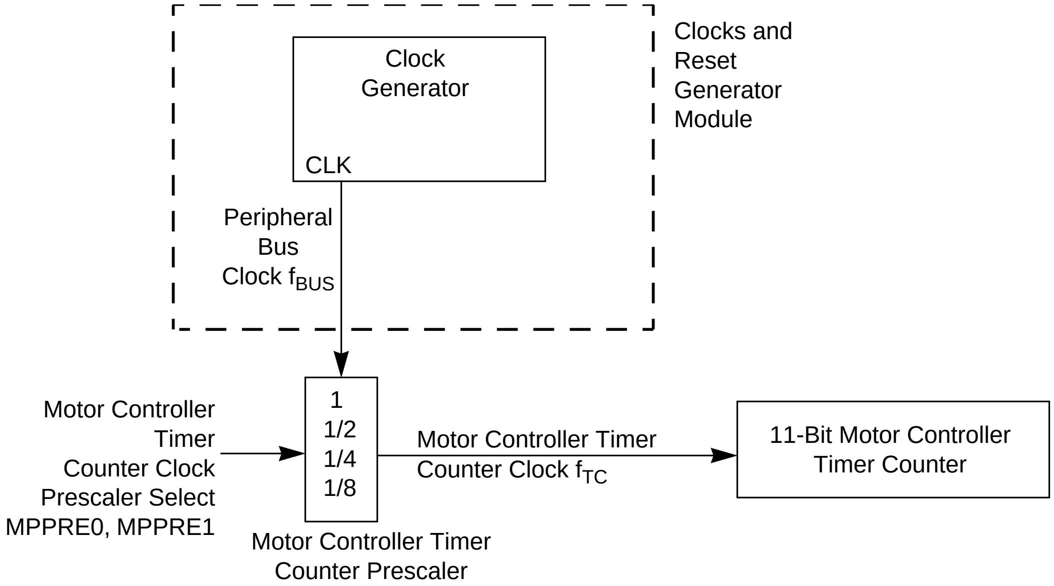


Figure 11-22. Motor Controller Counter Clock Selection

The peripheral bus clock is the source for the motor controller counter prescaler. The motor controller counter clock rate, f_{TC}, is set by selecting the appropriate prescaler value. The prescaler is selected with the MCPRE[1:0] bits in motor controller control register 0 (MCCTL0). The motor controller channel frequency of operation can be calculated using the following formula if DITH = 0:

$$\text{Motor Channel Frequency (Hz)} = \frac{f_{TC}}{\text{MCPER} \cdot \text{M}}$$

The motor controller channel frequency of operation can be calculated using the following formula if DITH = 1:

$$\text{Motor Channel Frequency (Hz)} = \frac{f_{TC}}{\text{MCPER} \cdot M/2}$$

NOTE

Both equations are only valid if MCPER is not equal to 0. M = 1 for left or right aligned mode, M = 2 for center aligned mode.

Table 11-14 shows examples of the motor controller channel frequencies that can be generated based on different peripheral bus clock frequencies and the prescaler value.

Table 11-14. Motor Controller Channel Frequencies (Hz), MCPER = 256, DITH = 0, MCAM = 10, 01

Prescaler	Peripheral Bus Clock Frequency				
	16 MHz	10 MHz	8 MHz	5 MHz	4 MHz
1	62500	39063	31250	19531	15625
1/2	31250	19531	15625	9766	7813
1/4	15625	9766	7813	4883	3906
1/8	7813	4883	3906	2441	1953

NOTE

Due to the selectable slew rate control of the outputs, clipping may occur on short output pulses.

11.4.4 Output Switching Delay

In order to prevent large peak current draw from the motor power supply, selectable delays can be used to stagger the high logic level to low logic level transitions on the motor controller outputs. The timing delay, t_d, is determined by the CD[1:0] bits in the corresponding channel control register (MCMCx) and is selectable between 0, 1, 2, or 3 motor controller timer counter clock cycles.

NOTE

A PWM channel gets disabled at the next timer counter overflow without notice of the switching delay.

11.4.5 Operation in Wait Mode

During wait mode, the operation of the motor controller pins are selectable between the following two options:

1. MCSWAI = 1: All module clocks are stopped and the associated port pins are set to their inactive state, which is defined by the state of the RECIRC bit during wait mode. The motor controller module registers stay the same as they were prior to entering wait mode. Therefore, after exiting from wait mode, the associated port pins will resume to the same functionality they had prior to entering wait mode.
2. MCSWAI = 0: The PWM clocks continue to run and the associated port pins maintain the functionality they had prior to entering wait mode both during wait mode and after exiting wait mode.

11.4.6 Operation in Stop and Pseudo-Stop Modes

All module clocks are stopped and the associated port pins are set to their inactive state, which is defined by the state of the RECIRC bit. The motor controller module registers stay the same as they were prior to entering stop or pseudo-stop modes. Therefore, after exiting from stop or pseudo-stop modes, the associated port pins will resume to the same functionality they had prior to entering stop or pseudo-stop modes.

11.5 Reset

The motor controller is reset by system reset. All associated ports are released, all registers of the motor controller module will switch to their reset state as defined in Section 11.3.2, “Register Descriptions”.

11.6 Interrupts

The motor controller has one interrupt source.

11.6.1 Timer Counter Overflow Interrupt

An interrupt will be requested when the MCTOIE bit in the motor controller control register 1 is set and the running PWM frame is finished. The interrupt is cleared by either setting the MCTOIE bit to 0 or to write a 1 to the MCTOIF bit in the motor controller control register 0.

11.7 Initialization/Application Information

This section provides an example of how the PWM motor controller can be initialized and used by application software. The configuration parameters (e.g., timer settings, duty cycle values, etc.) are not guaranteed to be adequate for any real application.

The example software is implemented in assembly language.

11.7.1 Code Example

One way to use the motor controller is:

1. Perform global initialization
    a) Set the motor controller control registers MCCTL0 and MCCTL1 to appropriate values.
        i) Prescaler disabled (MCPRE1 = 0, MCPRE0 = 0).
        ii) Fast mode and dither disabled (FAST = 0, DITH = 0).
        iii) Recirculation feature in dual full H-bridge mode disabled (RECIRC = 0).

        All other bits in MCCTL0 and MCCTL1 are set to 0.
    b) Configure the channel control registers for the desired mode.
        i) Dual full H-bridge mode (MCOM[1:0] = 11).
        ii) Left aligned PWM (MCAM[1:0] = 01).
        iii) No channel delay (MCCD[1:0] = 00).
2. Perform the startup phase
    a) Clear the duty cycle registers MCDC0 and MCDC1
    b) Initialize the period register MCPER, which is equivalent to enabling the motor controller.
    c) Enable the timer which generates the timebase for the updates of the duty cycle registers.
3. Main program
    a) Check if pin PB0 is set to “1” and execute the sub program if a timer interrupt is pending.
    b) Initiate the shutdown procedure if pin PB0 is set to “0”.
4. Sub program
    a) Update the duty cycle registers

        Load the duty cycle registers MCDC0 and MCDC1 with new values from the table and clear the timer interrupt flag.

        The sub program will initiate the shutdown procedure if pin PB0 is set to “0”.
    b) Shutdown procedure

The timer is disabled and the duty cycle registers are cleared to drive an inactive value on the PWM output as long as the motor controller is enabled. The period register is cleared after a certain time, which disables the motor controller. The table address is restored and the timer interrupt flag is cleared.

```
;------------------------------------------------------------------------------------------
; Motor Controller (MC10B8C) setup example
;------------------------------------------------------------------------------------------
; Timer defines
;------------------------------------------------------------------------------------------
T_START                 EQU  $0040
TSCR1                   EQU  T_START+$06
TFLG2                   EQU  T_START+$0F
;------------------------------------------------------------------------------------------
; Motor Controller defines
;------------------------------------------------------------------------------------------
MC_START                EQU  $0200
MCCTL0                  EQU  MC_START+$00
MCCTL1                  EQU  MC_START+$01
MCPER_HI                EQU  MC_START+$02
MCPER_LO                EQU  MC_START+$03
MCCC0                   EQU  MC_START+$10
MCCC1                   EQU  MC_START+$11
MCCC2                   EQU  MC_START+$12
MCCC3                   EQU  MC_START+$13
MCDC0_HI                EQU  MC_START+$20
MCDC0_LO                EQU  MC_START+$21
MCDC1_HI                EQU  MC_START+$22
MCDC1_LO                EQU  MC_START+$23
MCDC2_HI                EQU  MC_START+$24
MCDC2_LO                EQU  MC_START+$25
MCDC3_HI                EQU  MC_START+$26
MCDC3_LO                EQU  MC_START+$27
;------------------------------------------------------------------------------------------
; Port defines
;------------------------------------------------------------------------------------------
DDRB                    EQU  $0003
PORTB                   EQU  $0001
;------------------------------------------------------------------------------------------
; Flash defines
;------------------------------------------------------------------------------------------
FLASH_START             EQU  $0100
FCMD                    EQU  FLASH_START+$06
FCLKDIV                 EQU  FLASH_START+$00
FSTAT                   EQU  FLASH_START+$05
FTSTMOD                 EQU  FLASH_START+$02
; Variables
CODE_START              EQU  $1000          ; start of program code
DTYDAT                  EQU  $1500          ; start of motor controller duty cycle data
TEMP_X                  EQU  $1700          ; save location for IX reg in ISR
TABLESIZE               EQU  $1704          ; number of config entries in the table
MCPERIOD                EQU  $0250          ; motor controller period
;------------------------------------------------------------------------------------------
;------------------------------------------------------------------------------------------
        ORG        CODE_START               ; start of code
          LDS     #$1FFF                    ; set stack pointer
          MOVW    #$000A,TABLESIZE          ; number of configurations in the table
          MOVW    TABLESIZE,TEMP_X
```

```
;-----------------------------------------------------------------------------------------
;global motor controller init
;-----------------------------------------------------------------------------------------
GLB_INIT: MOVB    #$0000,MCCTL0             ; fMC = fBUS, FAST=0, DITH=0
          MOVB    #$0000,MCCTL1             ; RECIRC=0, MCTOIE=0
          MOVW    #$D0D0,MCCC0              ; dual full h-bridge mode, left aligned,
                                            ; no channel delay
          MOVW    #$0000,MCPER_HI           ; disable motor controller
;-----------------------------------------------------------------------------------------
;motor controller startup
;-----------------------------------------------------------------------------------------
STARTUP:
          MOVW    #$0000,MCDC0_HI           ; define startup duty cycles
          MOVW    #$0000,MCDC1_HI
          MOVW    #MCPERIOD,MCPER_HI        ; define PWM period
          MOVB    #$80,TSCR1                ; enable timer
MAIN:     LDAA    PORTB                     ; if PB=0, activate shutdown
          ANDA    #$01
          BEQ     MN0
          JSR     TIM_SR
MN0:      TST     TFLG2                     ; poll for timer counter overflow flag
          BEQ     MAIN                      ; TOF set?
          JSR     TIM_SR                    ; yes, go to TIM_SR
          BRA     MAIN
TIM_SR:   LDX     TEMP_X                    ; restore index register X
          LDAA    PORTB                     ; if PB=0, enter shutdown routine
          ANDA    #$01
          BNE     SHUTDOWN
          LDX     TEMP_X                    ; restore index register X
          BEQ     NEW_SEQ                   ; all mc configurations done?
NEW_CFG:  LDD     DTYDAT,X                  ; load new config’s
          STD     MCDC0_HI
          DEX
          DEX
          LDD     DTYDAT,X
          STD     MCDC1_HI
          BRA     END_SR                    ; leave sub-routine
SHUTDOWN: MOVB    #$00,TSCR1                ; disable timer
          MOVW    #$0000,MCDC0_HI           ; define startup duty cycle
          MOVW    #$0000,MCDC1_HI           ; define startup duty cycle
          LDAA    #$0000                    ; ensure that duty cycle registers are
                                            ; cleared for some time before disabling
                                            ; the motor controller
LOOP      DECA
          BNE     LOOP
          MOVW    #$0000,MCPER_HI           ; define pwm period
NEW_SEQ:  MOVW    TABLESIZE,TEMP_X          ; start new tx loop
          LDX     TEMP_X
END_SR:   STX     TEMP_X                    ; save byte counter
          MOVB    #$80,TFLG2                ; clear TOF
          RTS                               ; wait for new timer overflow
```

```
;-----------------------------------------------------------------------------------------
; motor controller duty cycles
;-----------------------------------------------------------------------------------------
        org     DTYDAT
          DC.B    $02, $FF[1]; MCDC1_HI, MCDC1_LO
          DC.B    $02, $D0 ; MCDC0_HI, MCDC0_LO
          DC.B    $02, $A0 ; MCDC1_HI, MCDC1_LO
          DC.B    $02, $90 ; MCDC0_HI, MCDC0_LO
          DC.B    $02, $60 ; MCDC1_HI, MCDC1_LO
          DC.B    $02, $25 ; MCDC0_HI, MCDC0_LO
```

---

[1] The values for the duty cycle table have to be defined for the needs of the target application.

Chapter 12
Stepper Stall Detector (SSDV1) Block Description

12.1 Introduction

The stepper stall detector (SSD) block provides a circuit to measure and integrate the induced voltage on the non-driven coil of a stepper motor using full steps when the gauge pointer is returning to zero (RTZ). During the RTZ event, the pointer is returned to zero using full steps in either clockwise or counter clockwise direction, where only one coil is driven at any point in time. The back electromotive force (EMF) signal present on the non-driven coil is integrated after a blanking time, and its results stored in a 16-bit accumulator. The 16-bit modulus down counter can be used to monitor the blanking time and the integration time. The value in the accumulator represents the change in linked flux (magnetic flux times the number of turns in the coil) and can be compared to a stored threshold. Values above the threshold indicate a moving motor, in which case the pointer can be advanced another full step in the same direction and integration be repeated. Values below the threshold indicate a stalled motor, thereby marking the cessation of the RTZ event. The SSD is capable of multiplexing two stepper motors.

12.1.1 Modes of Operation

- Return to zero modes
  - Blanking with no drive
  - Blanking with drive
  - Conversion
  - Integration
- Low-power modes

12.1.2 Features

- Programmable full step state
- Programmable integration polarity
- Blanking (recirculation) state
- 16-bit integration accumulator register
- 16-bit modulus down counter with interrupt
- Multiplex two stepper motors

12.1.3 Block Diagram

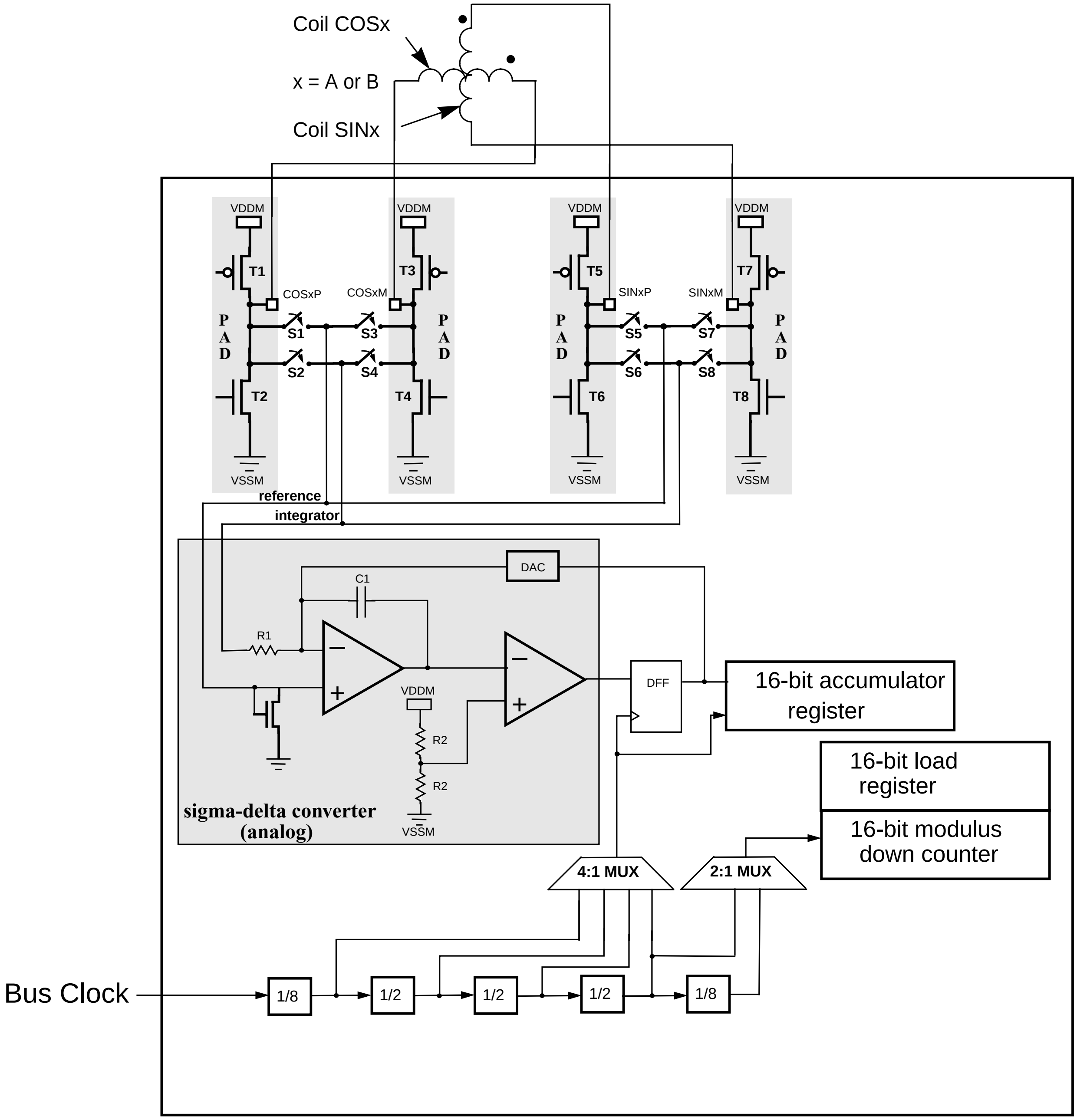


Figure 12-1. SSD Block Diagram

12.2 External Signal Description

Each SSD signal is the output pin of a half bridge, designed to source or sink current. The H-bridge pins drive the sine and cosine coils of a stepper motor to provide four-quadrant operation. The SSD is capable of multiplexing between stepper motor A and stepper motor B if two motors are connected.

Table 12-1. Pin Table[1]

Pin Name	Node	Coil
COSxM	Minus	COSx
COSxP	Plus	
SINxM	Minus	SINx
SINxP	Plus	

[1] x = A or B indicating motor A or motor B

12.2.1 COSxM/COSxP — Cosine Coil Pins for Motor x

These pins interface to the cosine coils of a stepper motor to measure the back EMF for calibration of the pointer reset position.

12.2.2 SINxM/SINxP — Sine Coil Pins for Motor x

These pins interface to the sine coils of a stepper motor to measure the back EMF for calibration of the pointer reset position.

12.3 Memory Map and Register Definition

This section provides a detailed description of all registers of the stepper stall detector (SSD) block.

12.3.1 Module Memory Map

Table 12-2 gives an overview of all registers in the SSDV1 memory map. The SSDV1 occupies eight bytes in the memory space. The register address results from the addition of *base address* and *address offset.* The *base address* is determined at the MCU level and is given in the Device Overview chapter. The *address offset* is defined at the block level and is given here.

Table 12-2. SSDV1 Memory Map

Address Offset	Use	Access
0x0000	RTZCTL	R/W
0x0001	MDCCTL	R/W
0x0002	SSDCTL	R/W
0x0003	SSDFLG	R/W
0x0004	MDCCNT (High)	R/W
0x0005	MDCCNT (Low)	R/W
0x0006	ITGACC (High)	R
0x0007	ITGACC (Low)	R

12.3.2 Register Descriptions

This section describes in detail all the registers and register bits in the SSDV1 block. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

12.3.2.1 Return-to-Zero Control Register (RTZCTL)

Module Base + 0x0000

	7	6	5	4	3	2	1	0
R	ITG	DCOIL	RCIR	POL	0	SMS	STEP	
W					▨			
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 12-2. Return-to-Zero Control Register (RTZCTL)

Read: anytime

Write: anytime

Table 12-3. RTZCTL Field Descriptions

Field	Description
7 ITG	**Integration** — During return to zero (RTZE = 1), one of the coils must be recirculated or non-driven determined by the STEP field. If the ITG bit is set, the coil is non-driven, and if the ITG bit is clear, the coil is being recirculated. Table 12-4 shows the condition state of each transistor from Figure 12-1 based on the STEP, ITG, DCOIL and RCIR bits. Regardless of the RTZE bit value, if the ITG bit is set, one end of the non-driven coil connects to the (non-zero) reference input and the other end connects to the integrator input of the sigma-delta converter. Regardless of the RTZE bit value, if the ITG bit is clear, the non-driven coil is in a blanking state, the converter is in a reset state, and the accumulator is initialized to zero. Table 12-5 shows the condition state of each switch from Figure 12-1 based on the ITG, STEP and POL bits. 0 Blanking 1 Integration
6 DCOIL	**Drive Coil** — During return to zero (RTZE=1), one of the coils must be driven determined by the STEP field. If the DCOIL bit is set, this coil is driven. If the DCOIL bit is clear, this coil is disconnected or drivers turned off. Table 12-4 shows the condition state of each transistor from Figure 12-1 based on the STEP, ITG, DCOIL and RCIR bits. 0 Disconnect Coil 1 Drive Coil
5 RCIR	**Recirculation in Blanking Mode** — During return to zero (RTZE = 1), one of the coils is recirculated prior to integration during the blanking period. This bit determines if the coil is recirculated via VDDM or via VSSM. Table 12-4 shows the condition state of each transistor from Figure 12-1 based on the STEP, ITG, DCOIL and RCIR bits. 0 Recirculation on the high side transistors 1 Recirculation on the low side transistors
4 POL	**Polarity** — This bit determines which end of the non-driven coil is routed to the sigma-delta converter during conversion or integration mode. Table 12-5 shows the condition state of each switch from Figure 12-1 based on the ITG, STEP and POL bits.

Table 12-3. RTZCTL Field Descriptions (continued)

Field	Description
2 SMS	**Stepper Motor Select** — This bit selects one of two possible stepper motors to be used for stall detection. See top level chip description for the stepper motor assignments to the SSD. 0 Stepper Motor A is selected for stall detection 1 Stepper Motor B is selected for stall detection
1:0 STEP	**Full Step State** — This field indicates one of the four possible full step states. Step 0 is considered the east pole or 0° angle, step 1 is the north Pole or 90° angle, step 2 is the west pole or 180° angle, and step 3 is the south pole or 270° angle. For each full step state, Table 12-6 shows the current through each of the two coils, and the coil nodes that are multiplexed to the sigma-delta converter during conversion or integration mode.

Table 12-4. Transistor Condition States (RTZE = 1)

STEP	ITG	DCOIL	RCIR	T1	T2	T3	T4	T5	T6	T7	T8
xx	1	0	x	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
00	0	0	0	OFF	OFF	OFF	OFF	ON	OFF	ON	OFF
00	0	0	1	OFF	OFF	OFF	OFF	OFF	ON	OFF	ON
00	0	1	0	ON	OFF	OFF	ON	ON	OFF	ON	OFF
00	0	1	1	ON	OFF	OFF	ON	OFF	ON	OFF	ON
00	1	1	x	ON	OFF	OFF	ON	OFF	OFF	OFF	OFF
01	0	0	0	ON	OFF	ON	OFF	OFF	OFF	OFF	OFF
01	0	0	1	OFF	ON	OFF	ON	OFF	OFF	OFF	OFF
01	0	1	0	ON	OFF	ON	OFF	ON	OFF	OFF	ON
01	0	1	1	OFF	ON	OFF	ON	ON	OFF	OFF	ON
01	1	1	x	OFF	OFF	OFF	OFF	ON	OFF	OFF	ON
10	0	0	0	OFF	OFF	OFF	OFF	ON	OFF	ON	OFF
10	0	0	1	OFF	OFF	OFF	OFF	OFF	ON	OFF	ON
10	0	1	0	OFF	ON	ON	OFF	ON	OFF	ON	OFF
10	0	1	1	OFF	ON	ON	OFF	OFF	ON	OFF	ON
10	1	1	x	OFF	ON	ON	OFF	OFF	OFF	OFF	OFF
11	0	0	0	ON	OFF	ON	OFF	OFF	OFF	OFF	OFF
11	0	0	1	OFF	ON	OFF	ON	OFF	OFF	OFF	OFF
11	0	1	0	ON	OFF	ON	OFF	OFF	ON	ON	OFF
11	0	1	1	OFF	ON	OFF	ON	OFF	ON	ON	OFF
11	1	1	x	OFF	OFF	OFF	OFF	OFF	ON	ON	OFF

Table 12-5. Switch Condition States (RTZE = 1 or 0)

ITG	STEP	POL	S1	S2	S3	S4	S5	S6	S7	S8
0	xx	x	Open	Open	Open	Open	Open	Open	Open	Open
1	00	0	Open	Open	Open	Open	Close	Open	Open	Close
1	00	1	Open	Open	Open	Open	Open	Close	Close	Open
1	01	0	Open	Close	Close	Open	Open	Open	Open	Open
1	01	1	Close	Open	Open	Close	Open	Open	Open	Open
1	10	0	Open	Open	Open	Open	Open	Close	Close	Open
1	10	1	Open	Open	Open	Open	Close	Open	Open	Close
1	11	0	Close	Open	Open	Close	Open	Open	Open	Open
1	11	1	Open	Close	Close	Open	Open	Open	Open	Open

Table 12-6. Full Step States

STEP	Pole	Angle	COSINE Coil Current		SINE Coil Current		Coil Node to Integrator input (Close Switch)		Coil Node to Reference input (Close Switch)	
			DCOIL = 0	DCOIL = 1	DCOIL = 0	DCOIL = 1	ITG = 1 POL = 0	ITG = 1 POL = 1	ITG = 1 POL = 0	ITG = 1 POL = 1
0	East	0°	0	+ I max	0	0	SINxM (S8)	SINxP (S6)	SINxP (S5)	SINxM (S7)
1	North	90°	0	0	0	+ I max	COSxP (S2)	COSxM (S4)	COSxM (S3)	COSxP (S1)
2	West	180°	0	– I max	0	0	SINxP (S6)	SINxM (S8)	SINxM (S7)	SINxP (S5)
3	South	270°	0	0	0	– I max	COSxM (S4)	COSxP (S2)	COSxP (S1)	COSxM (S3)

12.3.2.2 Modulus Down Counter Control Register (MDCCTL)

Module Base + 0x0001

	7	6	5	4	3	2	1	0
R	MCZIE	MODMC	RDMCL	PRE	0	MCEN	0	AOVIE
W					FLMC		▨	
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 12-3. Modulus Down Counter Control Register (MDCCTL)

Read: anytime

Write: anytime.

Table 12-7. MDCCTL Field Descriptions

Field	Description
7 MCZIE	**Modulus Counter Underflow Interrupt Enable** 0 Interrupt disabled. 1 Interrupt enabled. An interrupt will be generated when the modulus counter underflow interrupt flag (MCZIF) is set.
6 MODMC	**Modulus Mode Enable** 0 The modulus counter counts down from the value in the counter register and will stop at 0x0000. 1 Modulus mode is enabled. When the counter reaches 0x0000, the counter is loaded with the latest value written to the modulus counter register. **Note:** For proper operation, the MCEN bit should be cleared before modifying the MODMC bit in order to reset the modulus counter to 0xFFFF.
5 RDMCL	**Read Modulus Down-Counter Load** 0 Reads of the modulus count register (MDCCNT) will return the present value of the count register. 1 Reads of the modulus count register (MDCCNT) will return the contents of the load register.
4 PRE	**Prescaler** 0 The modulus down counter clock frequency is the bus frequency divided by 64. 1 The modulus down counter clock frequency is the bus frequency divided by 512. **Note:** A change in the prescaler division rate will not be effective until a load of the load register into the modulus counter count register occurs.
3 FLMC	**Force Load Register into the Modulus Counter Count Register** — This bit always reads zero. 0 Write zero to this bit has no effect. 1 Write one into this bit loads the load register into the modulus counter count register.
2 MCEN	**Modulus Down-Counter Enable** 0 Modulus down-counter is disabled. The modulus counter (MDCCNT) is preset to 0xFFFF. This will prevent an early interrupt flag when the modulus down-counter is enabled. 1 Modulus down-counter is enabled.
0 AOVIE	**Accumulator Overflow Interrupt Enable** 0 Interrupt disabled. 1 Interrupt enabled. An interrupt will be generated when the accumulator overflow interrupt flag (AOVIF) is set.

12.3.2.3 Stepper Stall Detector Control Register (SSDCTL)

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R	RTZE	SDCPU	SSDWAI	FTST	0	0	ACLKS	
W					▨	▨		
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 12-4. Stepper Stall Detector Control Register (SSDCTL)

Read: anytime

Write: anytime

Table 12-8. SSDCTL Field Descriptions

Field	Description
7 RTZE	**Return to Zero Enable** — If this bit is set, the coils are controlled by the SSD and are configured into one of the four full step states as shown in Table 12-6. If this bit is cleared, the coils are not controlled by the SSD. 0 RTZ is disabled. 1 RTZ is enabled.
6 SDCPU	**Sigma-Delta Converter Power Up** — This bit provides on/off control for the sigma-delta converter allowing reduced MCU power consumption. Because the analog circuit is turned off when powered down, the sigma-delta converter requires a recovery time after it is powered up. 0 Sigma-delta converter is powered down. 1 Sigma-delta converter is powered up.
5 SSDWAI	**SSD Disabled during Wait Mode** — When entering Wait Mode, this bit provides on/off control over the SSD allowing reduced MCU power consumption. Because the analog circuit is turned off when powered down, the sigma-delta converter requires a recovery time after exit from Wait Mode. 0 SSD continues to run in WAIT mode. 1 Entering WAIT mode freezes the clock to the prescaler divider, powers down the sigma-delta converter, and if RTZE bit is set, the sine and cosine coils are recirculated via VSSM.
4 FTST	**Factory Test** — This bit is reserved for factory test and reads zero in user mode.
1:0 ACLKS	**Accumulator Sample Frequency Select** — This field sets the accumulator sample frequency by pre-scaling the bus frequency by a factor of 8, 16, 32, or 64. A faster sample frequency can provide more accurate results but cause the accumulator to overflow. Best results are achieved with a frequency between 500 kHz and 2 MHz. Accumulator Sample Frequency = f_{BUS} / (8 x 2^{ACLKS})

Table 12-9. Accumulator Sample Frequency

ACLKS	Frequency	f_{BUS} = 40 MHz	f_{BUS} = 25 MHz	f_{BUS} = 16 MHz
0	f_{BUS} / 8	5.00 MHz	3.12 MHz	2.00 MHz
1	f_{BUS} / 16	2.50 MHz	1.56 MHz	1.00 MHz
2	f_{BUS} / 32	1.25 MHz	781 kHz	500 kHz
3	f_{BUS} / 64	625 kHz	391 kHz	250 kHz

NOTE

A change in the accumulator sample frequency will not be effective until the ITG bit is cleared.

12.3.2.4 Stepper Stall Detector Flag Register (SSDFLG)

Module Base + 0x0003

	7	6	5	4	3	2	1	0
R	MCZIF	0	0	0	0	0	0	AOVIF
W		▨	▨	▨	▨	▨	▨	
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 12-5. Stepper Stall Detector Flag Register (SSDFLG)

Read: anytime

Write: anytime.

Table 12-10. SSDFLG Field Descriptions

Field	Description
7 MCZIF	**Modulus Counter Underflow Interrupt Flag** — This flag is set when the modulus down-counter reaches 0x0000. If not masked (MCZIE = 1), a modulus counter underflow interrupt is pending while this flag is set. This flag is cleared by writing a ‘1’ to the bit. A write of ‘0’ has no effect.
0 AOVIF	**Accumulator Overflow Interrupt Flag** — This flag is set when the Integration Accumulator has a positive or negative overflow. If not masked (AOVIE = 1), an accumulator overflow interrupt is pending while this flag is set. This flag is cleared by writing a ‘1’ to the bit. A write of ‘0’ has no effect.

12.3.2.5 Modulus Down-Counter Count Register (MDCCNT)

Module Base + 0x0004

	15	14	13	12	11	10	9	8
R	MDCCNT							
W								
Reset	1	1	1	1	1	1	1	1

Figure 12-6. Modulus Down-Counter Count Register High (MDCCNT)

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R	MDCCNT							
W								
Reset	1	1	1	1	1	1	1	1

Figure 12-7. Modulus Down-Counter Count Register Low (MDCCNT)

Read: anytime

Write: anytime.

NOTE

> A separate read/write for high byte and low byte gives a different result than accessing the register as a word.

If the RDMCL bit in the MDCCTL register is cleared, reads of the MDCCNT register will return the present value of the count register. If the RDMCL bit is set, reads of the MDCCNT register will return the contents of the load register.

With a 0x0000 write to the MDCCNT register, the modulus counter stays at zero and does not set the MCZIF flag in the SSDFLG register.

If modulus mode is not enabled (MODMC = 0), a write to the MDCCNT register immediately updates the load register and the counter register with the value written to it. The modulus counter will count down from this value and will stop at 0x0000.

If modulus mode is enabled (MODMC = 1), a write to the MDCCNT register updates the load register with the value written to it. The count register will not be updated with the new value until the next counter underflow. The FLMC bit in the MDCCTL register can be used to immediately update the count register with the new value if an immediate load is desired.

The modulus down counter clock frequency is the bus frequency divided by 64 or 512.

12.3.2.6 Integration Accumulator Register (ITGACC)

Module Base + 0x0006

	15	14	13	12	11	10	9	8
R	ITGACC							
W								
Reset	0	0	0	0	0	0	0	0

Figure 12-8. Integration Accumulator Register High (ITGACC)

Module Base + 0x0007

	7	6	5	4	3	2	1	0
R	ITGACC							
W								
Reset	0	0	0	0	0	0	0	0

Figure 12-9. Integration Accumulator Register Low (ITGACC)

Read: anytime.

Write: Never.

NOTE

> A separate read for high byte and low byte gives a different result than accessing the register as a word.

This 16-bit field is signed and is represented in two's complement. It indicates the change in flux while integrating the back EMF present in the non-driven coil during a return to zero event.

When ITG is zero, the accumulator is initialized to 0x0000 and the sigma-delta converter is in a reset state.

When ITG is one, the accumulator increments or decrements depending on the sigma-delta conversion sample. The accumulator sample frequency is determined by the ACLKS field. The accumulator freezes at 0x7FFF on a positive overflow and freezes at 0x8000 on a negative overflow.

12.4 Functional Description

The stepper stall detector (SSD) has a simple control block to configure the H-bridge drivers of a stepper motor in four different full step states with four available modes during a return to zero event. The SSD has a detect circuit using a sigma-delta converter to measure and integrate changes in flux of the de-energized winding in the stepping motor and the conversion result is accumulated in a 16-bit signed register. The SSD also has a 16-bit modulus down counter to monitor blanking and integration times. DC offset compensation is implemented when using the modulus down counter to monitor integration times.

12.4.1 Return to Zero Modes

There are four return to zero modes as shown in Table 12-11.

Table 12-11. Return to Zero Modes

ITG	DCOIL	Mode
0	0	Blanking with no drive
0	1	Blanking with drive
1	0	Conversion
1	1	Integration

12.4.1.1 Blanking with No Drive

In blanking mode with no drive, one of the coils is masked from the sigma-delta converter, and if RTZ is enabled (RTZE = 1), it is set up to recirculate its current. If RTZ is enabled (RTZE = 1), the other coil is disconnected to prevent any loss of flux change that would occur when the motor starts moving before the end of recirculation and start of integration. In blanking mode with no drive, the accumulator is initialized to 0x0000 and the converter is in a reset state.

12.4.1.2 Blanking with Drive

In blanking mode with drive, one of the coils is masked from the sigma-delta converter, and if RTZ is enabled (RTZE = 1), it is set up to recirculate its current. If RTZ is enabled (RTZE = 1), the other coil is driven. In blanking mode with drive, the accumulator is initialized to 0x0000 and the converter is in a reset state.

12.4.1.3 Conversion

In conversion mode, one of the coils is routed for integration with one end connected to the (non-zero) reference input and the other end connected to the integrator input of the sigma-delta converter. If RTZ is enabled (RTZE=1), both coils are disconnected. This mode is not useful for stall detection.

12.4.1.4 Integration

In integration mode, one of the coils is routed for integration with one end connected to the (non-zero) reference input and the other end connected to the integrator input of the sigma-delta converter. If RTZ is enabled (RTZE = 1), the other coil is driven. This mode is used to rectify and integrate the back EMF produced by the coils to detect stepped rotary motion.

DC offset compensation is implemented when using the modulus down counter to monitor integration time.

12.4.2 Full Step States

During a return to zero (RTZ) event, the stepper motor pointer requires a 90° full motor electrical step with full amplitude pulses applied to each phase in turn. For counter clockwise rotation (CCW), the STEP value is incremented 0, 1, 2, 3, 0 and so on, and for a clockwise rotation the STEP value is decremented 3, 2, 1, 0 and so on. Figure 12-10 shows the current level through each coil for each full step in CCW rotation when DCOIL is set.

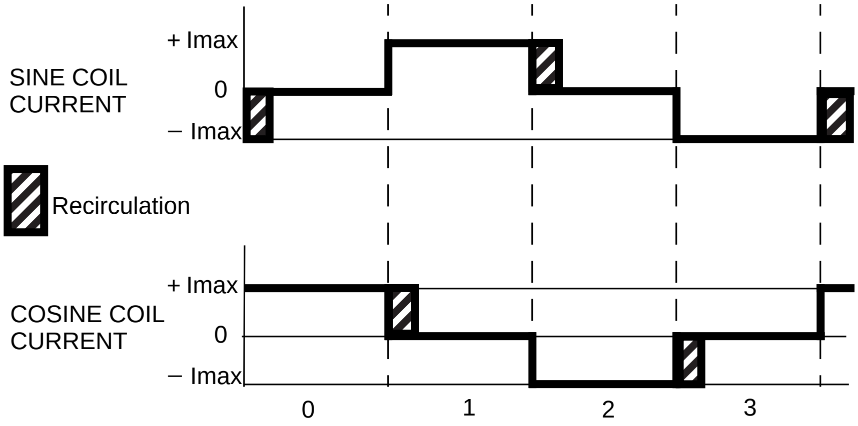


Figure 12-10. Full Steps (CCW)

Figure 12-11 shows the current flow in the SINx and COSx H-bridges when STEP = 0, DCOIL = 1, ITG = 0 and RCIR = 0.

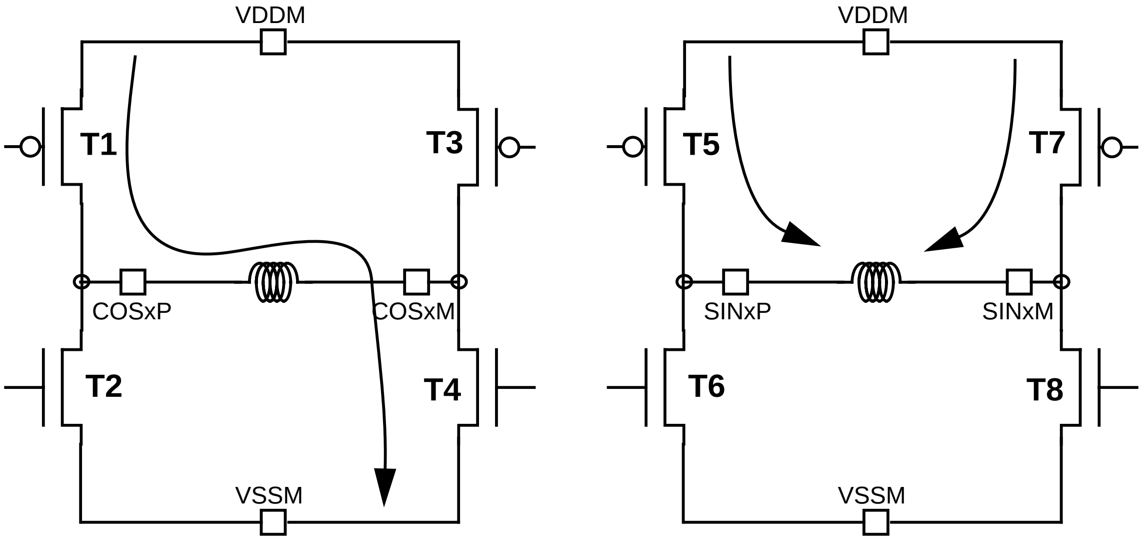


Figure 12-11. Current Flow when STEP = 0, DCOIL = 1, ITG = 0, RCIR = 0

Figure 12-12 shows the current flow in the SINx and COSx H-bridges when STEP = 1, DCOIL = 1, ITG = 0 and RCIR = 1.

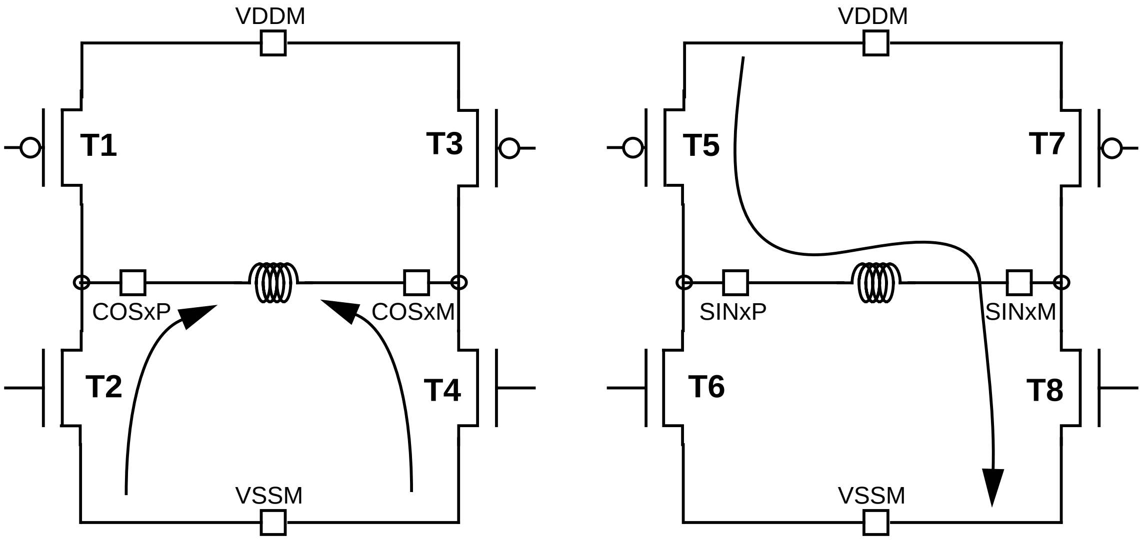


Figure 12-12. Current Flow when STEP = 1, DCOIL = 1, ITG = 0, RCIR = 1

Figure 12-13 shows the current flow in the SINx and COSx H-bridges when STEP = 2, DCOIL = 1 and ITG = 1.

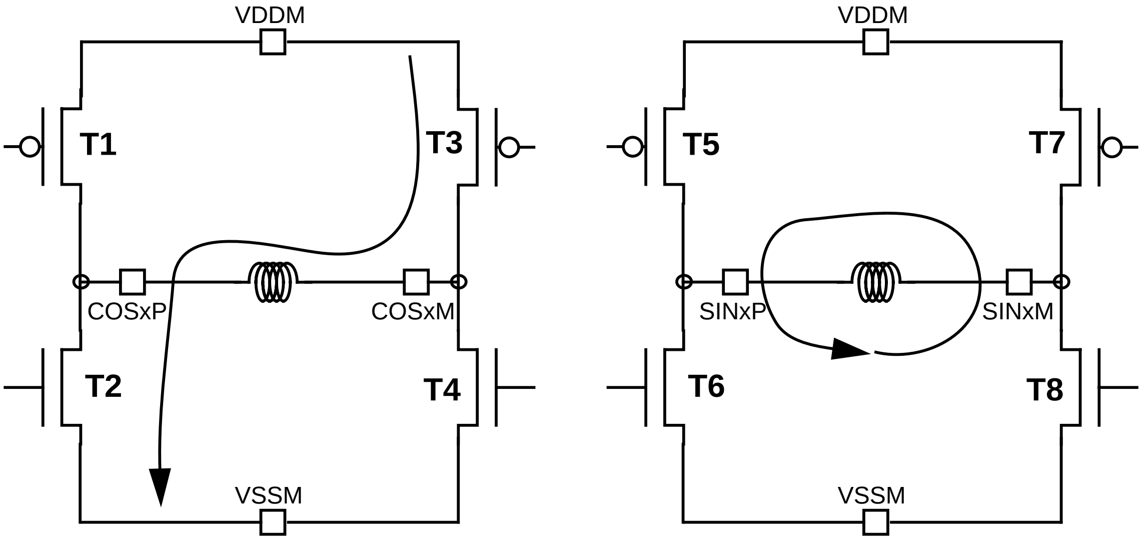


Figure 12-13. Current flow when STEP = 2, DCOIL = 1, ITG = 1

Figure 12-14 shows the current flow in the SINx and COSx H-bridges when STEP = 3, DCOIL = 1 and ITG = 1.

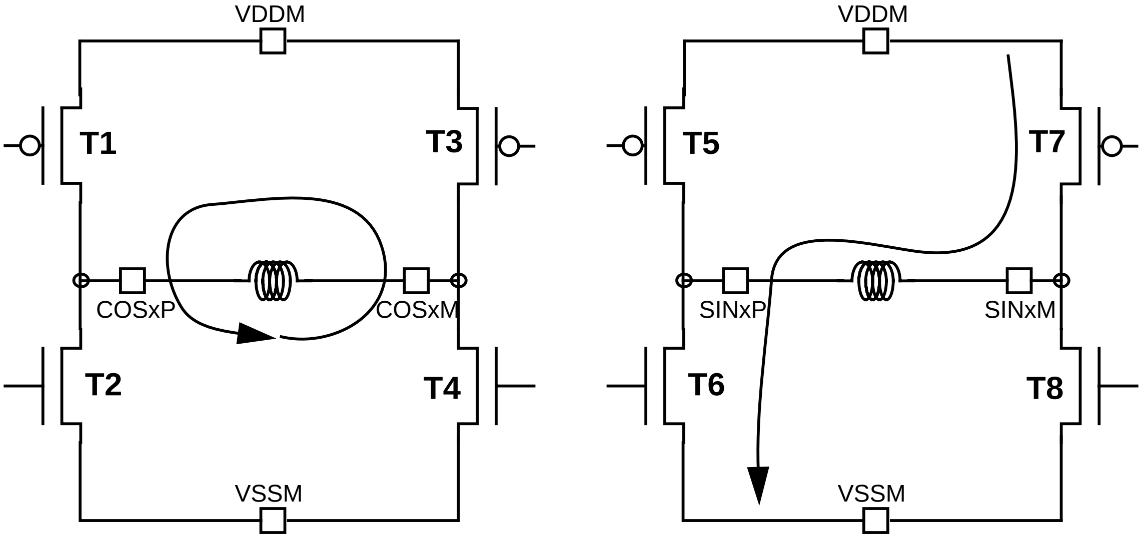


Figure 12-14. Current flow when STEP = 3, DCOIL = 1, ITG = 1

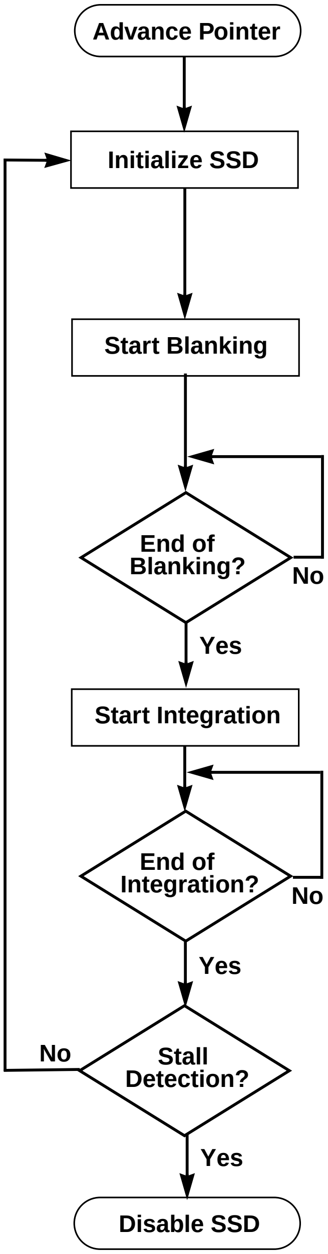
12.4.3 Operation in Low Power Modes

The SSD block can be configured for lower MCU power consumption in three different ways.

- Stop mode powers down the sigma-delta converter and halts clock to the modulus counter. Exit from Stop enables the sigma-delta converter and the clock to the modulus counter but due to the converter recovery time, the integration result should be ignored.
- Wait mode with SSDWAI bit set powers down the sigma-delta converter and halts the clock to the modulus counter. Exit from Wait enables the sigma-delta converter and clock to the modulus counter but due to the converter recovery time, the integration result should be ignored.
- Clearing SDCPU bit powers down the sigma-delta converter.

12.4.4 Stall Detection Flow

Figure 12-15 shows a flowchart and software setup for stall detection of a stepper motor. To control a second stepper motor, the SMS bit must be toggled during the SSD initialization.



Using Motor Control module, drive pointer to within 3 full steps of calibrated zero position.

1. Clear (or set) RCIR; clear (or set) POL; clear (or set) SMS;
2. Set MCZIE; clear MODMC; clear (or set) PRE; set MCEN.
3. Set RTZE; set SDCPU; write ACLKS (select sample frequency).
4. Store threshold value in RAM.

1. Clear MCZIF.
2. Write MDCCNT with blanking time value.
3. Clear ITG; clear (or set) DCOIL; increment (or decrement) STEP for CCW (or CW) motion.

MDCCNT = 0x0000? or MCZIF = 1?

1. Clear MCZIF.
2. Write MDCCNT with integration time value.
3. Set ITG; set DCOIL.

MDCCNT = 0x0000? or MCZIF = 1?

ITGACC < Threshold (RAM value)?

1. Clear MCZIF.
2. Clear MCEN.
3. Clear ITG.
4. Clear RTZE; clear SDCPU.

Figure 12-15. Return-to-Zero Flowchart

Chapter 13
Inter-Integrated Circuit (IICV3) Block Description

Table 13-1. Revision History

Revision Number	Revision Date	Sections Affected	Description of Changes
V01.03	28 Jul 2006	13.7.1.7/13-503	- Update flow-chart of interrupt routine for 10-bit address
V01.04	17 Nov 2006	13.3.1.2/13-483	- Revise Table1-5
Rev. 1.06	14 Aug 2007	13.3.1.1/13-483	- Backward compatible for IBAD bit name

13.1 Introduction

The inter-IC bus (IIC) is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange between devices. Being a two-wire device, the IIC bus minimizes the need for large numbers of connections between devices, and eliminates the need for an address decoder.

This bus is suitable for applications requiring occasional communications over a short distance between a number of devices. It also provides flexibility, allowing additional devices to be connected to the bus for further expansion and system development.

The interface is designed to operate up to 100 kbps with maximum bus loading and timing. The device is capable of operating at higher baud rates, up to a maximum of clock/20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF.

13.1.1 Features

The IIC module has the following key features:

- Compatible with I2C bus standard
- Multi-master operation
- Software programmable for one of 256 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation

- Acknowledge bit generation/detection
- Bus busy detection
- General Call Address detection
- Compliant to ten-bit address

13.1.2 Modes of Operation

The IIC functions the same in normal, special, and emulation modes. It has two low power modes: wait and stop modes.

13.1.3 Block Diagram

The block diagram of the IIC module is shown in Figure 13-1.

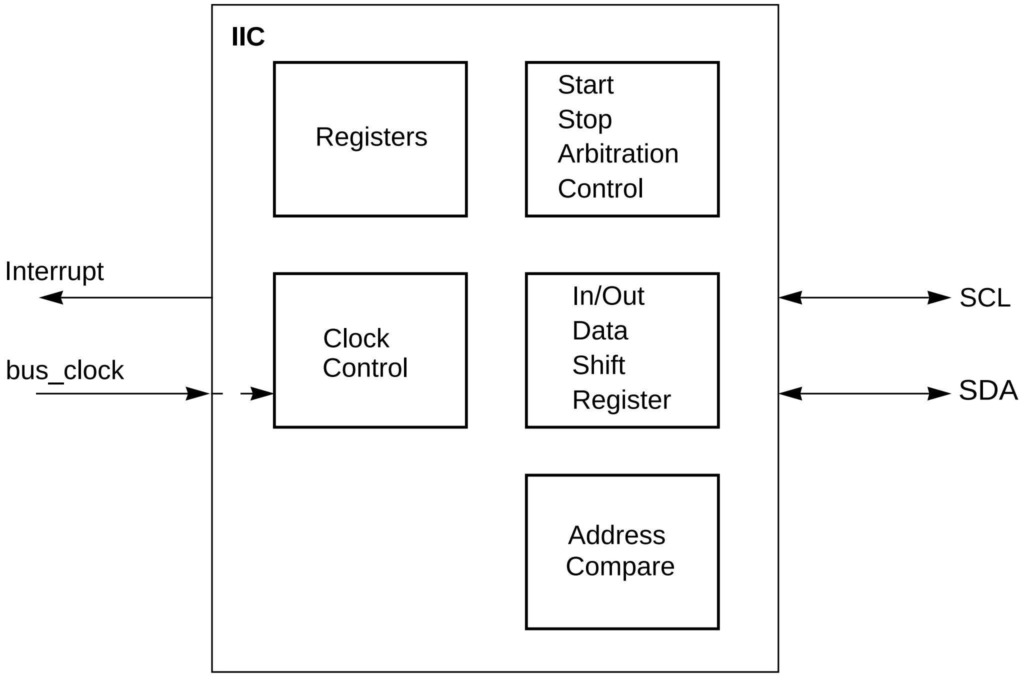


Figure 13-1. IIC Block Diagram

13.2 External Signal Description

The IICV3 module has two external pins.

13.2.1 IIC_SCL — Serial Clock Line Pin

This is the bidirectional serial clock line (SCL) of the module, compatible to the IIC bus specification.

13.2.2 IIC_SDA — Serial Data Line Pin

This is the bidirectional serial data line (SDA) of the module, compatible to the IIC bus specification.

13.3 Memory Map and Register Definition

This section provides a detailed description of all memory and registers for the IIC module.

13.3.1 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 IBAD	R	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	0
	W								▨
0x0001 IBFD	R	IBC7	IBC6	IBC5	IBC4	IBC3	IBC2	IBC1	IBC0
	W								
0x0002 IBCR	R	IBEN	IBIE	MS/$\overline{SL}$	Tx/$\overline{Rx}$	TXAK	0	0	IBSWAI
	W						RSTA	▨	
0x0003 IBSR	R	TCF	IAAS	IBB	IBAL	0	SRW	IBIF	RXAK
	W	▨	▨	▨		▨	▨		▨
0x0004 IBDR	R	D7	D6	D5	D4	D3	D2	D1	D0
	W								
0x0005 IBCR2	R	GCEN	ADTYPE	0	0	0	ADR10	ADR9	ADR8
	W			▨	▨	▨			

▨ = Unimplemented or Reserved

Figure 13-2. IIC Register Summary

13.3.1.1 IIC Address Register (IBAD)

Module Base +0x0000

	7	6	5	4	3	2	1	0
R	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	0
W								▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 13-3. IIC Bus Address Register (IBAD)

Read and write anytime

This register contains the address the IIC bus will respond to when addressed as a slave; note that it is not the address sent on the bus during the address transfer.

Table 13-2. IBAD Field Descriptions

Field	Description
7:1 ADR[7:1]	**Slave Address** — Bit 1 to bit 7 contain the specific slave address to be used by the IIC bus module.The default mode of IIC bus is slave mode for an address match on the bus.
0 Reserved	Reserved — Bit 0 of the IBAD is reserved for future compatibility. This bit will always read 0.

13.3.1.2 IIC Frequency Divider Register (IBFD)

Module Base + 0x0001

	7	6	5	4	3	2	1	0
R	IBC7	IBC6	IBC5	IBC4	IBC3	IBC2	IBC1	IBC0
W								
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 13-4. IIC Bus Frequency Divider Register (IBFD)

Read and write anytime

Table 13-3. IBFD Field Descriptions

Field	Description
7:0 IBC[7:0]	**I Bus Clock Rate 7:0** — This field is used to prescale the clock for bit rate selection. The bit clock generator is implemented as a prescale divider — IBC7:6, prescaled shift register — IBC5:3 select the prescaler divider and IBC2-0 select the shift register tap point. The IBC bits are decoded to give the tap and prescale values as shown in Table 13-4.

Table 13-4. I-Bus Tap and Prescale Values

IBC2-0 (bin)	SCL Tap (clocks)	SDA Tap (clocks)
000	5	1
001	6	1
010	7	2
011	8	2
100	9	3
101	10	3
110	12	4
111	15	4

Table 13-5. Prescale Divider Encoding

IBC5-3 (bin)	scl2start (clocks)	scl2stop (clocks)	scl2tap (clocks)	tap2tap (clocks)
000	2	7	4	1
001	2	7	4	2
010	2	9	6	4
011	6	9	6	8
100	14	17	14	16
101	30	33	30	32
110	62	65	62	64
111	126	129	126	128

Table 13-6. Multiplier Factor

IBC7-6	MUL
00	01
01	02
10	04
11	RESERVED

The number of clocks from the falling edge of SCL to the first tap (Tap[1]) is defined by the values shown in the scl2tap column of Table 13-4, all subsequent tap points are separated by 2^{IBC5-3} as shown in the tap2tap column in Table 13-5. The SCL Tap is used to generated the SCL period and the SDA Tap is used to determine the delay from the falling edge of SCL to SDA changing, the SDA hold time.

IBC7–6 defines the multiplier factor MUL. The values of MUL are shown in the Table 13-6.

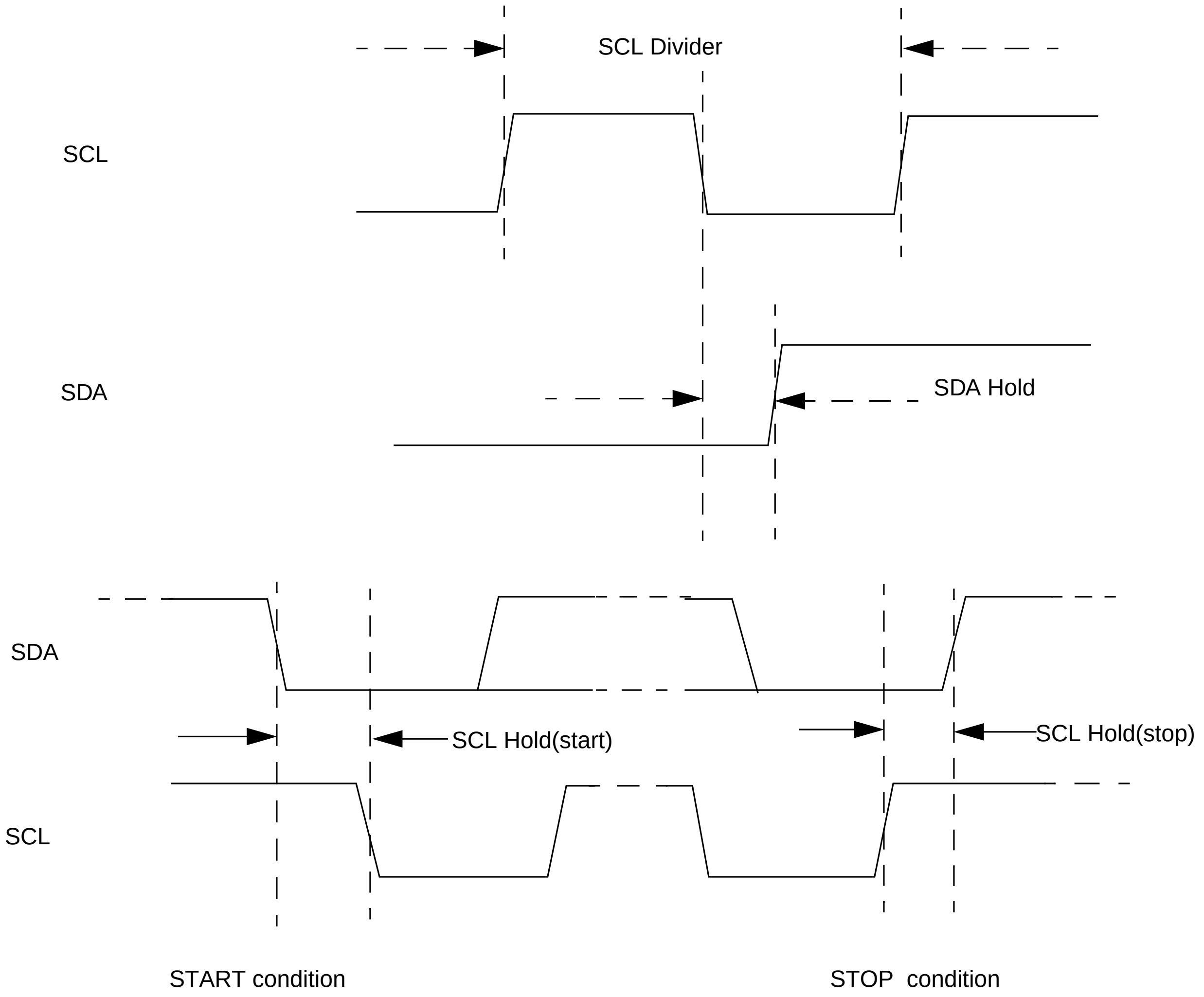


Figure 13-5. SCL Divider and SDA Hold

The equation used to generate the divider values from the IBFD bits is:

SCL Divider = MUL x {2 x (scl2tap + [(SCL_Tap -1) x tap2tap] + 2)}

The SDA hold delay is equal to the CPU clock period multiplied by the SDA Hold value shown in Table 13-7. The equation used to generate the SDA Hold value from the IBFD bits is:

SDA Hold = MUL x {scl2tap + [(SDA_Tap - 1) x tap2tap] + 3}

The equation for SCL Hold values to generate the start and stop conditions from the IBFD bits is:

SCL Hold(start) = MUL x [scl2start + (SCL_Tap - 1) x tap2tap]

SCL Hold(stop) = MUL x [scl2stop + (SCL_Tap - 1) x tap2tap]

Table 13-7. IIC Divider and Hold Values (Sheet 1 of 6)

IBC[7:0] (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL=1				

Table 13-7. IIC Divider and Hold Values (Sheet 2 of 6)

IBC[7:0] (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
00	20/22	7	6	11
01	22/24	7	7	12
02	24/26	8	8	13
03	26/28	8	9	14
04	28/30	9	10	15
05	30/32	9	11	16
06	34/36	10	13	18
07	40/42	10	16	21
08	28/32	7	10	15
09	32/36	7	12	17
0A	36/40	9	14	19
0B	40/44	9	16	21
0C	44/48	11	18	23
0D	48/52	11	20	25
0E	56/60	13	24	29
0F	68/72	13	30	35
10	48	9	18	25
11	56	9	22	29
12	64	13	26	33
13	72	13	30	37
14	80	17	34	41
15	88	17	38	45
16	104	21	46	53
17	128	21	58	65
18	80	9	38	41
19	96	9	46	49
1A	112	17	54	57
1B	128	17	62	65
1C	144	25	70	73
1D	160	25	78	81
1E	192	33	94	97
1F	240	33	118	121
20	160	17	78	81
21	192	17	94	97
22	224	33	110	113
23	256	33	126	129
24	288	49	142	145
25	320	49	158	161
26	384	65	190	193
27	480	65	238	241
28	320	33	158	161
29	384	33	190	193
2A	448	65	222	225
2B	512	65	254	257
2C	576	97	286	289

Table 13-7. IIC Divider and Hold Values (Sheet 3 of 6)

IBC[7:0] (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
2D	640	97	318	321
2E	768	129	382	385
2F	960	129	478	481
30	640	65	318	321
31	768	65	382	385
32	896	129	446	449
33	1024	129	510	513
34	1152	193	574	577
35	1280	193	638	641
36	1536	257	766	769
37	1920	257	958	961
38	1280	129	638	641
39	1536	129	766	769
3A	1792	257	894	897
3B	2048	257	1022	1025
3C	2304	385	1150	1153
3D	2560	385	1278	1281
3E	3072	513	1534	1537
3F	3840	513	1918	1921
MUL=2				
40	40	14	12	22
41	44	14	14	24
42	48	16	16	26
43	52	16	18	28
44	56	18	20	30
45	60	18	22	32
46	68	20	26	36
47	80	20	32	42
48	56	14	20	30
49	64	14	24	34
4A	72	18	28	38
4B	80	18	32	42
4C	88	22	36	46
4D	96	22	40	50
4E	112	26	48	58
4F	136	26	60	70
50	96	18	36	50
51	112	18	44	58
52	128	26	52	66
53	144	26	60	74
54	160	34	68	82
55	176	34	76	90
56	208	42	92	106
57	256	42	116	130
58	160	18	76	82

Table 13-7. IIC Divider and Hold Values (Sheet 4 of 6)

IBC[7:0] (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
59	192	18	92	98
5A	224	34	108	114
5B	256	34	124	130
5C	288	50	140	146
5D	320	50	156	162
5E	384	66	188	194
5F	480	66	236	242
60	320	34	156	162
61	384	34	188	194
62	448	66	220	226
63	512	66	252	258
64	576	98	284	290
65	640	98	316	322
66	768	130	380	386
67	960	130	476	482
68	640	66	316	322
69	768	66	380	386
6A	896	130	444	450
6B	1024	130	508	514
6C	1152	194	572	578
6D	1280	194	636	642
6E	1536	258	764	770
6F	1920	258	956	962
70	1280	130	636	642
71	1536	130	764	770
72	1792	258	892	898
73	2048	258	1020	1026
74	2304	386	1148	1154
75	2560	386	1276	1282
76	3072	514	1532	1538
77	3840	514	1916	1922
78	2560	258	1276	1282
79	3072	258	1532	1538
7A	3584	514	1788	1794
7B	4096	514	2044	2050
7C	4608	770	2300	2306
7D	5120	770	2556	2562
7E	6144	1026	3068	3074
7F	7680	1026	3836	3842
MUL=4				
80	72	28	24	44
81	80	28	28	48
82	88	32	32	52
83	96	32	36	56
84	104	36	40	60

Table 13-7. IIC Divider and Hold Values (Sheet 5 of 6)

IBC[7:0] (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
85	112	36	44	64
86	128	40	52	72
87	152	40	64	84
88	112	28	40	60
89	128	28	48	68
8A	144	36	56	76
8B	160	36	64	84
8C	176	44	72	92
8D	192	44	80	100
8E	224	52	96	116
8F	272	52	120	140
90	192	36	72	100
91	224	36	88	116
92	256	52	104	132
93	288	52	120	148
94	320	68	136	164
95	352	68	152	180
96	416	84	184	212
97	512	84	232	260
98	320	36	152	164
99	384	36	184	196
9A	448	68	216	228
9B	512	68	248	260
9C	576	100	280	292
9D	640	100	312	324
9E	768	132	376	388
9F	960	132	472	484
A0	640	68	312	324
A1	768	68	376	388
A2	896	132	440	452
A3	1024	132	504	516
A4	1152	196	568	580
A5	1280	196	632	644
A6	1536	260	760	772
A7	1920	260	952	964
A8	1280	132	632	644
A9	1536	132	760	772
AA	1792	260	888	900
AB	2048	260	1016	1028
AC	2304	388	1144	1156
AD	2560	388	1272	1284
AE	3072	516	1528	1540
AF	3840	516	1912	1924
B0	2560	260	1272	1284
B1	3072	260	1528	1540

Table 13-7. IIC Divider and Hold Values (Sheet 6 of 6)

IBC[7:0] (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
B2	3584	516	1784	1796
B3	4096	516	2040	2052
B4	4608	772	2296	2308
B5	5120	772	2552	2564
B6	6144	1028	3064	3076
B7	7680	1028	3832	3844
B8	5120	516	2552	2564
B9	6144	516	3064	3076
BA	7168	1028	3576	3588
BB	8192	1028	4088	4100
BC	9216	1540	4600	4612
BD	10240	1540	5112	5124
BE	12288	2052	6136	6148
BF	15360	2052	7672	7684

Note:Since the bus frequency is speeding up,the SCL Divider could be expanded by it.Therefore,in the table,when IBC[7:0] is from $00 to $0F,the SCL Divider is revised by the format value1/value2.Value1 is the divider under the low frequency.Value2 is the divider under the high frequency.How to select the divider depends on the bus frequency.When IBC[7:0] is from $10 to $BF,the divider is not changed.

13.3.1.3 IIC Control Register (IBCR)

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R	IBEN	IBIE	MS/SL	Tx/Rx	TXAK	0	0	IBSWAI
W						RSTA	▨	
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 13-6. IIC Bus Control Register (IBCR)

Read and write anytime

Table 13-8. IBCR Field Descriptions

Field	Description
7 IBEN	**I-Bus Enable** — This bit controls the software reset of the entire IIC bus module. 0 The module is reset and disabled. This is the power-on reset situation. When low the interface is held in reset but registers can be accessed 1 The IIC bus module is enabled.This bit must be set before any other IBCR bits have any effect If the IIC bus module is enabled in the middle of a byte transfer the interface behaves as follows: slave mode ignores the current transfer on the bus and starts operating whenever a subsequent start condition is detected. Master mode will not be aware that the bus is busy, hence if a start cycle is initiated then the current bus cycle may become corrupt. This would ultimately result in either the current bus master or the IIC bus module losing arbitration, after which bus operation would return to normal.
6 IBIE	**I-Bus Interrupt Enable** 0 Interrupts from the IIC bus module are disabled. Note that this does not clear any currently pending interrupt condition 1 Interrupts from the IIC bus module are enabled. An IIC bus interrupt occurs provided the IBIF bit in the status register is also set.
5 MS/SL	**Master/Slave Mode Select Bit** — Upon reset, this bit is cleared. When this bit is changed from 0 to 1, a START signal is generated on the bus, and the master mode is selected. When this bit is changed from 1 to 0, a STOP signal is generated and the operation mode changes from master to slave.A STOP signal should only be generated if the IBIF flag is set. MS/$\overline{\text{SL}}$ is cleared without generating a STOP signal when the master loses arbitration. 0 Slave Mode 1 Master Mode
4 Tx/Rx	**Transmit/Receive Mode Select Bit** — This bit selects the direction of master and slave transfers. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. In master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. 0 Receive 1 Transmit
3 TXAK	**Transmit Acknowledge Enable** — This bit specifies the value driven onto SDA during data acknowledge cycles for both master and slave receivers. The IIC module will always acknowledge address matches, provided it is enabled, regardless of the value of TXAK. Note that values written to this bit are only used when the IIC bus is a receiver, not a transmitter. 0 An acknowledge signal will be sent out to the bus at the 9th clock bit after receiving one byte data 1 No acknowledge signal response is sent (i.e., acknowledge bit = 1)
2 RSTA	**Repeat Start** — Writing a 1 to this bit will generate a repeated START condition on the bus, provided it is the current bus master. This bit will always be read as a low. Attempting a repeated start at the wrong time, if the bus is owned by another master, will result in loss of arbitration. 1 Generate repeat start cycle
1 RESERVED	**Reserved** — Bit 1 of the IBCR is reserved for future compatibility. This bit will always read 0.
0 IBSWAI	**I Bus Interface Stop in Wait Mode** 0 IIC bus module clock operates normally 1 Halt IIC bus module clock generation in wait mode

Wait mode is entered via execution of a CPU WAI instruction. In the event that the IBSWAI bit is set, all clocks internal to the IIC will be stopped and any transmission currently in progress will halt.If the CPU were woken up by a source other than the IIC module, then clocks would restart and the IIC would resume from where was during the previous transmission. It is not possible for the IIC to wake up the CPU when its internal clocks are stopped.

If it were the case that the IBSWAI bit was cleared when the WAI instruction was executed, the IIC internal clocks and interface would remain alive, continuing the operation which was currently underway. It is also possible to configure the IIC such that it will wake up the CPU via an interrupt at the conclusion of the current operation. See the discussion on the IBIF and IBIE bits in the IBSR and IBCR, respectively.

13.3.1.4 IIC Status Register (IBSR)

Module Base + 0x0003

	7	6	5	4	3	2	1	0
R	TCF	IAAS	IBB	IBAL	0	SRW	IBIF	RXAK
W	▨	▨	▨		▨	▨		▨
Reset	1	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 13-7. IIC Bus Status Register (IBSR)

This status register is read-only with exception of bit 1 (IBIF) and bit 4 (IBAL), which are software clearable.

Table 13-9. IBSR Field Descriptions

Field	Description
7 TCF	**Data Transferring Bit** — While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. Note that this bit is only valid during or immediately following a transfer to the IIC module or from the IIC module. 0 Transfer in progress 1 Transfer complete
6 IAAS	**Addressed as a Slave Bit** — When its own specific address (I-bus address register) is matched with the calling address or it receives the general call address with GCEN== 1,this bit is set.The CPU is interrupted provided the IBIE is set.Then the CPU needs to check the SRW bit and set its Tx/$\overline{\text{Rx}}$ mode accordingly.Writing to the I-bus control register clears this bit. 0 Not addressed 1 Addressed as a slave
5 IBB	Bus Busy Bit 0 This bit indicates the status of the bus. When a START signal is detected, the IBB is set. If a STOP signal is detected, IBB is cleared and the bus enters idle state. 1 Bus is busy
4 IBAL	**Arbitration Lost** — The arbitration lost bit (IBAL) is set by hardware when the arbitration procedure is lost. Arbitration is lost in the following circumstances: 1. SDA sampled low when the master drives a high during an address or data transmit cycle. 2. SDA sampled low when the master drives a high during the acknowledge bit of a data receive cycle. 3. A start cycle is attempted when the bus is busy. 4. A repeated start cycle is requested in slave mode. 5. A stop condition is detected when the master did not request it. This bit must be cleared by software, by writing a one to it. A write of 0 has no effect on this bit.

Table 13-9. IBSR Field Descriptions (continued)

Field	Description
3 RESERVED	**Reserved** — Bit 3 of IBSR is reserved for future use. A read operation on this bit will return 0.
2 SRW	**Slave Read/Write** — When IAAS is set this bit indicates the value of the R/W command bit of the calling address sent from the master This bit is only valid when the I-bus is in slave mode, a complete address transfer has occurred with an address match and no other transfers have been initiated. Checking this bit, the CPU can select slave transmit/receive mode according to the command of the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave
1 IBIF	**I-Bus Interrupt** — The IBIF bit is set when one of the following conditions occurs: — Arbitration lost (IBAL bit set) — Data transfer complete (TCF bit set) — Addressed as slave (IAAS bit set) It will cause a processor interrupt request if the IBIE bit is set. This bit must be cleared by software, writing a one to it. A write of 0 has no effect on this bit.
0 RXAK	**Received Acknowledge** — The value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion of 8 bits data transmission on the bus. If RXAK is high, it means no acknowledge signal is detected at the 9th clock. 0 Acknowledge received 1 No acknowledge received

13.3.1.5 IIC Data I/O Register (IBDR)

Module Base + 0x0004

	7	6	5	4	3	2	1	0
R W	D7	D6	D5	D4	D3	D2	D1	D0
Reset	0	0	0	0	0	0	0	0

Figure 13-8. IIC Bus Data I/O Register (IBDR)

In master transmit mode, when data is written to the IBDR a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates next byte data receiving. In slave mode, the same functions are available after an address match has occurred.Note that the Tx/Rx bit in the IBCR must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the IIC is configured for master transmit but a master receive is desired, then reading the IBDR will not initiate the receive.

Reading the IBDR will return the last byte received while the IIC is configured in either master receive or slave receive modes. The IBDR does not reflect every byte that is transmitted on the IIC bus, nor can software verify that a byte has been written to the IBDR correctly by reading it back.

In master transmit mode, the first byte of data written to IBDR following assertion of MS/$\overline{\text{SL}}$ is used for the address transfer and should com.prise of the calling address (in position D7:D1) concatenated with the required R/$\overline{\text{W}}$ bit (in position D0).

13.3.1.6 IIC Control Register 2(IBCR2)

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R	GCEN	ADTYPE	0	0	0	ADR10	ADR9	ADR8
W								
Reset	0	0	0	0	0	0	0	0

Figure 13-9. IIC Bus Control Register 2(IBCR2)

This register contains the variables used in general call and in ten-bit address.

Read and write anytime

Table 13-10. IBCR2 Field Descriptions

Field	Description
7 GCEN	**General Call Enable**. 0 General call is disabled. The module dont receive any general call data and address. 1 enable general call. It indicates that the module can receive address and any data.
6 ADTYPE	**Address Type—** This bit selects the address length. The variable must be configured correctly before IIC enters slave mode. 0 7-bit address 1 10-bit address
5,4,3 RESERVED	**Reserved** — Bit 5,4 and 3 of the IBCR2 are reserved for future compatibility. These bits will always read 0.
2:0 ADR[10:8]	**Slave Address [10:8]** —These 3 bits represent the MSB of the 10-bit address when address type is asserted (ADTYPE = 1).

13.4 Functional Description

This section provides a complete functional description of the IICV3.

13.4.1 I-Bus Protocol

The IIC bus system uses a serial data line (SDA) and a serial clock line (SCL) for data transfer. All devices connected to it must have open drain or open collector outputs. Logic AND function is exercised on both lines with external pull-up resistors. The value of these resistors is system dependent.

Normally, a standard communication is composed of four parts: START signal, slave address transmission, data transfer and STOP signal. They are described briefly in the following sections and illustrated in Figure 13-10.

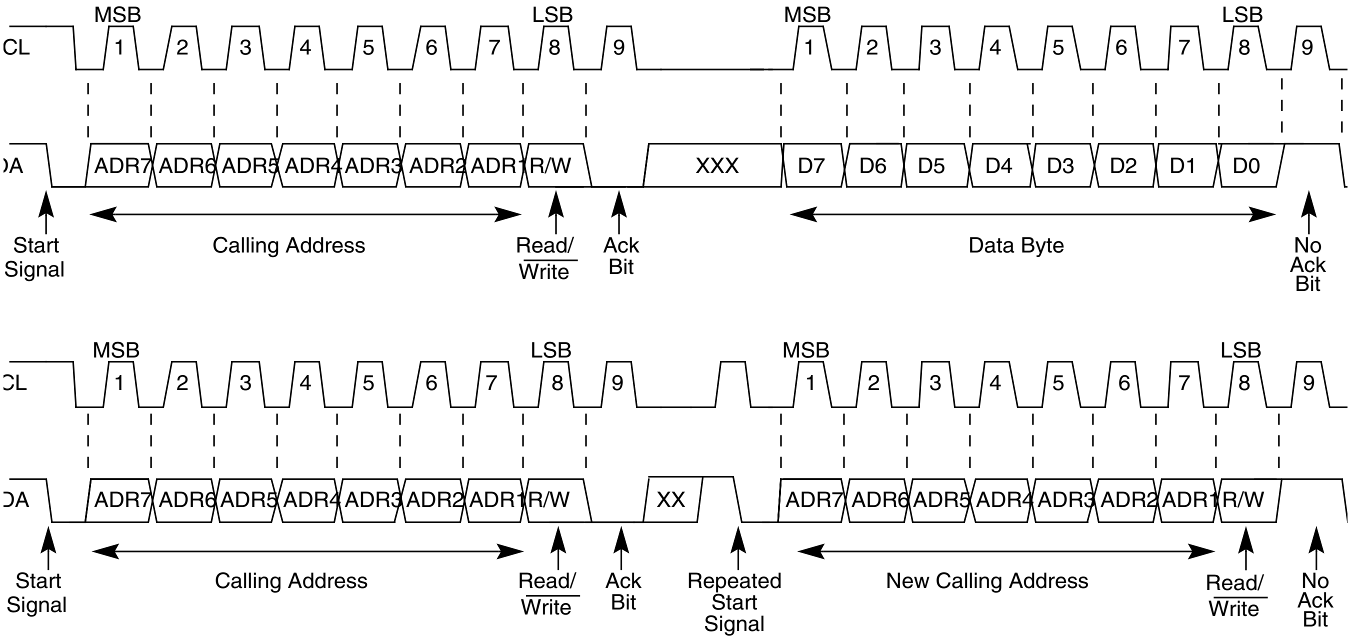


Figure 13-10. IIC-Bus Transmission Signals

13.4.1.1 START Signal

When the bus is free, i.e. no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START signal.As shown in Figure 13-10, a START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

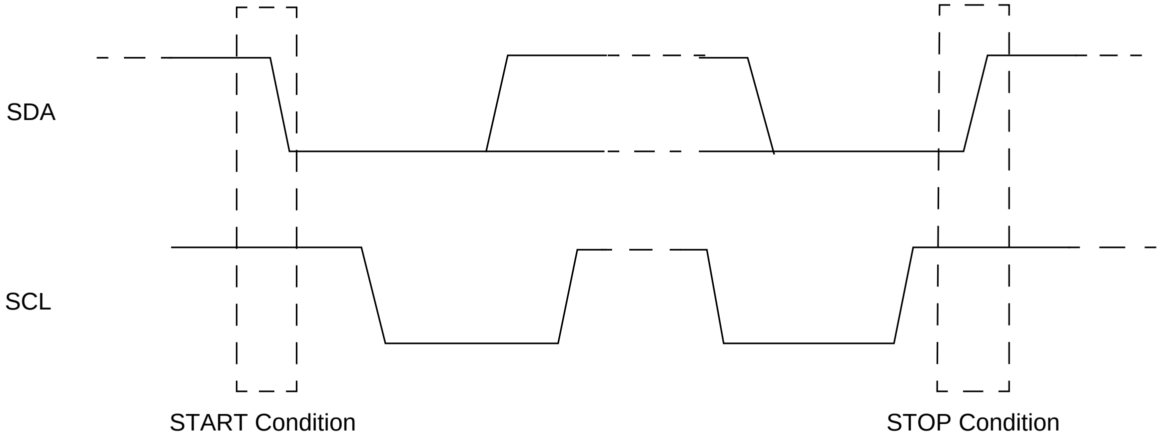


Figure 13-11. Start and Stop Conditions

13.4.1.2 Slave Address Transmission

The first byte of data transfer immediately after the START signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/W bit. The R/W bit tells the slave the desired direction of data transfer.

1 = Read transfer, the slave transmits data to the master.
0 = Write transfer, the master transmits data to the slave.

If the calling address is 10-bit, another byte is followed by the first byte.Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see Figure 13-10).

No two slaves in the system may have the same address. If the IIC bus is master, it must not transmit an address that is equal to its own slave address. The IIC bus cannot be master and slave at the same time.However, if arbitration is lost during an address cycle the IIC bus will revert to slave mode and operate correctly even if it is being addressed by another master.

13.4.1.3 Data Transfer

As soon as successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/W bit sent by the calling master

All transfers that come after an address cycle are referred to as data transfers, even if they carry sub-address information for the slave device.

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in Figure 13-10. There is one clock pulse on SCL for each data bit, the MSB being transferred first. Each data byte has to be followed by an acknowledge bit, which is signalled from the receiving device by pulling the SDA low at the ninth clock. So one complete data byte transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop signal to abort the data transfer or a start signal (repeated start) to commence a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it means 'end of data' to the slave, so the slave releases the SDA line for the master to generate STOP or START signal.Note in order to release the bus correctly,after no-acknowledge to the master,the slave must be immediately switched to receiver and a following dummy reading of the IBDR is necessary.

13.4.1.4 STOP Signal

The master can terminate the communication by generating a STOP signal to free the bus. However, the master may generate a START signal followed by a calling command without generating a STOP signal first. This is called repeated START. A STOP signal is defined as a low-to-high transition of SDA while SCL at logical 1 (see Figure 13-10).

The master can generate a STOP even if the slave has generated an acknowledge at which point the slave must release the bus.

13.4.1.5 Repeated START Signal

As shown in Figure 13-10, a repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication. This is used by the master to communicate with another slave or with the same slave in different mode (transmit/receive mode) without releasing the bus.

13.4.1.6 Arbitration Procedure

The Inter-IC bus is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure, a bus master loses arbitration if it transmits logic 1 while another master transmits logic 0. The losing masters immediately switch over to slave receive mode and stop driving SDA output. In this case the transition from master to slave mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

13.4.1.7 Clock Synchronization

Because wire-AND logic is performed on SCL line, a high-to-low transition on SCL line affects all the devices connected on the bus. The devices start counting their low period and as soon as a device's clock has gone low, it holds the SCL line low until the clock high state is reached.However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see Figure 13-11). When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. There is then no difference between the device clocks and the state of the SCL line and all the devices start counting their high periods.The first device to complete its high period pulls the SCL line low again.

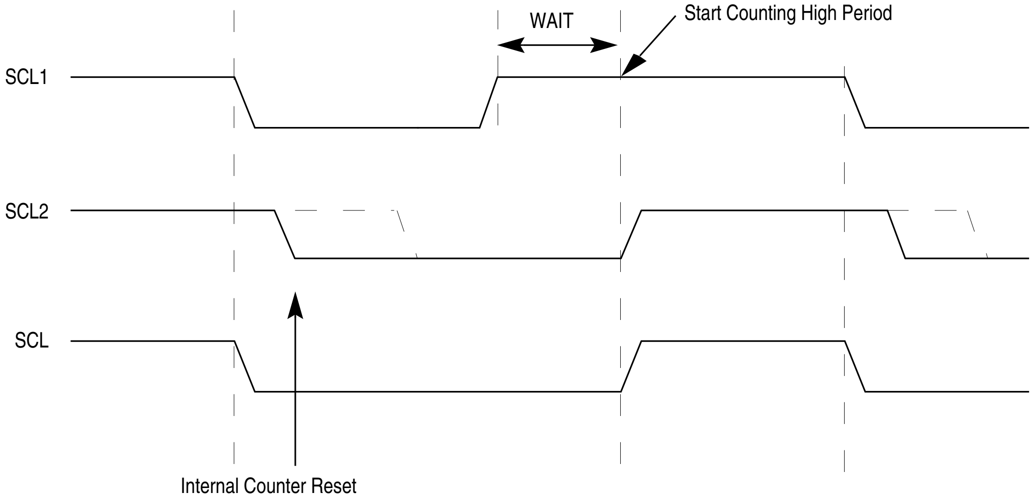


Figure 13-12. IIC-Bus Clock Synchronization

13.4.1.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such case, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

13.4.1.9 Clock Stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low the slave can drive SCL low for the required period and then release it.If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

13.4.1.10 Ten-bit Address

A ten-bit address is indicated if the first 5 bits of the first address byte are 0x11110. The following rules apply to the first address byte.

SLAVE ADDRESS	R/W BIT	DESCRIPTION
0000000	0	General call address
0000010	x	Reserved for different bus format
0000011	x	Reserved for future purposes
11111XX	x	Reserved for future purposes
11110XX	x	10-bit slave addressing

Figure 13-13. Definition of bits in the first byte.

The address type is identified by ADTYPE. When ADTYPE is 0, 7-bit address is applied. Reversely, the address is 10-bit address.Generally, there are two cases of 10-bit address.See the Figure 13-14 and Figure 13-15.

S	Slave Add1st 7bits 11110+ADR10+ADR9	R/W 0	A1	Slave Add 2nd byte ADR[8:1]	A2	Data	A3

Figure 13-14. A master-transmitter addresses a slave-receiver with a 10-bit address

S	Slave Add1st 7bits 11110+ADR10+ADR9	R/W 0	A1	Slave Add 2nd byte ADR[8:1]	A2	Sr	Slave Add 1st 7bits 11110+ADR10+ADR9	R/W 1	A3	Data	A4

Figure 13-15. A master-receiver addresses a slave-transmitter with a 10-bit address.

In the Figure 13-15, the first two bytes are the similar to Figure 13-14. After the repeated START(Sr),the first slave address is transmitted again, but the R/W is 1, meaning that the slave is acted as a transmitter.

13.4.1.11 General Call Address

To broadcast using a general call, a device must first generate the general call address($00), then after receiving acknowledge, it must transmit data.

In communication, as a slave device, provided the GCEN is asserted, a device acknowledges the broadcast and receives data until the GCEN is disabled or the master device releases the bus or generates a new transfer. In the broadcast, slaves always act as receivers. In general call, IAAS is also used to indicate the address match.

In order to distinguish whether the address match is the normal address match or the general call address match, IBDR should be read after the address byte has been received. If the data is $00, the match is general call address match. The meaning of the general call address is always specified in the first data byte and must be dealt with by S/W, the IIC hardware does not decode and process the first data byte.

When one byte transfer is done, the received data can be read from IBDR. The user can control the procedure by enabling or disabling GCEN.

13.4.2 Operation in Run Mode

This is the basic mode of operation.

13.4.3 Operation in Wait Mode

IIC operation in wait mode can be configured. Depending on the state of internal bits, the IIC can operate normally when the CPU is in wait mode or the IIC clock generation can be turned off and the IIC module enters a power conservation state during wait mode. In the later case, any transmission or reception in progress stops at wait mode entry.

13.4.4 Operation in Stop Mode

The IIC is inactive in stop mode for reduced power consumption. The STOP instruction does not affect IIC register states.

13.5 Resets

The reset state of each individual bit is listed in Section 13.3, “Memory Map and Register Definition,” which details the registers and their bit-fields.

13.6 Interrupts

IICV3 uses only one interrupt vector.

Table 13-11. Interrupt Summary

Interrupt	Offset	Vector	Priority	Source	Description

IIC Interrupt	—	—	—	IBAL, TCF, IAAS bits in IBSR register	When either of IBAL, TCF or IAAS bits is set may cause an interrupt based on arbitration lost, transfer complete or address detect conditions

Internally there are three types of interrupts in IIC. The interrupt service routine can determine the interrupt type by reading the status register.

IIC Interrupt can be generated on

1. Arbitration lost condition (IBAL bit set)
2. Byte transfer condition (TCF bit set)
3. Address detect condition (IAAS bit set)

The IIC interrupt is enabled by the IBIE bit in the IIC control register. It must be cleared by writing 0 to the IBF bit in the interrupt service routine.

13.7 Application Information

13.7.1 IIC Programming Examples

13.7.1.1 Initialization Sequence

Reset will put the IIC bus control register to its default status. Before the interface can be used to transfer serial data, an initialization procedure must be carried out, as follows:

1. Update the frequency divider register (IBFD) and select the required division ratio to obtain SCL frequency from system clock.
2. Update the ADTYPE of IBCR2 to define the address length, 7 bits or 10 bits.
3. Update the IIC bus address register (IBAD) to define its slave address. If 10-bit address is applied IBCR2 should be updated to define the rest bits of address.
4. Set the IBEN bit of the IIC bus control register (IBCR) to enable the IIC interface system.
5. Modify the bits of the IIC bus control register (IBCR) to select master/slave mode, transmit/receive mode and interrupt enable or not.
6. If supported general call, the GCEN in IBCR2 should be asserted.

13.7.1.2 Generation of START

After completion of the initialization procedure, serial data can be transmitted by selecting the 'master transmitter' mode. If the device is connected to a multi-master bus system, the state of the IIC bus busy bit (IBB) must be tested to check whether the serial bus is free.

If the bus is free (IBB=0), the start condition and the first byte (the slave address) can be sent. The data written to the data register comprises the slave calling address and the LSB set to indicate the direction of transfer required from the slave.

The bus free time (i.e., the time between a STOP condition and the following START condition) is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period it may be necessary to wait until the IIC is busy after writing the calling address to the IBDR before proceeding with the following instructions. This is illustrated in the following example.

An example of a program which generates the START signal and transmits the first byte of data (slave address) is shown below:

```
CHFLAG      BRSET   IBSR,#$20,*      ;WAIT FOR IBB FLAG TO CLEAR
TXSTART     BSET    IBCR,#$30        ;SET TRANSMIT AND MASTER MODE;i.e. GENERATE START CONDITION
            MOVB    CALLING,IBDR     ;TRANSMIT THE CALLING ADDRESS, D0=R/W
IBFREE      BRCLR   IBSR,#$20,*      ;WAIT FOR IBB FLAG TO SET
```

13.7.1.3 Post-Transfer Software Response

Transmission or reception of a byte will set the data transferring bit (TCF) to 1, which indicates one byte communication is finished. The IIC bus interrupt bit (IBIF) is set also; an interrupt will be generated if the interrupt function is enabled during initialization by setting the IBIE bit. Software must clear the IBIF bit in the interrupt routine first. The TCF bit will be cleared by reading from the IIC bus data I/O register (IBDR) in receive mode or writing to IBDR in transmit mode.

Software may service the IIC I/O in the main program by monitoring the IBIF bit if the interrupt function is disabled. Note that polling should monitor the IBIF bit rather than the TCF bit because their operation is different when arbitration is lost.

Note that when an interrupt occurs at the end of the address cycle the master will always be in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/W bit in IBDR, then the Tx/Rx bit should be toggled at this stage.

During slave mode address cycles (IAAS=1), the SRW bit in the status register is read to determine the direction of the subsequent transfer and the Tx/Rx bit is programmed accordingly.For slave mode data cycles (IAAS=0) the SRW bit is not valid, the Tx/Rx bit in the control register should be read to determine the direction of the current transfer.

The following is an example of a software response by a 'master transmitter' in the interrupt routine.

```
ISR         BCLR     IBSR,#$02              ;CLEAR THE IBIF FLAG
            BRCLR    IBCR,#$20,SLAVE        ;BRANCH IF IN SLAVE MODE
            BRCLR    IBCR,#$10,RECEIVE      ;BRANCH IF IN RECEIVE MODE
            BRSET    IBSR,#$01,END          ;IF NO ACK, END OF TRANSMISSION
TRANSMIT    MOVB     DATABUF,IBDR           ;TRANSMIT NEXT BYTE OF DATA
```

13.7.1.4 Generation of STOP

A data transfer ends with a STOP signal generated by the 'master' device. A master transmitter can simply generate a STOP signal after all the data has been transmitted. The following is an example showing how a stop condition is generated by a master transmitter.

```
MASTX     TST      TXCNT            ;GET VALUE FROM THE TRANSMITING COUNTER
          BEQ      END              ;END IF NO MORE DATA
          BRSET    IBSR,#$01,END    ;END IF NO ACK
          MOVB     DATABUF,IBDR     ;TRANSMIT NEXT BYTE OF DATA
          DEC      TXCNT            ;DECREASE THE TXCNT
          BRA      EMASTX           ;EXIT
END       BCLR     IBCR,#$20        ;GENERATE A STOP CONDITION
EMASTX    RTI                       ;RETURN FROM INTERRUPT
```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the transmit acknowledge bit (TXAK) before reading the 2nd last byte of data. Before reading the last byte of data, a STOP signal must be generated first. The following is an example showing how a STOP signal is generated by a master receiver.

```
MASR      DEC      RXCNT            ;DECREASE THE RXCNT
          BEQ      ENMASR           ;LAST BYTE TO BE READ
          MOVB     RXCNT,D1         ;CHECK SECOND LAST BYTE
          DEC      D1               ;TO BE READ
          BNE      NXMAR            ;NOT LAST OR SECOND LAST
LAMAR     BSET     IBCR,#$08        ;SECOND LAST, DISABLE ACK
                                    ;TRANSMITTING
          BRA      NXMAR
ENMASR    BCLR     IBCR,#$20        ;LAST ONE, GENERATE ‘STOP’ SIGNAL
NXMAR     MOVB     IBDR,RXBUF       ;READ DATA AND STORE
          RTI
```

13.7.1.5 Generation of Repeated START

At the end of data transfer, if the master continues to want to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown.

```
RESTART   BSET     IBCR,#$04        ;ANOTHER START (RESTART)
          MOVB     CALLING,IBDR     ;TRANSMIT THE CALLING ADDRESS;D0=R/W
```

13.7.1.6 Slave Mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (Tx/Rx bit of IBCR) according to the R/W command bit (SRW). Writing to the IBCR clears the IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred, interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR, for slave transmits, or dummy reading from IBDR, in slave receive mode. The slave will drive SCL low in-between byte transfers, SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means an 'end of data' signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a STOP signal.

13.7.1.7 Arbitration Lost

If several masters try to engage the bus simultaneously, only one master wins and the others lose arbitration. The devices which lost arbitration are immediately switched to slave receive mode by the hardware. Their data output to the SDA line is stopped, but SCL continues to be generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with IBAL=1 and MS/SL=0. If one master attempts to start transmission while the bus is being engaged by another master, the hardware will inhibit the transmission; switch the MS/SL bit from 1 to 0 without generating STOP condition; generate an interrupt to CPU and set the IBAL to indicate that the attempt to engage the bus is failed. When considering these cases, the slave service routine should test the IBAL first and the software should clear the IBAL bit if it is set.

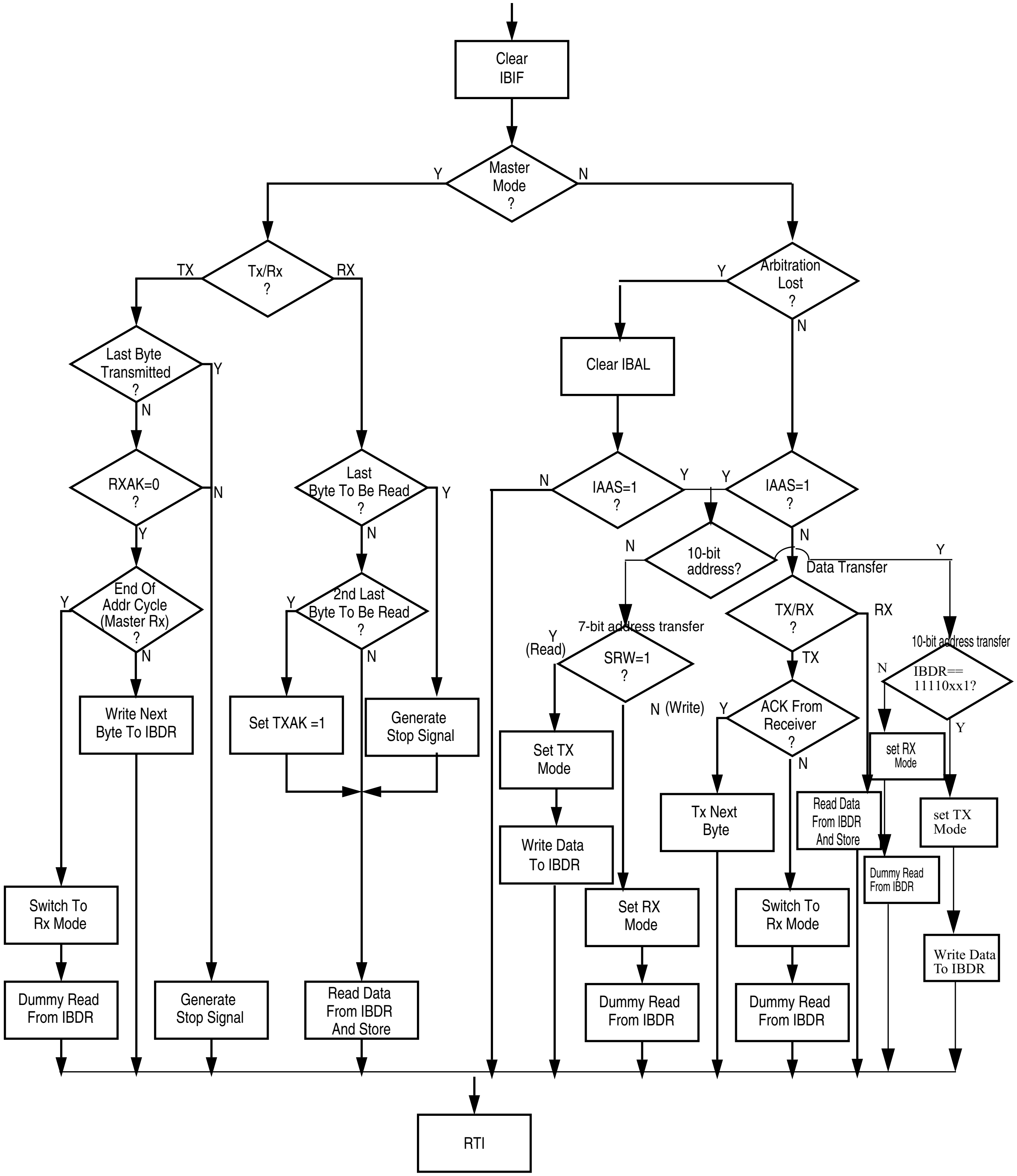


Figure 13-16. Flow-Chart of Typical IIC Interrupt Routine

Caution:When IIC is configured as 10-bit address,the point of the data array in interrupt routine must be reset after it’s addressed.

Chapter 14
Freescale's Scalable Controller Area Network (S12MSCANV3)

Table 14-1. Revision History

Revision Number	Revision Date	Sections Affected	Description of Changes
V03.10	19 Aug 2008	14.4.7.4/14-559 14.4.4.5/14-553 14.2/14-510	- Corrected wake-up description - Relocated initialization section - Added note to external pin descriptions for use with integrated physical layer - Minor corrections
V03.11	31 Mar 2009		- Orthographic corrections
V03.12	09 Aug 2010	Table 14-37	- Added 'Bosch CAN 2.0A/B' to bit time settings table

14.1 Introduction

Freescale's scalable controller area network (S12MSCANV3) definition is based on the MSCAN12 definition, which is the specific implementation of the MSCAN concept targeted for the M68HC12 microcontroller family.

The module is a communication controller implementing the CAN 2.0A/B protocol as defined in the Bosch specification dated September 1991. For users to fully understand the MSCAN specification, it is recommended that the Bosch specification be read first to familiarize the reader with the terms and concepts contained within this document.

Though not exclusively intended for automotive applications, CAN protocol is designed to meet the specific requirements of a vehicle serial data bus: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness, and required bandwidth.

MSCAN uses an advanced buffer arrangement resulting in predictable real-time behavior and simplified application software.

14.1.1 Glossary

Table 14-2. Terminology

ACK	Acknowledge of CAN message
CAN	Controller Area Network
CRC	Cyclic Redundancy Code
EOF	End of Frame
FIFO	First-In-First-Out Memory
IFS	Inter-Frame Sequence
SOF	Start of Frame
CPU bus	CPU related read/write data bus
CAN bus	CAN protocol related serial bus
oscillator clock	Direct clock from external oscillator
bus clock	CPU bus related clock
CAN clock	CAN protocol related clock

14.1.2 Block Diagram

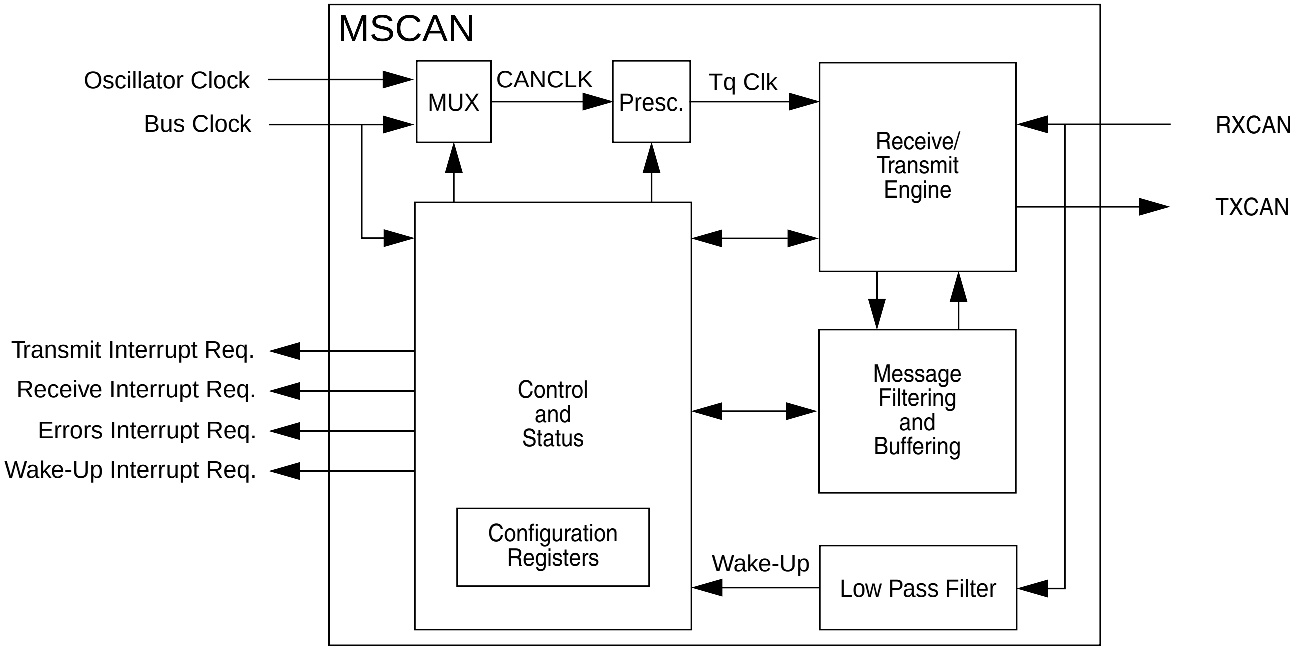


Figure 14-1. MSCAN Block Diagram

14.1.3 Features

The basic features of the MSCAN are as follows:

- Implementation of the CAN protocol — Version 2.0A/B
    - Standard and extended data frames
    - Zero to eight bytes data length
    - Programmable bit rate up to 1 Mbps[1]
    - Support for remote frames
- Five receive buffers with FIFO storage scheme
- Three transmit buffers with internal prioritization using a “local priority” concept
- Flexible maskable identifier filter supports two full-size (32-bit) extended identifier filters, or four 16-bit filters, or eight 8-bit filters
- Programmable wake-up functionality with integrated low-pass filter
- Programmable loopback mode supports self-test operation
- Programmable listen-only mode for monitoring of CAN bus
- Programmable bus-off recovery functionality
- Separate signalling and interrupt capabilities for all CAN receiver and transmitter error states (warning, error passive, bus-off)
- Programmable MSCAN clock source either bus clock or oscillator clock
- Internal timer for time-stamping of received and transmitted messages
- Three low-power modes: sleep, power down, and MSCAN enable
- Global initialization of configuration registers

14.1.4 Modes of Operation

For a description of the specific MSCAN modes and the module operation related to the system operating modes refer to Section 14.4.4, “Modes of Operation”.

---

[1] Depending on the actual bit timing and the clock jitter of the PLL.

14.2 External Signal Description

The MSCAN uses two external pins.

NOTE

> On MCUs with an integrated CAN physical interface (transceiver) the MSCAN interface is connected internally to the transceiver interface. In these cases the external availability of signals TXCAN and RXCAN is optional.

14.2.1 RXCAN — CAN Receiver Input Pin

RXCAN is the MSCAN receiver input pin.

14.2.2 TXCAN — CAN Transmitter Output Pin

TXCAN is the MSCAN transmitter output pin. The TXCAN output pin represents the logic level on the CAN bus:

0 = Dominant state
1 = Recessive state

14.2.3 CAN System

A typical CAN system with MSCAN is shown in Figure 14-2. Each CAN station is connected physically to the CAN bus lines through a transceiver device. The transceiver is capable of driving the large current needed for the CAN bus and has current protection against defective CAN or defective stations.

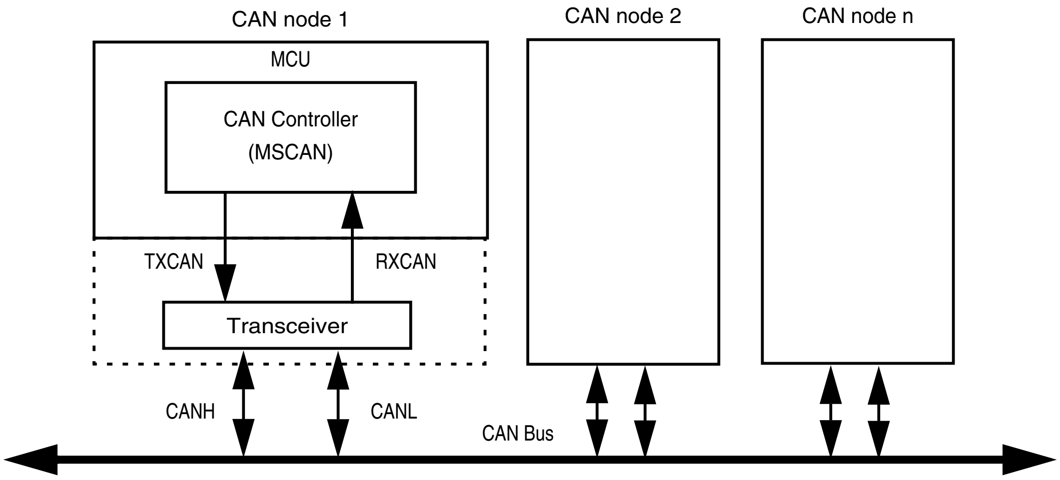


Figure 14-2. CAN System

14.3 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in the MSCAN.

14.3.1 Module Memory Map

Figure 14-3 gives an overview on all registers and their individual bits in the MSCAN memory map. The *register address* results from the addition of *base address* and *address offset*. The *base address* is determined at the MCU level and can be found in the MCU memory map description. The *address offset* is defined at the module level.

The MSCAN occupies 64 bytes in the memory space. The base address of the MSCAN module is determined at the MCU level when the MCU is defined. The register decode map is fixed and begins at the first address of the module address offset.

The detailed register descriptions follow in the order they appear in the register map.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 CANCTL0	R	RXFRM	RXACT	CSWAI	SYNCH	TIME	WUPE	SLPRQ	INITRQ
	W		▨		▨				
0x0001 CANCTL1	R	CANE	CLKSRC	LOOPB	LISTEN	BORM	WUPM	SLPAK	INITAK
	W							▨	▨
0x0002 CANBTR0	R	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
	W								
0x0003 CANBTR1	R	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
	W								
0x0004 CANRFLG	R	WUPIF	CSCIF	RSTAT1	RSTAT0	TSTAT1	TSTAT0	OVRIF	RXF
	W			▨	▨	▨	▨		
0x0005 CANRIER	R	WUPIE	CSCIE	RSTATE1	RSTATE0	TSTATE1	TSTATE0	OVRIE	RXFIE
	W								
0x0006 CANTFLG	R	0	0	0	0	0	TXE2	TXE1	TXE0
	W	▨	▨	▨	▨	▨			
0x0007 CANTIER	R	0	0	0	0	0	TXEIE2	TXEIE1	TXEIE0
	W	▨	▨	▨	▨	▨			
0x0008 CANTARQ	R	0	0	0	0	0	ABTRQ2	ABTRQ1	ABTRQ0
	W	▨	▨	▨	▨	▨			
0x0009 CANTAAK	R	0	0	0	0	0	ABTAK2	ABTAK1	ABTAK0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x000A CANTBSEL	R	0	0	0	0	0	TX2	TX1	TX0
	W	▨	▨	▨	▨	▨			
0x000B CANIDAC	R	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0
	W	▨	▨			▨	▨	▨	▨
0x000C Reserved	R	0	0	0	0	0	0	0	0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x000D CANMISC	R	0	0	0	0	0	0	0	BOHOLD
	W	▨	▨	▨	▨	▨	▨	▨	
0x000E CANRXERR	R	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
	W	▨	▨	▨	▨	▨	▨	▨	▨

▨ = Unimplemented or Reserved

Figure 14-3. MSCAN Register Summary

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x000F CANTXERR	R	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0010–0x0013 CANIDAR0–3	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
0x0014–0x0017 CANIDMRx	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
0x0018–0x001B CANIDAR4–7	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
0x001C–0x001F CANIDMR4–7	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
0x0020–0x002F CANRXFG	R	See Section 14.3.3, “Programmer’s Model of Message Storage”							
	W								
0x0030–0x003F CANTXFG	R	See Section 14.3.3, “Programmer’s Model of Message Storage”							
	W								

▨ = Unimplemented or Reserved

Figure 14-3. MSCAN Register Summary (continued)

14.3.2 Register Descriptions

This section describes in detail all the registers and register bits in the MSCAN module. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order. All bits of all registers in this module are completely synchronous to internal clocks during a register read.

14.3.2.1 MSCAN Control Register 0 (CANCTL0)

The CANCTL0 register provides various control bits of the MSCAN module as described below.

Module Base + 0x0000

Access: User read/write(1)

	7	6	5	4	3	2	1	0
R	RXFRM	RXACT	CSWAI	SYNCH	TIME	WUPE	SLPRQ	INITRQ
W		▨		▨				
Reset:	0	0	0	0	0	0	0	1

▨ = Unimplemented

Figure 14-4. MSCAN Control Register 0 (CANCTL0)

1. Read: Anytime
   Write: Anytime when out of initialization mode; exceptions are read-only RXACT and SYNCH, RXFRM (which is set by the module only), and INITRQ (which is also writable in initialization mode)

NOTE

The CANCTL0 register, except WUPE, INITRQ, and SLPRQ, is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is exited (INITRQ = 0 and INITAK = 0).

Table 14-3. CANCTL0 Register Field Descriptions

Field	Description
7 RXFRM(1)	**Received Frame Flag** — This bit is read and clear only. It is set when a receiver has received a valid message correctly, independently of the filter configuration. After it is set, it remains set until cleared by software or reset. Clearing is done by writing a 1. Writing a 0 is ignored. This bit is not valid in loopback mode. 0 No valid message was received since last clearing this flag 1 A valid message was received since last clearing of this flag
6 RXACT	**Receiver Active Status** — This read-only flag indicates the MSCAN is receiving a message. The flag is controlled by the receiver front end. This bit is not valid in loopback mode. 0 MSCAN is transmitting or idle[2] 1 MSCAN is receiving a message (including when arbitration is lost)(2)
5 CSWAI(3)	**CAN Stops in Wait Mode** — Enabling this bit allows for lower power consumption in wait mode by disabling all the clocks at the CPU bus interface to the MSCAN module. 0 The module is not affected during wait mode 1 The module ceases to be clocked during wait mode
4 SYNCH	**Synchronized Status** — This read-only flag indicates whether the MSCAN is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the MSCAN. 0 MSCAN is not synchronized to the CAN bus 1 MSCAN is synchronized to the CAN bus
3 TIME	**Timer Enable** — This bit activates an internal 16-bit wide free running timer which is clocked by the bit clock rate. If the timer is enabled, a 16-bit time stamp will be assigned to each transmitted/received message within the active TX/RX buffer. Right after the EOF of a valid message on the CAN bus, the time stamp is written to the highest bytes (0x000E, 0x000F) in the appropriate buffer (see Section 14.3.3, "Programmer's Model of Message Storage"). The internal timer is reset (all bits set to 0) when disabled. This bit is held low in initialization mode. 0 Disable internal MSCAN timer 1 Enable internal MSCAN timer
2 WUPE(4)	**Wake-Up Enable** — This configuration bit allows the MSCAN to restart from sleep mode or from power down mode (entered from sleep) when traffic on CAN is detected (see Section 14.4.5.5, "MSCAN Sleep Mode"). This bit must be configured before sleep mode entry for the selected function to take effect. 0 Wake-up disabled — The MSCAN ignores traffic on CAN 1 Wake-up enabled — The MSCAN is able to restart

Table 14-3. CANCTL0 Register Field Descriptions (continued)

Field	Description
1 SLPRQ[5]	**Sleep Mode Request** — This bit requests the MSCAN to enter sleep mode, which is an internal power saving mode (see Section 14.4.5.5, “MSCAN Sleep Mode”). The sleep mode request is serviced when the CAN bus is idle, i.e., the module is not receiving a message and all transmit buffers are empty. The module indicates entry to sleep mode by setting SLPAK = 1 (see Section 14.3.2.2, “MSCAN Control Register 1 (CANCTL1)”). SLPRQ cannot be set while the WUPIF flag is set (see Section 14.3.2.5, “MSCAN Receiver Flag Register (CANRFLG)”). Sleep mode will be active until SLPRQ is cleared by the CPU or, depending on the setting of WUPE, the MSCAN detects activity on the CAN bus and clears SLPRQ itself. 0 Running — The MSCAN functions normally 1 Sleep mode request — The MSCAN enters sleep mode when CAN bus idle
0 INITRQ[6],[7]	**Initialization Mode Request** — When this bit is set by the CPU, the MSCAN skips to initialization mode (see Section 14.4.4.5, “MSCAN Initialization Mode”). Any ongoing transmission or reception is aborted and synchronization to the CAN bus is lost. The module indicates entry to initialization mode by setting INITAK = 1 (Section 14.3.2.2, “MSCAN Control Register 1 (CANCTL1)”). The following registers enter their hard reset state and restore their default values: CANCTL0[8], CANRFLG[9], CANRIER[10], CANTFLG, CANTIER, CANTARQ, CANTAAK, and CANTBSEL. The registers CANCTL1, CANBTR0, CANBTR1, CANIDAC, CANIDAR0-7, and CANIDMR0-7 can only be written by the CPU when the MSCAN is in initialization mode (INITRQ = 1 and INITAK = 1). The values of the error counters are not affected by initialization mode. When this bit is cleared by the CPU, the MSCAN restarts and then tries to synchronize to the CAN bus. If the MSCAN is not in bus-off state, it synchronizes after 11 consecutive recessive bits on the CAN bus; if the MSCAN is in bus-off state, it continues to wait for 128 occurrences of 11 consecutive recessive bits. Writing to other bits in CANCTL0, CANRFLG, CANRIER, CANTFLG, or CANTIER must be done only after initialization mode is exited, which is INITRQ = 0 and INITAK = 0. 0 Normal operation 1 MSCAN in initialization mode

1. The MSCAN must be in normal mode for this bit to become set.
2. See the Bosch CAN 2.0A/B specification for a detailed definition of transmitter and receiver states.
3. In order to protect from accidentally violating the CAN protocol, TXCAN is immediately forced to a recessive state when the CPU enters wait (CSWAI = 1) or stop mode (see Section 14.4.5.2, “Operation in Wait Mode” and Section 14.4.5.3, “Operation in Stop Mode”).
4. The CPU has to make sure that the WUPE register and the WUPIE wake-up interrupt enable register (see Section 14.3.2.6, “MSCAN Receiver Interrupt Enable Register (CANRIER)) is enabled, if the recovery mechanism from stop or wait is required.
5. The CPU cannot clear SLPRQ before the MSCAN has entered sleep mode (SLPRQ = 1 and SLPAK = 1).
6. The CPU cannot clear INITRQ before the MSCAN has entered initialization mode (INITRQ = 1 and INITAK = 1).
7. In order to protect from accidentally violating the CAN protocol, TXCAN is immediately forced to a recessive state when the initialization mode is requested by the CPU. Thus, the recommended procedure is to bring the MSCAN into sleep mode (SLPRQ = 1 and SLPAK = 1) before requesting initialization mode.
8. Not including WUPE, INITRQ, and SLPRQ.
9. TSTAT1 and TSTAT0 are not affected by initialization mode.
10. RSTAT1 and RSTAT0 are not affected by initialization mode.

14.3.2.2 MSCAN Control Register 1 (CANCTL1)

The CANCTL1 register provides various control bits and handshake status information of the MSCAN module as described below.

Module Base + 0x0001 Access: User read/write(1)

	7	6	5	4	3	2	1	0
R	CANE	CLKSRC	LOOPB	LISTEN	BORM	WUPM	SLPAK	INITAK
W							▨	▨
Reset:	0	0	0	1	0	0	0	1

▨ = Unimplemented

Figure 14-5. MSCAN Control Register 1 (CANCTL1)

1. Read: Anytime
Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1); CANE is write once

Table 14-4. CANCTL1 Register Field Descriptions

Field	Description
7 CANE	**MSCAN Enable** 0 MSCAN module is disabled 1 MSCAN module is enabled
6 CLKSRC	**MSCAN Clock Source** — This bit defines the clock source for the MSCAN module (only for systems with a clock generation module; Section 14.4.3.2, “Clock System,” and Section Figure 14-43., “MSCAN Clocking Scheme,”). 0 MSCAN clock source is the oscillator clock 1 MSCAN clock source is the bus clock
5 LOOPB	**Loopback Self Test Mode** — When this bit is set, the MSCAN performs an internal loopback which can be used for self test operation. The bit stream output of the transmitter is fed back to the receiver internally. The RXCAN input is ignored and the TXCAN output goes to the recessive state (logic 1). The MSCAN behaves as it does normally when transmitting and treats its own transmitted message as a message received from a remote node. In this state, the MSCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated. 0 Loopback self test disabled 1 Loopback self test enabled
4 LISTEN	**Listen Only Mode** — This bit configures the MSCAN as a CAN bus monitor. When LISTEN is set, all valid CAN messages with matching ID are received, but no acknowledgement or error frames are sent out (see Section 14.4.4.4, “Listen-Only Mode”). In addition, the error counters are frozen. Listen only mode supports applications which require “hot plugging” or throughput analysis. The MSCAN is unable to transmit any messages when listen only mode is active. 0 Normal operation 1 Listen only mode activated
3 BORM	**Bus-Off Recovery Mode** — This bit configures the bus-off state recovery mode of the MSCAN. Refer to Section 14.5.2, “Bus-Off Recovery,” for details. 0 Automatic bus-off recovery (see Bosch CAN 2.0A/B protocol specification) 1 Bus-off recovery upon user request
2 WUPM	**Wake-Up Mode** — If WUPE in CANCTL0 is enabled, this bit defines whether the integrated low-pass filter is applied to protect the MSCAN from spurious wake-up (see Section 14.4.5.5, “MSCAN Sleep Mode”). 0 MSCAN wakes up on any dominant level on the CAN bus 1 MSCAN wakes up only in case of a dominant pulse on the CAN bus that has a length of T_{wup}

Table 14-4. CANCTL1 Register Field Descriptions (continued)

Field	Description
1 SLPAK	**Sleep Mode Acknowledge** — This flag indicates whether the MSCAN module has entered sleep mode (see Section 14.4.5.5, "MSCAN Sleep Mode"). It is used as a handshake flag for the SLPRQ sleep mode request. Sleep mode is active when SLPRQ = 1 and SLPAK = 1. Depending on the setting of WUPE, the MSCAN will clear the flag if it detects activity on the CAN bus while in sleep mode. 0 Running — The MSCAN operates normally 1 Sleep mode active — The MSCAN has entered sleep mode
0 INITAK	**Initialization Mode Acknowledge** — This flag indicates whether the MSCAN module is in initialization mode (see Section 14.4.4.5, "MSCAN Initialization Mode"). It is used as a handshake flag for the INITRQ initialization mode request. Initialization mode is active when INITRQ = 1 and INITAK = 1. The registers CANCTL1, CANBTR0, CANBTR1, CANIDAC, CANIDAR0–CANIDAR7, and CANIDMR0–CANIDMR7 can be written only by the CPU when the MSCAN is in initialization mode. 0 Running — The MSCAN operates normally 1 Initialization mode active — The MSCAN has entered initialization mode

14.3.2.3 MSCAN Bus Timing Register 0 (CANBTR0)

The CANBTR0 register configures various CAN bus timing parameters of the MSCAN module.

Module Base + 0x0002 Access: User read/write(1)

	7	6	5	4	3	2	1	0
R W	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
Reset:	0	0	0	0	0	0	0	0

Figure 14-6. MSCAN Bus Timing Register 0 (CANBTR0)

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

Table 14-5. CANBTR0 Register Field Descriptions

Field	Description
7-6 SJW[1:0]	**Synchronization Jump Width** — The synchronization jump width defines the maximum number of time quanta (Tq) clock cycles a bit can be shortened or lengthened to achieve resynchronization to data transitions on the CAN bus (see Table 14-6).
5-0 BRP[5:0]	**Baud Rate Prescaler** — These bits determine the time quanta (Tq) clock which is used to build up the bit timing (see Table 14-7).

Table 14-6. Synchronization Jump Width

SJW1	SJW0	Synchronization Jump Width
0	0	1 Tq clock cycle
0	1	2 Tq clock cycles
1	0	3 Tq clock cycles
1	1	4 Tq clock cycles

Table 14-7. Baud Rate Prescaler

BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	Prescaler value (P)
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
0	0	0	0	1	1	4
:	:	:	:	:	:	:
1	1	1	1	1	1	64

14.3.2.4 MSCAN Bus Timing Register 1 (CANBTR1)

The CANBTR1 register configures various CAN bus timing parameters of the MSCAN module.

Module Base + 0x0003

Access: User read/write[1]

	7	6	5	4	3	2	1	0
R W	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
Reset:	0	0	0	0	0	0	0	0

Figure 14-7. MSCAN Bus Timing Register 1 (CANBTR1)

[1] Read: Anytime
Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

Table 14-8. CANBTR1 Register Field Descriptions

Field	Description
7 SAMP	**Sampling** — This bit determines the number of CAN bus samples taken per bit time. 0 One sample per bit. 1 Three samples per bit[1]. If SAMP = 0, the resulting bit value is equal to the value of the single bit positioned at the sample point. If SAMP = 1, the resulting bit value is determined by using majority rule on the three total samples. For higher bit rates, it is recommended that only one sample is taken per bit time (SAMP = 0).
6-4 TSEG2[2:0]	**Time Segment 2** — Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point (see Figure 14-44). Time segment 2 (TSEG2) values are programmable as shown in Table 14-9.
3-0 TSEG1[3:0]	**Time Segment 1** — Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point (see Figure 14-44). Time segment 1 (TSEG1) values are programmable as shown in Table 14-10.

[1] In this case, PHASE_SEG1 must be at least 2 time quanta (Tq).

Table 14-9. Time Segment 2 Values

TSEG22	TSEG21	TSEG20	Time Segment 2
0	0	0	1 Tq clock cycle(1)
0	0	1	2 Tq clock cycles
:	:	:	:
1	1	0	7 Tq clock cycles
1	1	1	8 Tq clock cycles

1. This setting is not valid. Please refer to Table 14-37 for valid settings.

Table 14-10. Time Segment 1 Values

TSEG13	TSEG12	TSEG11	TSEG10	Time segment 1
0	0	0	0	1 Tq clock cycle(1)
0	0	0	1	2 Tq clock cycles[1]
0	0	1	0	3 Tq clock cycles[1]
0	0	1	1	4 Tq clock cycles
:	:	:	:	:
1	1	1	0	15 Tq clock cycles
1	1	1	1	16 Tq clock cycles

1. This setting is not valid. Please refer to Table 14-37 for valid settings.

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of time quanta (Tq) clock cycles per bit (as shown in Table 14-9 and Table 14-10).

Eqn. 14-1

$$\mathbf{Bit\ Time} = \frac{(\mathbf{Prescaler\ value})}{f_{\mathbf{CANCLK}}} \bullet (\mathbf{1} + \mathbf{TimeSegment1} + \mathbf{TimeSegment2})$$

14.3.2.5 MSCAN Receiver Flag Register (CANRFLG)

A flag can be cleared only by software (writing a 1 to the corresponding bit position) when the condition which caused the setting is no longer valid. Every flag has an associated interrupt enable bit in the CANRIER register.

Module Base + 0x0004

Access: User read/write(1)

	7	6	5	4	3	2	1	0
R	WUPIF	CSCIF	RSTAT1	RSTAT0	TSTAT1	TSTAT0	OVRIF	RXF
W			▨	▨	▨	▨		
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-8. MSCAN Receiver Flag Register (CANRFLG)

1. Read: Anytime
   Write: Anytime when not in initialization mode, except RSTAT[1:0] and TSTAT[1:0] flags which are read-only; write of 1 clears flag; write of 0 is ignored

NOTE

The CANRFLG register is held in the reset state[1] when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is exited (INITRQ = 0 and INITAK = 0).

Table 14-11. CANRFLG Register Field Descriptions

Field	Description
7 WUPIF	**Wake-Up Interrupt Flag** — If the MSCAN detects CAN bus activity while in sleep mode (see Section 14.4.5.5, “MSCAN Sleep Mode,”) and WUPE = 1 in CANTCTL0 (see Section 14.3.2.1, “MSCAN Control Register 0 (CANCTL0)”), the module will set WUPIF. If not masked, a wake-up interrupt is pending while this flag is set. 0 No wake-up activity observed while in sleep mode 1 MSCAN detected activity on the CAN bus and requested wake-up
6 CSCIF	**CAN Status Change Interrupt Flag** — This flag is set when the MSCAN changes its current CAN bus status due to the actual value of the transmit error counter (TEC) and the receive error counter (REC). An additional 4-bit (RSTAT[1:0], TSTAT[1:0]) status register, which is split into separate sections for TEC/REC, informs the system on the actual CAN bus status (see Section 14.3.2.6, “MSCAN Receiver Interrupt Enable Register (CANRIER)”). If not masked, an error interrupt is pending while this flag is set. CSCIF provides a blocking interrupt. That guarantees that the receiver/transmitter status bits (RSTAT/TSTAT) are only updated when no CAN status change interrupt is pending. If the TECs/RECs change their current value after the CSCIF is asserted, which would cause an additional state change in the RSTAT/TSTAT bits, these bits keep their status until the current CSCIF interrupt is cleared again. 0 No change in CAN bus status occurred since last interrupt 1 MSCAN changed current CAN bus status
5-4 RSTAT[1:0]	**Receiver Status Bits** — The values of the error counters control the actual CAN bus status of the MSCAN. As soon as the status change interrupt flag (CSCIF) is set, these bits indicate the appropriate receiver related CAN bus status of the MSCAN. The coding for the bits RSTAT1, RSTAT0 is: 00 RxOK: 0 ≤ receive error counter ≤ 96 01 RxWRN: 96 < receive error counter ≤ 127 10 RxERR: 127 < receive error counter 11 Bus-off(1): transmit error counter > 255
3-2 TSTAT[1:0]	**Transmitter Status Bits** — The values of the error counters control the actual CAN bus status of the MSCAN. As soon as the status change interrupt flag (CSCIF) is set, these bits indicate the appropriate transmitter related CAN bus status of the MSCAN. The coding for the bits TSTAT1, TSTAT0 is: 00 TxOK: 0 ≤ transmit error counter ≤ 96 01 TxWRN: 96 < transmit error counter ≤ 127 10 TxERR: 127 < transmit error counter ≤ 255 11 Bus-Off: transmit error counter > 255

---

[1] The RSTAT[1:0], TSTAT[1:0] bits are not affected by initialization mode.

Table 14-11. CANRFLG Register Field Descriptions (continued)

Field	Description
1 OVRIF	**Overrun Interrupt Flag** — This flag is set when a data overrun condition occurs. If not masked, an error interrupt is pending while this flag is set. 0 No data overrun condition 1 A data overrun detected
0 RXF[2]	**Receive Buffer Full Flag** — RXF is set by the MSCAN when a new message is shifted in the receiver FIFO. This flag indicates whether the shifted buffer is loaded with a correctly received message (matching identifier, matching cyclic redundancy code (CRC) and no other errors detected). After the CPU has read that message from the RxFG buffer in the receiver FIFO, the RXF flag must be cleared to release the buffer. A set RXF flag prohibits the shifting of the next FIFO entry into the foreground buffer (RxFG). If not masked, a receive interrupt is pending while this flag is set. 0 No new message available within the RxFG 1 The receiver FIFO is not empty. A new message is available in the RxFG

1. Redundant information for the most critical CAN bus status which is “bus-off”. This only occurs if the Tx error counter exceeds a number of 255 errors. Bus-off affects the receiver state. As soon as the transmitter leaves its bus-off state the receiver state skips to RxOK too. Refer also to TSTAT[1:0] coding in this register.
2. To ensure data integrity, do not read the receive buffer registers while the RXF flag is cleared. For MCUs with dual CPUs, reading the receive buffer registers while the RXF flag is cleared may result in a CPU fault condition.

14.3.2.6 MSCAN Receiver Interrupt Enable Register (CANRIER)

This register contains the interrupt enable bits for the interrupt flags described in the CANRFLG register.

Module Base + 0x0005

Access: User read/write[1]

	7	6	5	4	3	2	1	0
R W	WUPIE	CSCIE	RSTATE1	RSTATE0	TSTATE1	TSTATE0	OVRIE	RXFIE
Reset:	0	0	0	0	0	0	0	0

Figure 14-9. MSCAN Receiver Interrupt Enable Register (CANRIER)

1. Read: Anytime
   Write: Anytime when not in initialization mode

NOTE

> The CANRIER register is held in the reset state when the initialization mode is active (INITRQ=1 and INITAK=1). This register is writable when not in initialization mode (INITRQ=0 and INITAK=0).
>
> The RSTATE[1:0], TSTATE[1:0] bits are not affected by initialization mode.

Table 14-12. CANRIER Register Field Descriptions

Field	Description
7 WUPIE[1]	**Wake-Up Interrupt Enable** 0 No interrupt request is generated from this event. 1 A wake-up event causes a Wake-Up interrupt request.
6 CSCIE	**CAN Status Change Interrupt Enable** 0 No interrupt request is generated from this event. 1 A CAN Status Change event causes an error interrupt request.
5-4 RSTATE[1:0]	**Receiver Status Change Enable** — These RSTAT enable bits control the sensitivity level in which receiver state changes are causing CSCIF interrupts. Independent of the chosen sensitivity level the RSTAT flags continue to indicate the actual receiver state and are only updated if no CSCIF interrupt is pending. 00 Do not generate any CSCIF interrupt caused by receiver state changes. 01 Generate CSCIF interrupt only if the receiver enters or leaves “bus-off” state. Discard other receiver state changes for generating CSCIF interrupt. 10 Generate CSCIF interrupt only if the receiver enters or leaves “RxErr” or “bus-off”[2] state. Discard other receiver state changes for generating CSCIF interrupt. 11 Generate CSCIF interrupt on all state changes.
3-2 TSTATE[1:0]	**Transmitter Status Change Enable** — These TSTAT enable bits control the sensitivity level in which transmitter state changes are causing CSCIF interrupts. Independent of the chosen sensitivity level, the TSTAT flags continue to indicate the actual transmitter state and are only updated if no CSCIF interrupt is pending. 00 Do not generate any CSCIF interrupt caused by transmitter state changes. 01 Generate CSCIF interrupt only if the transmitter enters or leaves “bus-off” state. Discard other transmitter state changes for generating CSCIF interrupt. 10 Generate CSCIF interrupt only if the transmitter enters or leaves “TxErr” or “bus-off” state. Discard other transmitter state changes for generating CSCIF interrupt. 11 Generate CSCIF interrupt on all state changes.
1 OVRIE	**Overrun Interrupt Enable** 0 No interrupt request is generated from this event. 1 An overrun event causes an error interrupt request.
0 RXFIE	**Receiver Full Interrupt Enable** 0 No interrupt request is generated from this event. 1 A receive buffer full (successful message reception) event causes a receiver interrupt request.

[1] WUPIE and WUPE (see Section 14.3.2.1, “MSCAN Control Register 0 (CANCTL0)”) must both be enabled if the recovery mechanism from stop or wait is required.

[2] Bus-off state is only defined for transmitters by the CAN standard (see Bosch CAN 2.0A/B protocol specification). Because the only possible state change for the transmitter from bus-off to TxOK also forces the receiver to skip its current state to RxOK, the coding of the RXSTAT[1:0] flags define an additional bus-off state for the receiver (see Section 14.3.2.5, “MSCAN Receiver Flag Register (CANRFLG)”).

14.3.2.7 MSCAN Transmitter Flag Register (CANTFLG)

The transmit buffer empty flags each have an associated interrupt enable bit in the CANTIER register.

Module Base + 0x0006 Access: User read/write[1]

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	TXE2	TXE1	TXE0
W	▨	▨	▨	▨	▨			
Reset:	0	0	0	0	0	1	1	1

▨ = Unimplemented

Figure 14-10. MSCAN Transmitter Flag Register (CANTFLG)

[1] Read: Anytime
Write: Anytime when not in initialization mode; write of 1 clears flag, write of 0 is ignored

NOTE

The CANTFLG register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

Table 14-13. CANTFLG Register Field Descriptions

Field	Description
2-0 TXE[2:0]	**Transmitter Buffer Empty** — This flag indicates that the associated transmit message buffer is empty, and thus not scheduled for transmission. The CPU must clear the flag after a message is set up in the transmit buffer and is due for transmission. The MSCAN sets the flag after the message is sent successfully. The flag is also set by the MSCAN when the transmission request is successfully aborted due to a pending abort request (see Section 14.3.2.9, “MSCAN Transmitter Message Abort Request Register (CANTARQ)”). If not masked, a transmit interrupt is pending while this flag is set. Clearing a TXEx flag also clears the corresponding ABTAKx (see Section 14.3.2.10, “MSCAN Transmitter Message Abort Acknowledge Register (CANTAAK)”). When a TXEx flag is set, the corresponding ABTRQx bit is cleared (see Section 14.3.2.9, “MSCAN Transmitter Message Abort Request Register (CANTARQ)”). When listen-mode is active (see Section 14.3.2.2, “MSCAN Control Register 1 (CANCTL1)”) the TXEx flags cannot be cleared and no transmission is started. Read and write accesses to the transmit buffer will be blocked, if the corresponding TXEx bit is cleared (TXEx = 0) and the buffer is scheduled for transmission. 0 The associated message buffer is full (loaded with a message due for transmission) 1 The associated message buffer is empty (not scheduled)

14.3.2.8 MSCAN Transmitter Interrupt Enable Register (CANTIER)

This register contains the interrupt enable bits for the transmit buffer empty interrupt flags.

Module Base + 0x0007 Access: User read/write[1]

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	TXEIE2	TXEIE1	TXEIE0
W	▨	▨	▨	▨	▨			
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-11. MSCAN Transmitter Interrupt Enable Register (CANTIER)

1. Read: Anytime
   Write: Anytime when not in initialization mode

NOTE

The CANTIER register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

Table 14-14. CANTIER Register Field Descriptions

Field	Description
2-0 TXEIE[2:0]	**Transmitter Empty Interrupt Enable** 0 No interrupt request is generated from this event. 1 A transmitter empty (transmit buffer available for transmission) event causes a transmitter empty interrupt request.

14.3.2.9 MSCAN Transmitter Message Abort Request Register (CANTARQ)

The CANTARQ register allows abort request of queued messages as described below.

Module Base + 0x0008

Access: User read/write[1]

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	ABTRQ2	ABTRQ1	ABTRQ0
W	▨	▨	▨	▨	▨			
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-12. MSCAN Transmitter Message Abort Request Register (CANTARQ)

1. Read: Anytime
   Write: Anytime when not in initialization mode

NOTE

The CANTARQ register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

Table 14-15. CANTARQ Register Field Descriptions

Field	Description
2-0 ABTRQ[2:0]	**Abort Request** — The CPU sets the ABTRQx bit to request that a scheduled message buffer (TXEx = 0) be aborted. The MSCAN grants the request if the message has not already started transmission, or if the transmission is not successful (lost arbitration or error). When a message is aborted, the associated TXE (see Section 14.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and abort acknowledge flags (ABTAK, see Section 14.3.2.10, “MSCAN Transmitter Message Abort Acknowledge Register (CANTAAK)”) are set and a transmit interrupt occurs if enabled. The CPU cannot reset ABTRQx. ABTRQx is reset whenever the associated TXE flag is set. 0 No abort request 1 Abort request pending

14.3.2.10 MSCAN Transmitter Message Abort Acknowledge Register (CANTAAK)

The CANTAAK register indicates the successful abort of a queued message, if requested by the appropriate bits in the CANTARQ register.

Module Base + 0x0009 Access: User read/write(1)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	ABTAK2	ABTAK1	ABTAK0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-13. MSCAN Transmitter Message Abort Acknowledge Register (CANTAAK)

1. Read: Anytime
   Write: Unimplemented

NOTE

The CANTAAK register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1).

Table 14-16. CANTAAK Register Field Descriptions

Field	Description
2-0 ABTAK[2:0]	**Abort Acknowledge** — This flag acknowledges that a message was aborted due to a pending abort request from the CPU. After a particular message buffer is flagged empty, this flag can be used by the application software to identify whether the message was aborted successfully or was sent anyway. The ABTAKx flag is cleared whenever the corresponding TXE flag is cleared. 0 The message was not aborted. 1 The message was aborted.

14.3.2.11 MSCAN Transmit Buffer Selection Register (CANTBSEL)

The CANTBSEL register allows the selection of the actual transmit message buffer, which then will be accessible in the CANTXFG register space.

Module Base + 0x000A Access: User read/write(1)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	TX2	TX1	TX0
W	▨	▨	▨	▨	▨			
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-14. MSCAN Transmit Buffer Selection Register (CANTBSEL)

1. Read: Find the lowest ordered bit set to 1, all other bits will be read as 0
   Write: Anytime when not in initialization mode

NOTE

The CANTBSEL register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK=1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

Table 14-17. CANTBSEL Register Field Descriptions

Field	Description
2-0 TX[2:0]	**Transmit Buffer Select** — The lowest numbered bit places the respective transmit buffer in the CANTXFG register space (e.g., TX1 = 1 and TX0 = 1 selects transmit buffer TX0; TX1 = 1 and TX0 = 0 selects transmit buffer TX1). Read and write accesses to the selected transmit buffer will be blocked, if the corresponding TXEx bit is cleared and the buffer is scheduled for transmission (see Section 14.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”). 0 The associated message buffer is deselected 1 The associated message buffer is selected, if lowest numbered bit

The following gives a short programming example of the usage of the CANTBSEL register:

To get the next available transmit buffer, application software must read the CANTFLG register and write this value back into the CANTBSEL register. In this example Tx buffers TX1 and TX2 are available. The value read from CANTFLG is therefore 0b0000_0110. When writing this value back to CANTBSEL, the Tx buffer TX1 is selected in the CANTXFG because the lowest numbered bit set to 1 is at bit position 1. Reading back this value out of CANTBSEL results in 0b0000_0010, because only the lowest numbered bit position set to 1 is presented. This mechanism eases the application software’s selection of the next available Tx buffer.

- LDAA CANTFLG; value read is 0b0000_0110
- STAA CANTBSEL; value written is 0b0000_0110
- LDAA CANTBSEL; value read is 0b0000_0010

If all transmit message buffers are deselected, no accesses are allowed to the CANTXFG registers.

14.3.2.12 MSCAN Identifier Acceptance Control Register (CANIDAC)

The CANIDAC register is used for identifier acceptance control as described below.

Module Base + 0x000B

Access: User read/write[1]

	7	6	5	4	3	2	1	0
R	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0
W	▨	▨			▨	▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-15. MSCAN Identifier Acceptance Control Register (CANIDAC)

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1), except bits IDHITx, which are read-only

Table 14-18. CANIDAC Register Field Descriptions

Field	Description
5-4 IDAM[1:0]	**Identifier Acceptance Mode** — The CPU sets these flags to define the identifier acceptance filter organization (see Section 14.4.3, "Identifier Acceptance Filter"). Table 14-19 summarizes the different settings. In filter closed mode, no message is accepted such that the foreground buffer is never reloaded.
2-0 IDHIT[2:0]	Identifier Acceptance Hit Indicator — The MSCAN sets these flags to indicate an identifier acceptance hit (see Section 14.4.3, "Identifier Acceptance Filter"). Table 14-20 summarizes the different settings.

Table 14-19. Identifier Acceptance Mode Settings

IDAM1	IDAM0	Identifier Acceptance Mode
0	0	Two 32-bit acceptance filters
0	1	Four 16-bit acceptance filters
1	0	Eight 8-bit acceptance filters
1	1	Filter closed

Table 14-20. Identifier Acceptance Hit Indication

IDHIT2	IDHIT1	IDHIT0	Identifier Acceptance Hit
0	0	0	Filter 0 hit
0	0	1	Filter 1 hit
0	1	0	Filter 2 hit
0	1	1	Filter 3 hit
1	0	0	Filter 4 hit
1	0	1	Filter 5 hit
1	1	0	Filter 6 hit
1	1	1	Filter 7 hit

The IDHITx indicators are always related to the message in the foreground buffer (RxFG). When a message gets shifted into the foreground buffer of the receiver FIFO the indicators are updated as well.

14.3.2.13 MSCAN Reserved Register

This register is reserved for factory testing of the MSCAN module and is not available in normal system operating modes.

Module Base + 0x000C to Module Base + 0x000D

Access: User read/write(1)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-16. MSCAN Reserved Register

1. Read: Always reads zero in normal system operation modes
   Write: Unimplemented in normal system operation modes

NOTE

Writing to this register when in special system operating modes can alter the MSCAN functionality.

14.3.2.14 MSCAN Miscellaneous Register (CANMISC)

This register provides additional features.

Module Base + 0x000D

Access: User read/write(1)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	BOHOLD
W	▨	▨	▨	▨	▨	▨	▨	
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-17. MSCAN Miscellaneous Register (CANMISC)

1. Read: Anytime
   Write: Anytime; write of ‘1’ clears flag; write of ‘0’ ignored

Table 14-21. CANMISC Register Field Descriptions

Field	Description
0 BOHOLD	**Bus-off State Hold Until User Request** — If BORM is set in MSCAN Control Register 1 (CANCTL1), this bit indicates whether the module has entered the bus-off state. Clearing this bit requests the recovery from bus-off. Refer to Section 14.5.2, “Bus-Off Recovery,” for details. 0 Module is not bus-off or recovery has been requested by user in bus-off state 1 Module is bus-off and holds this state until user request

14.3.2.15 MSCAN Receive Error Counter (CANRXERR)

This register reflects the status of the MSCAN receive error counter.

Module Base + 0x000E

Access: User read/write(1)

	7	6	5	4	3	2	1	0
R	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-18. MSCAN Receive Error Counter (CANRXERR)

1. Read: Only when in sleep mode (SLPRQ = 1 and SLPAK = 1) or initialization mode (INITRQ = 1 and INITAK = 1)
   Write: Unimplemented

NOTE

Reading this register when in any other mode other than sleep or initialization mode may return an incorrect value. For MCUs with dual CPUs, this may result in a CPU fault condition.

Writing to this register when in special modes can alter the MSCAN functionality.

14.3.2.16 MSCAN Transmit Error Counter (CANTXERR)

This register reflects the status of the MSCAN transmit error counter.

Module Base + 0x000F

Access: User read/write(1)

	7	6	5	4	3	2	1	0
R	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-19. MSCAN Transmit Error Counter (CANTXERR)

1. Read: Only when in sleep mode (SLPRQ = 1 and SLPAK = 1) or initialization mode (INITRQ = 1 and INITAK = 1)
   Write: Unimplemented

NOTE

Reading this register when in any other mode other than sleep or initialization mode, may return an incorrect value. For MCUs with dual CPUs, this may result in a CPU fault condition.

Writing to this register when in special modes can alter the MSCAN functionality.

14.3.2.17 MSCAN Identifier Acceptance Registers (CANIDAR0-7)

On reception, each message is written into the background receive buffer. The CPU is only signalled to read the message if it passes the criteria in the identifier acceptance and identifier mask registers (accepted); otherwise, the message is overwritten by the next message (dropped).

The acceptance registers of the MSCAN are applied on the IDR0–IDR3 registers (see Section 14.3.3.1, "Identifier Registers (IDR0–IDR3)") of incoming messages in a bit by bit manner (see Section 14.4.3, "Identifier Acceptance Filter").

For extended identifiers, all four acceptance and mask registers are applied. For standard identifiers, only the first two (CANIDAR0/1, CANIDMR0/1) are applied.

Module Base + 0x0010 to Module Base + 0x0013 Access: User read/write[1]

	7	6	5	4	3	2	1	0
R W	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
Reset	0	0	0	0	0	0	0	0

Figure 14-20. MSCAN Identifier Acceptance Registers (First Bank) — CANIDAR0–CANIDAR3

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

Table 14-22. CANIDAR0–CANIDAR3 Register Field Descriptions

Field	Description
7-0 AC[7:0]	**Acceptance Code Bits** — AC[7:0] comprise a user-defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register.

Module Base + 0x0018 to Module Base + 0x001B Access: User read/write[1]

	7	6	5	4	3	2	1	0
R W	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
Reset	0	0	0	0	0	0	0	0

Figure 14-21. MSCAN Identifier Acceptance Registers (Second Bank) — CANIDAR4–CANIDAR7

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

Table 14-23. CANIDAR4–CANIDAR7 Register Field Descriptions

Field	Description
7-0 AC[7:0]	**Acceptance Code Bits** — AC[7:0] comprise a user-defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register.

14.3.2.18 MSCAN Identifier Mask Registers (CANIDMR0–CANIDMR7)

The identifier mask register specifies which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering. To receive standard identifiers in 32 bit filter mode, it is required to program the last three bits (AM[2:0]) in the mask registers CANIDMR1 and CANIDMR5 to “don’t care.” To receive standard identifiers in 16 bit filter mode, it is required to program the last three bits (AM[2:0]) in the mask registers CANIDMR1, CANIDMR3, CANIDMR5, and CANIDMR7 to “don’t care.”

Module Base + 0x0014 to Module Base + 0x0017

Access: User read/write(1)

	7	6	5	4	3	2	1	0
R W	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset	0	0	0	0	0	0	0	0

Figure 14-22. MSCAN Identifier Mask Registers (First Bank) — CANIDMR0–CANIDMR3

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

Table 14-24. CANIDMR0–CANIDMR3 Register Field Descriptions

Field	Description
7-0 AM[7:0]	**Acceptance Mask Bits** — If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match is detected. The message is accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register does not affect whether or not the message is accepted. 0 Match corresponding acceptance code register and identifier bits 1 Ignore corresponding acceptance code register bit

Module Base + 0x001C to Module Base + 0x001F

Access: User read/write(1)

	7	6	5	4	3	2	1	0
R W	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset	0	0	0	0	0	0	0	0

Figure 14-23. MSCAN Identifier Mask Registers (Second Bank) — CANIDMR4–CANIDMR7

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

Table 14-25. CANIDMR4–CANIDMR7 Register Field Descriptions

Field	Description
7-0 AM[7:0]	**Acceptance Mask Bits** — If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match is detected. The message is accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register does not affect whether or not the message is accepted. 0 Match corresponding acceptance code register and identifier bits 1 Ignore corresponding acceptance code register bit

14.3.3 Programmer's Model of Message Storage

The following section details the organization of the receive and transmit message buffers and the associated control registers.

To simplify the programmer interface, the receive and transmit message buffers have the same outline. Each message buffer allocates 16 bytes in the memory map containing a 13 byte data structure.

An additional transmit buffer priority register (TBPR) is defined for the transmit buffers. Within the last two bytes of this memory map, the MSCAN stores a special 16-bit time stamp, which is sampled from an internal timer after successful transmission or reception of a message. This feature is only available for transmit and receiver buffers, if the TIME bit is set (see Section 14.3.2.1, "MSCAN Control Register 0 (CANCTL0)").

The time stamp register is written by the MSCAN. The CPU can only read these registers.

Table 14-26. Message Buffer Organization

Offset Address	Register	Access
0x00X0	Identifier Register 0	R/W
0x00X1	Identifier Register 1	R/W
0x00X2	Identifier Register 2	R/W
0x00X3	Identifier Register 3	R/W
0x00X4	Data Segment Register 0	R/W
0x00X5	Data Segment Register 1	R/W
0x00X6	Data Segment Register 2	R/W
0x00X7	Data Segment Register 3	R/W
0x00X8	Data Segment Register 4	R/W
0x00X9	Data Segment Register 5	R/W
0x00XA	Data Segment Register 6	R/W
0x00XB	Data Segment Register 7	R/W
0x00XC	Data Length Register	R/W
0x00XD	Transmit Buffer Priority Register(1)	R/W
0x00XE	Time Stamp Register (High Byte)	R
0x00XF	Time Stamp Register (Low Byte)	R

1. Not applicable for receive buffers

Figure 14-24 shows the common 13-byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR registers is shown in Figure 14-25.

All bits of the receive and transmit buffers are ‘x’ out of reset because of RAM-based implementation[1]. All reserved or unused bits of the receive and transmit buffers always read ‘x’.

---

[1] Exception: The transmit buffer priority registers are 0 out of reset.

Figure 14-24. Receive/Transmit Message Buffer — Extended Identifier Mapping

Register Name		Bit 7	6	5	4	3	2	1	Bit0
0x00X0 IDR0	R W	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
0x00X1 IDR1	R W	ID20	ID19	ID18	SRR (=1)	IDE (=1)	ID17	ID16	ID15
0x00X2 IDR2	R W	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
0x00X3 IDR3	R W	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
0x00X4 DSR0	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00X5 DSR1	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00X6 DSR2	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00X7 DSR3	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00X8 DSR4	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00X9 DSR5	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00XA DSR6	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00XB DSR7	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00XC DLR	R W					DLC3	DLC2	DLC1	DLC0

Figure 14-24. Receive/Transmit Message Buffer — Extended Identifier Mapping (continued)

Register Name		Bit 7	6	5	4	3	2	1	Bit0

▨ = Unused, always read ‘x’

Read:

- For transmit buffers, anytime when TXEx flag is set (see Section 14.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and the corresponding transmit buffer is selected in CANTBSEL (see Section 14.3.2.11, “MSCAN Transmit Buffer Selection Register (CANTBSEL)”).
- For receive buffers, only when RXF flag is set (see Section 14.3.2.5, “MSCAN Receiver Flag Register (CANRFLG)”).

Write:

- For transmit buffers, anytime when TXEx flag is set (see Section 14.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and the corresponding transmit buffer is selected in CANTBSEL (see Section 14.3.2.11, “MSCAN Transmit Buffer Selection Register (CANTBSEL)”).
- Unimplemented for receive buffers.

Reset: Undefined because of RAM-based implementation

Figure 14-25. Receive/Transmit Message Buffer — Standard Identifier Mapping

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
IDR0 0x00X0	R W	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
IDR1 0x00X1	R W	ID2	ID1	ID0	RTR	IDE (=0)	▨	▨	▨
IDR2 0x00X2	R W	▨	▨	▨	▨	▨	▨	▨	▨
IDR3 0x00X3	R W	▨	▨	▨	▨	▨	▨	▨	▨

▨ = Unused, always read ‘x’

14.3.3.1 Identifier Registers (IDR0–IDR3)

The identifier registers for an extended format identifier consist of a total of 32 bits: ID[28:0], SRR, IDE, and RTR. The identifier registers for a standard format identifier consist of a total of 13 bits: ID[10:0], RTR, and IDE.

14.3.3.1.1 IDR0–IDR3 for Extended Identifier Mapping

Module Base + 0x00X0

	7	6	5	4	3	2	1	0
R W	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
Reset:	x	x	x	x	x	x	x	x

Figure 14-26. Identifier Register 0 (IDR0) — Extended Identifier Mapping

Table 14-27. IDR0 Register Field Descriptions — Extended

Field	Description
7-0 ID[28:21]	**Extended Format Identifier —** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

Module Base + 0x00X1

	7	6	5	4	3	2	1	0
R W	ID20	ID19	ID18	SRR (=1)	IDE (=1)	ID17	ID16	ID15
Reset:	x	x	x	x	x	x	x	x

Figure 14-27. Identifier Register 1 (IDR1) — Extended Identifier Mapping

Table 14-28. IDR1 Register Field Descriptions — Extended

Field	Description
7-5 ID[20:18]	**Extended Format Identifier —** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.
4 SRR	**Substitute Remote Request** — This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and is stored as received on the CAN bus for receive buffers.
3 IDE	**ID Extended** — This flag indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the flag is set as received and indicates to the CPU how to process the buffer identifier registers. In the case of a transmit buffer, the flag indicates to the MSCAN what type of identifier to send. 0 Standard format (11 bit) 1 Extended format (29 bit)
2-0 ID[17:15]	**Extended Format Identifier —** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

Module Base + 0x00X2

	7	6	5	4	3	2	1	0
R W	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
Reset:	x	x	x	x	x	x	x	x

Figure 14-28. Identifier Register 2 (IDR2) — Extended Identifier Mapping

Table 14-29. IDR2 Register Field Descriptions — Extended

Field	Description
7-0 ID[14:7]	**Extended Format Identifier —** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

Module Base + 0x00X3

	7	6	5	4	3	2	1	0
R W	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
Reset:	x	x	x	x	x	x	x	x

Figure 14-29. Identifier Register 3 (IDR3) — Extended Identifier Mapping

Table 14-30. IDR3 Register Field Descriptions — Extended

Field	Description
7-1 ID[6:0]	**Extended Format Identifier —** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.
0 RTR	**Remote Transmission Request** — This flag reflects the status of the remote transmission request bit in the CAN frame. In the case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In the case of a transmit buffer, this flag defines the setting of the RTR bit to be sent. 0 Data frame 1 Remote frame

14.3.3.1.2 IDR0–IDR3 for Standard Identifier Mapping

Module Base + 0x00X0

	7	6	5	4	3	2	1	0
R W	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
Reset:	x	x	x	x	x	x	x	x

Figure 14-30. Identifier Register 0 — Standard Mapping

Table 14-31. IDR0 Register Field Descriptions — Standard

Field	Description
7-0 ID[10:3]	**Standard Format Identifier —** The identifiers consist of 11 bits (ID[10:0]) for the standard format. ID10 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. See also ID bits in Table 14-32.

Module Base + 0x00X1

	7	6	5	4	3	2	1	0
R W	ID2	ID1	ID0	RTR	IDE (=0)	▨	▨	▨
Reset:	x	x	x	x	x	x	x	x

▨ = Unused; always read 'x'

Figure 14-31. Identifier Register 1 — Standard Mapping

Table 14-32. IDR1 Register Field Descriptions

Field	Description
7-5 ID[2:0]	**Standard Format Identifier —** The identifiers consist of 11 bits (ID[10:0]) for the standard format. ID10 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. See also ID bits in Table 14-31.
4 RTR	**Remote Transmission Request** — This flag reflects the status of the Remote Transmission Request bit in the CAN frame. In the case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In the case of a transmit buffer, this flag defines the setting of the RTR bit to be sent. 0 Data frame 1 Remote frame
3 IDE	**ID Extended** — This flag indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the flag is set as received and indicates to the CPU how to process the buffer identifier registers. In the case of a transmit buffer, the flag indicates to the MSCAN what type of identifier to send. 0 Standard format (11 bit) 1 Extended format (29 bit)

Module Base + 0x00X2

	7	6	5	4	3	2	1	0
R	▨	▨	▨	▨	▨	▨	▨	▨
W								
Reset:	x	x	x	x	x	x	x	x

▨ = Unused; always read ‘x’

Figure 14-32. Identifier Register 2 — Standard Mapping

Module Base + 0x00X3

	7	6	5	4	3	2	1	0
R	▨	▨	▨	▨	▨	▨	▨	▨
W								
Reset:	x	x	x	x	x	x	x	x

▨ = Unused; always read ‘x’

Figure 14-33. Identifier Register 3 — Standard Mapping

14.3.3.2 Data Segment Registers (DSR0-7)

The eight data segment registers, each with bits DB[7:0], contain the data to be transmitted or received. The number of bytes to be transmitted or received is determined by the data length code in the corresponding DLR register.

Module Base + 0x00X4 to Module Base + 0x00XB

	7	6	5	4	3	2	1	0
R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
W								
Reset:	x	x	x	x	x	x	x	x

Figure 14-34. Data Segment Registers (DSR0–DSR7) — Extended Identifier Mapping

Table 14-33. DSR0–DSR7 Register Field Descriptions

Field	Description
7-0 DB[7:0]	**Data bits 7-0**

14.3.3.3 Data Length Register (DLR)

This register keeps the data length field of the CAN frame.

Module Base + 0x00XC

	7	6	5	4	3	2	1	0
R W	▨	▨	▨	▨	DLC3	DLC2	DLC1	DLC0
Reset:	x	x	x	x	x	x	x	x

▨ = Unused; always read “x”

Figure 14-35. Data Length Register (DLR) — Extended Identifier Mapping

Table 14-34. DLR Register Field Descriptions

Field	Description
3-0 DLC[3:0]	**Data Length Code Bits** — The data length code contains the number of bytes (data byte count) of the respective message. During the transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted data bytes is always 0. The data byte count ranges from 0 to 8 for a data frame. Table 14-35 shows the effect of setting the DLC bits.

Table 14-35. Data Length Codes

Data Length Code				Data Byte Count
DLC3	DLC2	DLC1	DLC0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8

14.3.3.4 Transmit Buffer Priority Register (TBPR)

This register defines the local priority of the associated message buffer. The local priority is used for the internal prioritization process of the MSCAN and is defined to be highest for the smallest binary number. The MSCAN implements the following internal prioritization mechanisms:

- All transmission buffers with a cleared TXEx flag participate in the prioritization immediately before the SOF (start of frame) is sent.

- The transmission buffer with the lowest local priority field wins the prioritization.

In cases of more than one buffer having the same lowest priority, the message buffer with the lower index number wins.

Module Base + 0x00XD

Access: User read/write(1)

	7	6	5	4	3	2	1	0
R W	PRIO7	PRIO6	PRIO5	PRIO4	PRIO3	PRIO2	PRIO1	PRIO0
Reset:	0	0	0	0	0	0	0	0

Figure 14-36. Transmit Buffer Priority Register (TBPR)

1. Read: Anytime when TXEx flag is set (see Section 14.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and the corresponding transmit buffer is selected in CANTBSEL (see Section 14.3.2.11, “MSCAN Transmit Buffer Selection Register (CANTBSEL)”)
   Write: Anytime when TXEx flag is set (see Section 14.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and the corresponding transmit buffer is selected in CANTBSEL (see Section 14.3.2.11, “MSCAN Transmit Buffer Selection Register (CANTBSEL)”)

14.3.3.5 Time Stamp Register (TSRH–TSRL)

If the TIME bit is enabled, the MSCAN will write a time stamp to the respective registers in the active transmit or receive buffer right after the EOF of a valid message on the CAN bus (see Section 14.3.2.1, “MSCAN Control Register 0 (CANCTL0)”). In case of a transmission, the CPU can only read the time stamp after the respective transmit buffer has been flagged empty.

The timer value, which is used for stamping, is taken from a free running internal CAN bit clock. A timer overrun is not indicated by the MSCAN. The timer is reset (all bits set to 0) during initialization mode. The CPU can only read the time stamp registers.

Module Base + 0x00XE

Access: User read/write(1)

	7	6	5	4	3	2	1	0
R	TSR15	TSR14	TSR13	TSR12	TSR11	TSR10	TSR9	TSR8
W								
Reset:	x	x	x	x	x	x	x	x

Figure 14-37. Time Stamp Register — High Byte (TSRH)

1. Read: Anytime when TXEx flag is set (see Section 14.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and the corresponding transmit buffer is selected in CANTBSEL (see Section 14.3.2.11, “MSCAN Transmit Buffer Selection Register (CANTBSEL)”)
   Write: Unimplemented

Module Base + 0x00XF

Access: User read/write[1]

	7	6	5	4	3	2	1	0
R	TSR7	TSR6	TSR5	TSR4	TSR3	TSR2	TSR1	TSR0
W								
Reset:	x	x	x	x	x	x	x	x

Figure 14-38. Time Stamp Register — Low Byte (TSRL)

1. Read: Anytime when TXEx flag is set (see Section 14.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and the corresponding transmit buffer is selected in CANTBSEL (see Section 14.3.2.11, “MSCAN Transmit Buffer Selection Register (CANTBSEL)”)
   Write: Unimplemented

14.4 Functional Description

14.4.1 General

This section provides a complete functional description of the MSCAN.

14.4.2 Message Storage

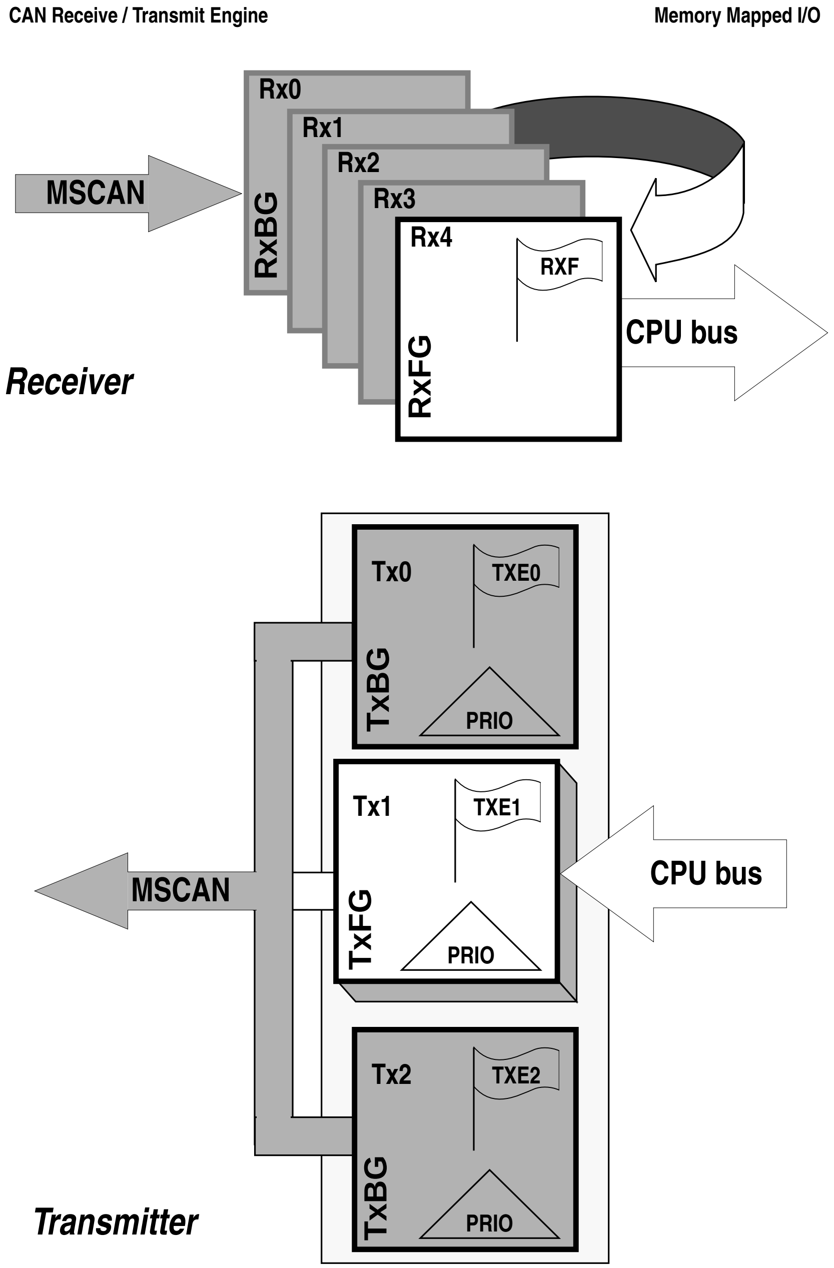


Figure 14-39. User Model for Message Buffer Organization

The MSCAN facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

14.4.2.1 Message Transmit Background

Modern application layer software is built upon two fundamental assumptions:

- Any CAN node is able to send out a stream of scheduled messages without releasing the CAN bus between the two messages. Such nodes arbitrate for the CAN bus immediately after sending the previous message and only release the CAN bus in case of lost arbitration.
- The internal message queue within any CAN node is organized such that the highest priority message is sent out first, if more than one message is ready to be sent.

The behavior described in the bullets above cannot be achieved with a single transmit buffer. That buffer must be reloaded immediately after the previous message is sent. This loading process lasts a finite amount of time and must be completed within the inter-frame sequence (IFS) to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds, it requires that the CPU reacts with short latencies to the transmit interrupt.

A double buffer scheme de-couples the reloading of the transmit buffer from the actual message sending and, therefore, reduces the reactiveness requirements of the CPU. Problems can arise if the sending of a message is finished while the CPU re-loads the second buffer. No buffer would then be ready for transmission, and the CAN bus would be released.

At least three transmit buffers are required to meet the first of the above requirements under all circumstances. The MSCAN has three transmit buffers.

The second requirement calls for some sort of internal prioritization which the MSCAN implements with the "local priority" concept described in Section 14.4.2.2, "Transmit Structures."

14.4.2.2 Transmit Structures

The MSCAN triple transmit buffer scheme optimizes real-time performance by allowing multiple messages to be set up in advance. The three buffers are arranged as shown in Figure 14-39.

All three buffers have a 13-byte data structure similar to the outline of the receive buffers (see Section 14.3.3, "Programmer's Model of Message Storage"). An additional Transmit Buffer Priority Register (TBPR) contains an 8-bit local priority field (PRIO) (see Section 14.3.3.4, "Transmit Buffer Priority Register (TBPR)"). The remaining two bytes are used for time stamping of a message, if required (see Section 14.3.3.5, "Time Stamp Register (TSRH–TSRL)").

To transmit a message, the CPU must identify an available transmit buffer, which is indicated by a set transmitter buffer empty (TXEx) flag (see Section 14.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)"). If a transmit buffer is available, the CPU must set a pointer to this buffer by writing to the CANTBSEL register (see Section 14.3.2.11, "MSCAN Transmit Buffer Selection Register (CANTBSEL)"). This makes the respective buffer accessible within the CANTXFG address space (see Section 14.3.3, "Programmer's Model of Message Storage"). The algorithmic feature associated with the CANTBSEL register simplifies the transmit buffer selection. In addition, this scheme makes the handler software simpler because only one address area is applicable for the transmit process, and the required address space is minimized.

The CPU then stores the identifier, the control bits, and the data content into one of the transmit buffers. Finally, the buffer is flagged as ready for transmission by clearing the associated TXE flag.

The MSCAN then schedules the message for transmission and signals the successful transmission of the buffer by setting the associated TXE flag. A transmit interrupt (see Section 14.4.7.2, "Transmit Interrupt") is generated[1] when TXEx is set and can be used to drive the application software to re-load the buffer.

If more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN uses the local priority setting of the three buffers to determine the prioritization. For this purpose, every transmit buffer has an 8-bit local priority field (PRIO). The application software programs this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being transmitted from this node. The lowest binary value of the PRIO field is defined to be the highest priority. The internal scheduling process takes place whenever the MSCAN arbitrates for the CAN bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software, it may become necessary to abort a lower priority message in one of the three transmit buffers. Because messages that are already in transmission cannot be aborted, the user must request the abort by setting the corresponding abort request bit (ABTRQ) (see Section 14.3.2.9, "MSCAN Transmitter Message Abort Request Register (CANTARQ)".) The MSCAN then grants the request, if possible, by:

1. Setting the corresponding abort acknowledge flag (ABTAK) in the CANTAAK register.
2. Setting the associated TXE flag to release the buffer.
3. Generating a transmit interrupt. The transmit interrupt handler software can determine from the setting of the ABTAK flag whether the message was aborted (ABTAK = 1) or sent (ABTAK = 0).

14.4.2.3 Receive Structures

The received messages are stored in a five stage input FIFO. The five message buffers are alternately mapped into a single memory area (see Figure 14-39). The background receive buffer (RxBG) is exclusively associated with the MSCAN, but the foreground receive buffer (RxFG) is addressable by the CPU (see Figure 14-39). This scheme simplifies the handler software because only one address area is applicable for the receive process.

All receive buffers have a size of 15 bytes to store the CAN control bits, the identifier (standard or extended), the data contents, and a time stamp, if enabled (see Section 14.3.3, "Programmer's Model of Message Storage").

The receiver full flag (RXF) (see Section 14.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)") signals the status of the foreground receive buffer. When the buffer contains a correctly received message with a matching identifier, this flag is set.

On reception, each message is checked to see whether it passes the filter (see Section 14.4.3, "Identifier Acceptance Filter") and simultaneously is written into the active RxBG. After successful reception of a valid message, the MSCAN shifts the content of RxBG into the receiver FIFO, sets the RXF flag, and

---

[1] The transmit interrupt occurs only if not masked. A polling scheme can be applied on TXEx also.

generates a receive interrupt[1] (see Section 14.4.7.3, "Receive Interrupt") to the CPU. The user's receive handler must read the received message from the RxFG and then reset the RXF flag to acknowledge the interrupt and to release the foreground buffer. A new message, which can follow immediately after the IFS field of the CAN frame, is received into the next available RxBG. If the MSCAN receives an invalid message in its RxBG (wrong identifier, transmission errors, etc.) the actual contents of the buffer will be over-written by the next message. The buffer will then not be shifted into the FIFO.

When the MSCAN module is transmitting, the MSCAN receives its own transmitted messages into the background receive buffer, RxBG, but does not shift it into the receiver FIFO, generate a receive interrupt, or acknowledge its own messages on the CAN bus. The exception to this rule is in loopback mode (see Section 14.3.2.2, "MSCAN Control Register 1 (CANCTL1)") where the MSCAN treats its own messages exactly like all other incoming messages. The MSCAN receives its own transmitted messages in the event that it loses arbitration. If arbitration is lost, the MSCAN must be prepared to become a receiver.

An overrun condition occurs when all receive message buffers in the FIFO are filled with correctly received messages with accepted identifiers and another message is correctly received from the CAN bus with an accepted identifier. The latter message is discarded and an error interrupt with overrun indication is generated if enabled (see Section 14.4.7.5, "Error Interrupt"). The MSCAN remains able to transmit messages while the receiver FIFO is being filled, but all incoming messages are discarded. As soon as a receive buffer in the FIFO is available again, new valid messages will be accepted.

14.4.3 Identifier Acceptance Filter

The MSCAN identifier acceptance registers (see Section 14.3.2.12, "MSCAN Identifier Acceptance Control Register (CANIDAC)") define the acceptable patterns of the standard or extended identifier (ID[10:0] or ID[28:0]). Any of these bits can be marked 'don't care' in the MSCAN identifier mask registers (see Section 14.3.2.18, "MSCAN Identifier Mask Registers (CANIDMR0–CANIDMR7)").

A filter hit is indicated to the application software by a set receive buffer full flag (RXF = 1) and three bits in the CANIDAC register (see Section 14.3.2.12, "MSCAN Identifier Acceptance Control Register (CANIDAC)"). These identifier hit flags (IDHIT[2:0]) clearly identify the filter section that caused the acceptance. They simplify the application software's task to identify the cause of the receiver interrupt. If more than one hit occurs (two or more filters match), the lower hit has priority.

A very flexible programmable generic identifier acceptance filter has been introduced to reduce the CPU interrupt loading. The filter is programmable to operate in four different modes:

- Two identifier acceptance filters, each to be applied to:
  - The full 29 bits of the extended identifier and to the following bits of the CAN 2.0B frame:
    - Remote transmission request (RTR)
    - Identifier extension (IDE)
    - Substitute remote request (SRR)
  - The 11 bits of the standard identifier plus the RTR and IDE bits of the CAN 2.0A/B messages. This mode implements two filters for a full length CAN 2.0B compliant extended identifier. Although this mode can be used for standard identifiers, it is recommended to use the four or eight identifier acceptance filters.

---

[1] The receive interrupt occurs only if not masked. A polling scheme can be applied on RXF also.

Figure 14-40 shows how the first 32-bit filter bank (CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3) produces a filter 0 hit. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces a filter 1 hit.

- Four identifier acceptance filters, each to be applied to:
  - The 14 most significant bits of the extended identifier plus the SRR and IDE bits of CAN 2.0B messages.
  - The 11 bits of the standard identifier, the RTR and IDE bits of CAN 2.0A/B messages. Figure 14-41 shows how the first 32-bit filter bank (CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3) produces filter 0 and 1 hits. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces filter 2 and 3 hits.
- Eight identifier acceptance filters, each to be applied to the first 8 bits of the identifier. This mode implements eight independent filters for the first 8 bits of a CAN 2.0A/B compliant standard identifier or a CAN 2.0B compliant extended identifier.
  Figure 14-42 shows how the first 32-bit filter bank (CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3) produces filter 0 to 3 hits. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces filter 4 to 7 hits.
- Closed filter. No CAN message is copied into the foreground buffer RxFG, and the RXF flag is never set.

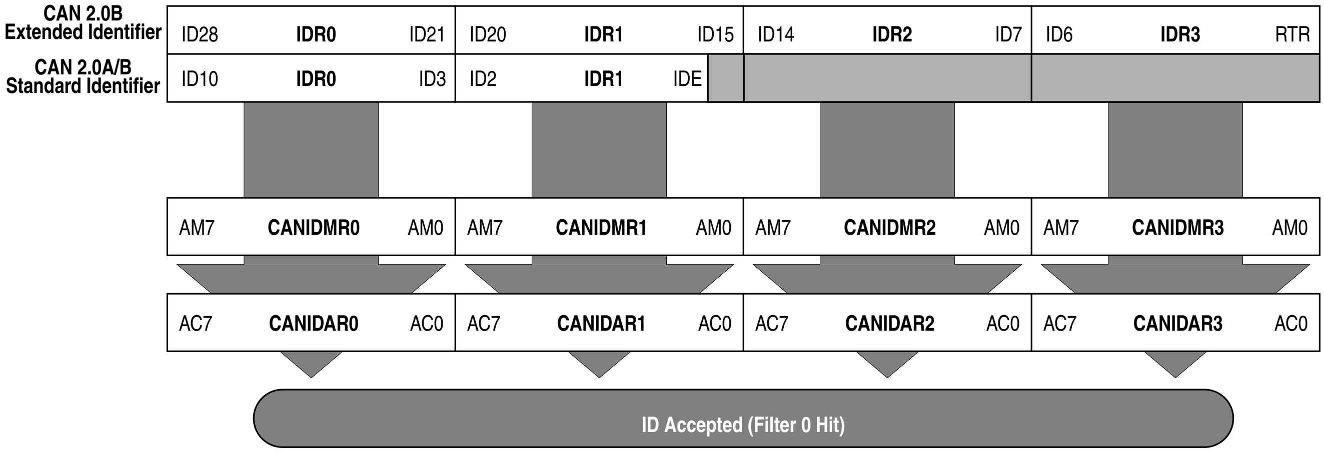


Figure 14-40. 32-bit Maskable Identifier Acceptance Filter

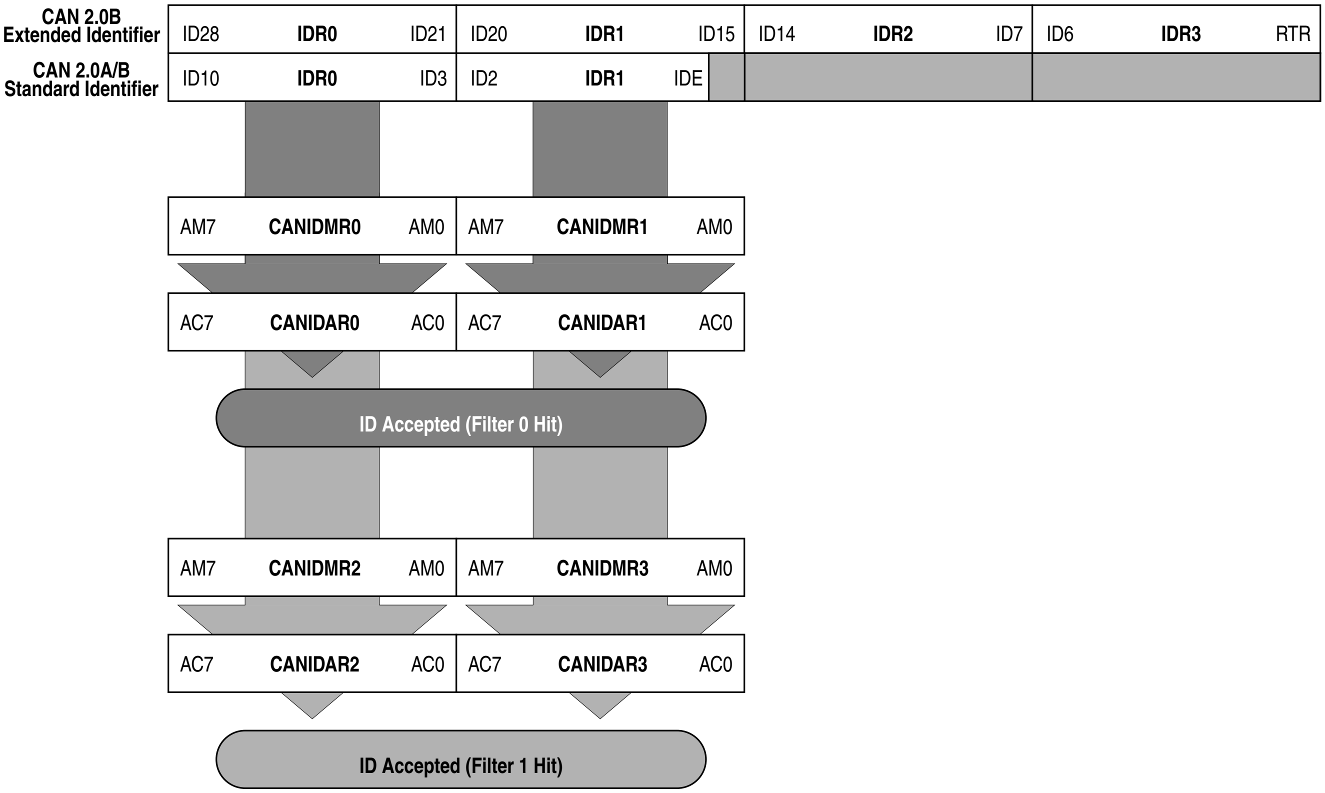


Figure 14-41. 16-bit Maskable Identifier Acceptance Filters

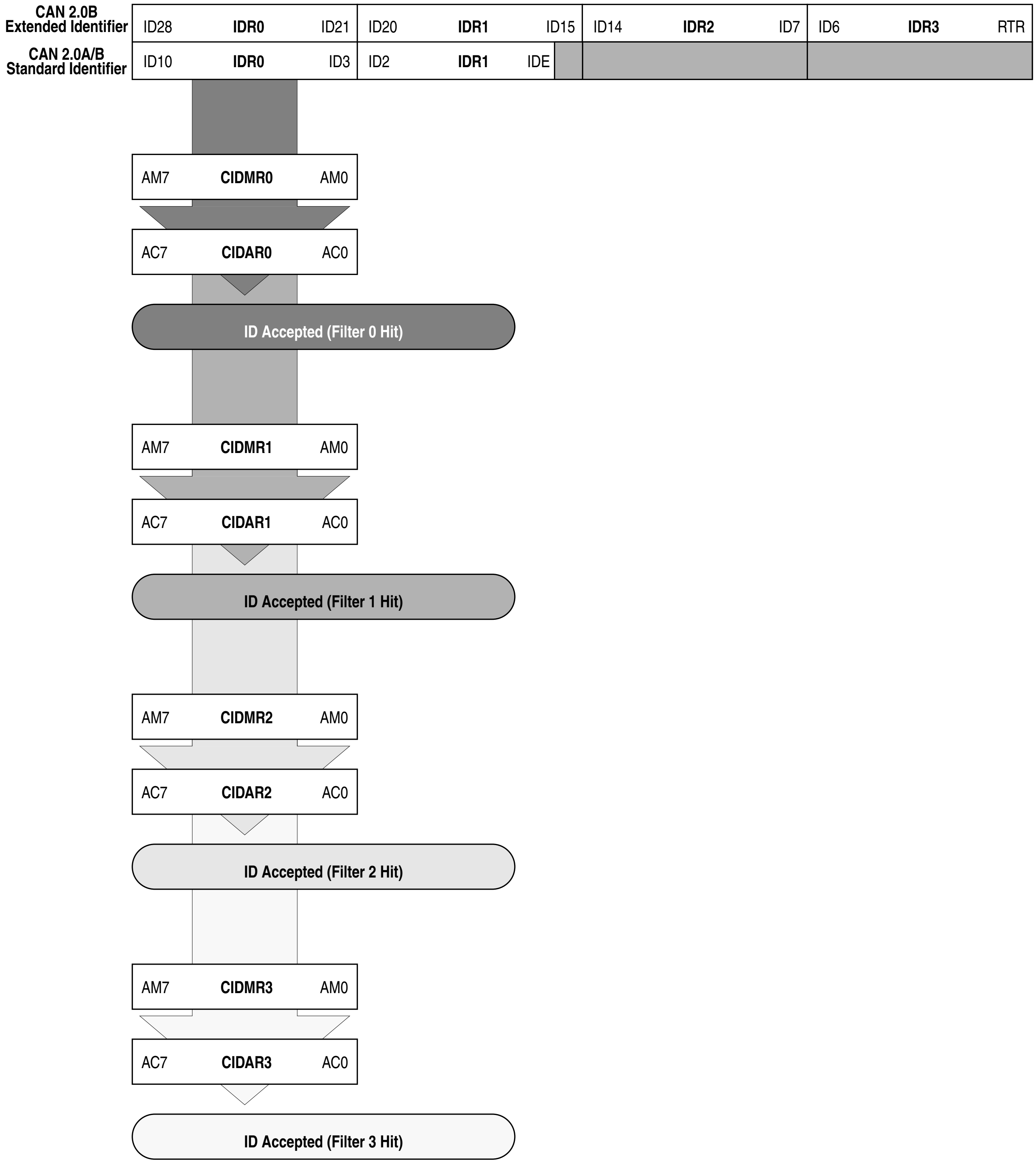


Figure 14-42. 8-bit Maskable Identifier Acceptance Filters

14.4.3.1 Protocol Violation Protection

The MSCAN protects the user from accidentally violating the CAN protocol through programming errors. The protection logic implements the following features:

- The receive and transmit error counters cannot be written or otherwise manipulated.
- All registers which control the configuration of the MSCAN cannot be modified while the MSCAN is on-line. The MSCAN has to be in Initialization Mode. The corresponding INITRQ/INITAK handshake bits in the CANCTL0/CANCTL1 registers (see Section 14.3.2.1, "MSCAN Control Register 0 (CANCTL0)") serve as a lock to protect the following registers:
  - — MSCAN control 1 register (CANCTL1)
  - — MSCAN bus timing registers 0 and 1 (CANBTR0, CANBTR1)
  - — MSCAN identifier acceptance control register (CANIDAC)
  - — MSCAN identifier acceptance registers (CANIDAR0–CANIDAR7)
  - — MSCAN identifier mask registers (CANIDMR0–CANIDMR7)
- The TXCAN is immediately forced to a recessive state when the MSCAN goes into the power down mode or initialization mode (see Section 14.4.5.6, "MSCAN Power Down Mode," and Section 14.4.4.5, "MSCAN Initialization Mode").
- The MSCAN enable bit (CANE) is writable only once in normal system operation modes, which provides further protection against inadvertently disabling the MSCAN.

14.4.3.2 Clock System

Figure 14-43 shows the structure of the MSCAN clock generation circuitry.

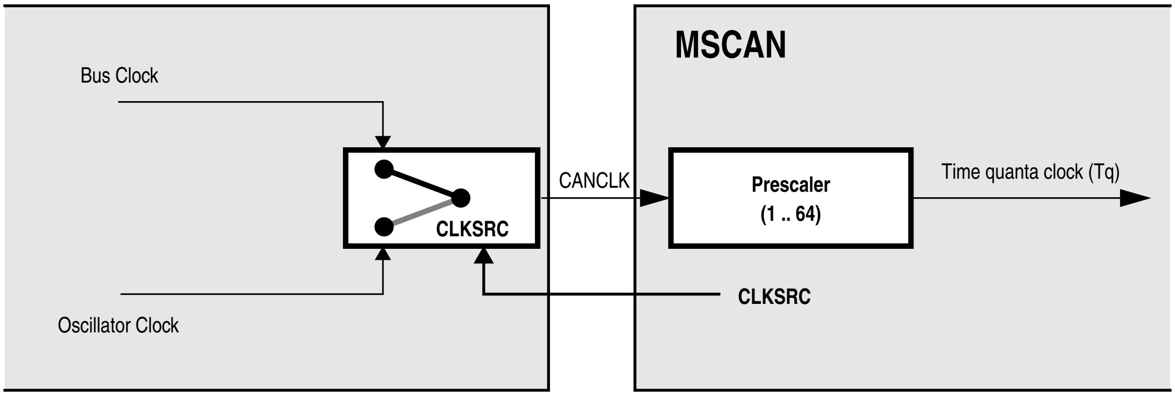


Figure 14-43. MSCAN Clocking Scheme

The clock source bit (CLKSRC) in the CANCTL1 register (14.3.2.2/14-515) defines whether the internal CANCLK is connected to the output of a crystal oscillator (oscillator clock) or to the bus clock.

The clock source has to be chosen such that the tight oscillator tolerance requirements (up to 0.4%) of the CAN protocol are met. Additionally, for high CAN bus rates (1 Mbps), a 45% to 55% duty cycle of the clock is required.

If the bus clock is generated from a PLL, it is recommended to select the oscillator clock rather than the bus clock due to jitter considerations, especially at the faster CAN bus rates.

For microcontrollers without a clock and reset generator (CRG), CANCLK is driven from the crystal oscillator (oscillator clock).

A programmable prescaler generates the time quanta (Tq) clock from CANCLK. A time quantum is the atomic unit of time handled by the MSCAN.

Eqn. 14-2

$$Tq = \frac{f_{CANCLK}}{(Prescaler\ value)}$$

A bit time is subdivided into three segments as described in the Bosch CAN 2.0A/B specification. (see Figure 14-44):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time Segment 1: This segment includes the PROP_SEG and the PHASE_SEG1 of the CAN standard. It can be programmed by setting the parameter TSEG1 to consist of 4 to 16 time quanta.
- Time Segment 2: This segment represents the PHASE_SEG2 of the CAN standard. It can be programmed by setting the TSEG2 parameter to be 2 to 8 time quanta long.

Eqn. 14-3

$$Bit\ Rate = \frac{f_{Tq}}{(number\ of\ Time\ Quanta)}$$

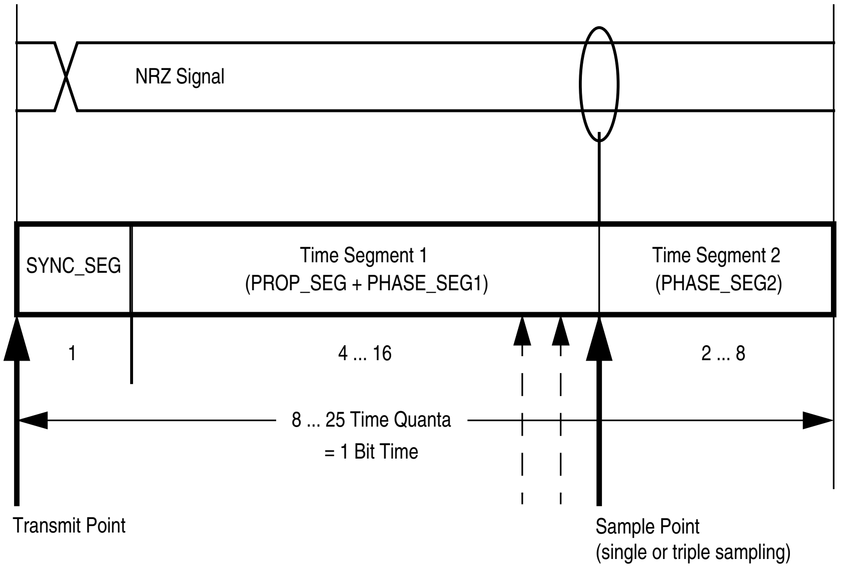


Figure 14-44. Segments within the Bit Time

Table 14-36. Time Segment Syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the CAN bus during this period.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node in receive mode samples the CAN bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

The synchronization jump width (see the Bosch CAN 2.0A/B specification for details) can be programmed in a range of 1 to 4 time quanta by setting the SJW parameter.

The SYNC_SEG, TSEG1, TSEG2, and SJW parameters are set by programming the MSCAN bus timing registers (CANBTR0, CANBTR1) (see Section 14.3.2.3, “MSCAN Bus Timing Register 0 (CANBTR0)” and Section 14.3.2.4, “MSCAN Bus Timing Register 1 (CANBTR1)”).

Table 14-37 gives an overview of the Bosch CAN 2.0A/B specification compliant segment settings and the related parameter values.

NOTE

It is the user’s responsibility to ensure the bit time settings are in compliance with the CAN standard.

Table 14-37. Bosch CAN 2.0A/B Compliant Bit Time Segment Settings

Time Segment 1	TSEG1	Time Segment 2	TSEG2	Synchronization Jump Width	SJW
5 .. 10	4 .. 9	2	1	1 .. 2	0 .. 1
4 .. 11	3 .. 10	3	2	1 .. 3	0 .. 2
5 .. 12	4 .. 11	4	3	1 .. 4	0 .. 3
6 .. 13	5 .. 12	5	4	1 .. 4	0 .. 3
7 .. 14	6 .. 13	6	5	1 .. 4	0 .. 3
8 .. 15	7 .. 14	7	6	1 .. 4	0 .. 3
9 .. 16	8 .. 15	8	7	1 .. 4	0 .. 3

14.4.4 Modes of Operation

14.4.4.1 Normal System Operating Modes

The MSCAN module behaves as described within this specification in all normal system operating modes. Write restrictions exist for some registers.

14.4.4.2 Special System Operating Modes

The MSCAN module behaves as described within this specification in all special system operating modes. Write restrictions which exist on specific registers in normal modes are lifted for test purposes in special modes.

14.4.4.3 Emulation Modes

In all emulation modes, the MSCAN module behaves just like in normal system operating modes as described within this specification.

14.4.4.4 Listen-Only Mode

In an optional CAN bus monitoring mode (listen-only), the CAN node is able to receive valid data frames and valid remote frames, but it sends only “recessive” bits on the CAN bus. In addition, it cannot start a transmission.

If the MAC sub-layer is required to send a “dominant” bit (ACK bit, overload flag, or active error flag), the bit is rerouted internally so that the MAC sub-layer monitors this “dominant” bit, although the CAN bus may remain in recessive state externally.

14.4.4.5 MSCAN Initialization Mode

The MSCAN enters initialization mode when it is enabled (CANE=1).

When entering initialization mode during operation, any on-going transmission or reception is immediately aborted and synchronization to the CAN bus is lost, potentially causing CAN protocol violations. To protect the CAN bus system from fatal consequences of violations, the MSCAN immediately drives TXCAN into a recessive state.

NOTE

> The user is responsible for ensuring that the MSCAN is not active when initialization mode is entered. The recommended procedure is to bring the MSCAN into sleep mode (SLPRQ = 1 and SLPAK = 1) before setting the INITRQ bit in the CANCTL0 register. Otherwise, the abort of an on-going message can cause an error condition and can impact other CAN bus devices.

In initialization mode, the MSCAN is stopped. However, interface registers remain accessible. This mode is used to reset the CANCTL0, CANRFLG, CANRIER, CANTFLG, CANTIER, CANTARQ, CANTAAK, and CANTBSEL registers to their default values. In addition, the MSCAN enables the configuration of the CANBTR0, CANBTR1 bit timing registers; CANIDAC; and the CANIDAR, CANIDMR message filters. See Section 14.3.2.1, “MSCAN Control Register 0 (CANCTL0),” for a detailed description of the initialization mode.

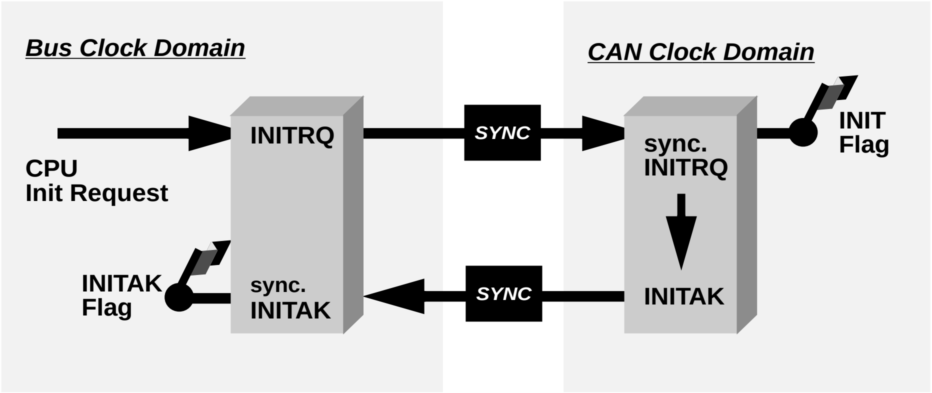


Figure 14-45. Initialization Request/Acknowledge Cycle

Due to independent clock domains within the MSCAN, INITRQ must be synchronized to all domains by using a special handshake mechanism. This handshake causes additional synchronization delay (see Figure 14-45).

If there is no message transfer ongoing on the CAN bus, the minimum delay will be two additional bus clocks and three additional CAN clocks. When all parts of the MSCAN are in initialization mode, the INITAK flag is set. The application software must use INITAK as a handshake indication for the request (INITRQ) to go into initialization mode.

NOTE

The CPU cannot clear INITRQ before initialization mode (INITRQ = 1 and INITAK = 1) is active.

14.4.5 Low-Power Options

If the MSCAN is disabled (CANE = 0), the MSCAN clocks are stopped for power saving.

If the MSCAN is enabled (CANE = 1), the MSCAN has two additional modes with reduced power consumption, compared to normal mode: sleep and power down mode. In sleep mode, power consumption is reduced by stopping all clocks except those to access the registers from the CPU side. In power down mode, all clocks are stopped and no power is consumed.

Table 14-38 summarizes the combinations of MSCAN and CPU modes. A particular combination of modes is entered by the given settings on the CSWAI and SLPRQ/SLPAK bits.

Table 14-38. CPU vs. MSCAN Operating Modes

CPU Mode	MSCAN Mode			
	Normal	Reduced Power Consumption		
		Sleep	Power Down	Disabled (CANE=0)
RUN	CSWAI = X[1] SLPRQ = 0 SLPAK = 0	CSWAI = X SLPRQ = 1 SLPAK = 1		CSWAI = X SLPRQ = X SLPAK = X
WAIT	CSWAI = 0 SLPRQ = 0 SLPAK = 0	CSWAI = 0 SLPRQ = 1 SLPAK = 1	CSWAI = 1 SLPRQ = X SLPAK = X	CSWAI = X SLPRQ = X SLPAK = X
STOP			CSWAI = X SLPRQ = X SLPAK = X	CSWAI = X SLPRQ = X SLPAK = X

[1] ‘X’ means don’t care.

14.4.5.1 Operation in Run Mode

As shown in Table 14-38, only MSCAN sleep mode is available as low power option when the CPU is in run mode.

14.4.5.2 Operation in Wait Mode

The WAI instruction puts the MCU in a low power consumption stand-by mode. If the CSWAI bit is set, additional power can be saved in power down mode because the CPU clocks are stopped. After leaving this power down mode, the MSCAN restarts and enters normal mode again.

While the CPU is in wait mode, the MSCAN can be operated in normal mode and generate interrupts (registers can be accessed via background debug mode).

14.4.5.3 Operation in Stop Mode

The STOP instruction puts the MCU in a low power consumption stand-by mode. In stop mode, the MSCAN is set in power down mode regardless of the value of the SLPRQ/SLPAK and CSWAI bits (Table 14-38).

14.4.5.4 MSCAN Normal Mode

This is a non-power-saving mode. Enabling the MSCAN puts the module from disabled mode into normal mode. In this mode the module can either be in initialization mode or out of initialization mode. See Section 14.4.4.5, “MSCAN Initialization Mode”.

14.4.5.5 MSCAN Sleep Mode

The CPU can request the MSCAN to enter this low power mode by asserting the SLPRQ bit in the CANCTL0 register. The time when the MSCAN enters sleep mode depends on a fixed synchronization delay and its current activity:

- If there are one or more message buffers scheduled for transmission (TXEx = 0), the MSCAN will continue to transmit until all transmit message buffers are empty (TXEx = 1, transmitted successfully or aborted) and then goes into sleep mode.
- If the MSCAN is receiving, it continues to receive and goes into sleep mode as soon as the CAN bus next becomes idle.
- If the MSCAN is neither transmitting nor receiving, it immediately goes into sleep mode.

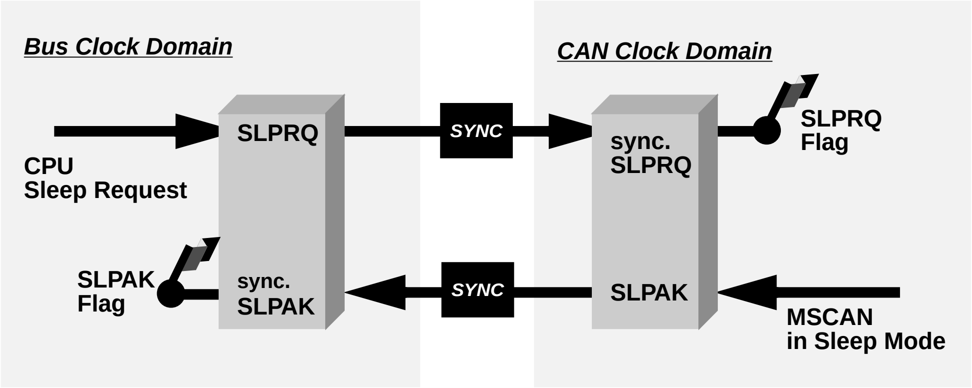


Figure 14-46. Sleep Request / Acknowledge Cycle

NOTE

The application software must avoid setting up a transmission (by clearing one or more TXEx flag(s)) and immediately request sleep mode (by setting SLPRQ). Whether the MSCAN starts transmitting or goes into sleep mode directly depends on the exact sequence of operations.

If sleep mode is active, the SLPRQ and SLPAK bits are set (Figure 14-46). The application software must use SLPAK as a handshake indication for the request (SLPRQ) to go into sleep mode.

When in sleep mode (SLPRQ = 1 and SLPAK = 1), the MSCAN stops its internal clocks. However, clocks that allow register accesses from the CPU side continue to run.

If the MSCAN is in bus-off state, it stops counting the 128 occurrences of 11 consecutive recessive bits due to the stopped clocks. TXCAN remains in a recessive state. If RXF = 1, the message can be read and RXF can be cleared. Shifting a new message into the foreground buffer of the receiver FIFO (RxFG) does not take place while in sleep mode.

It is possible to access the transmit buffers and to clear the associated TXE flags. No message abort takes place while in sleep mode.

If the WUPE bit in CANCTL0 is not asserted, the MSCAN will mask any activity it detects on CAN. RXCAN is therefore held internally in a recessive state. This locks the MSCAN in sleep mode. WUPE must be set before entering sleep mode to take effect.

The MSCAN is able to leave sleep mode (wake up) only when:

- CAN bus activity occurs and WUPE = 1

  or
- the CPU clears the SLPRQ bit

NOTE

> The CPU cannot clear the SLPRQ bit before sleep mode (SLPRQ = 1 and SLPAK = 1) is active.

After wake-up, the MSCAN waits for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, if the MSCAN is woken-up by a CAN frame, this frame is not received.

The receive message buffers (RxFG and RxBG) contain messages if they were received before sleep mode was entered. All pending actions will be executed upon wake-up; copying of RxBG into RxFG, message aborts and message transmissions. If the MSCAN remains in bus-off state after sleep mode was exited, it continues counting the 128 occurrences of 11 consecutive recessive bits.

14.4.5.6 MSCAN Power Down Mode

The MSCAN is in power down mode (Table 14-38) when

- CPU is in stop mode

  or
- CPU is in wait mode and the CSWAI bit is set

When entering the power down mode, the MSCAN immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations. To protect the CAN bus system from fatal consequences of violations to the above rule, the MSCAN immediately drives TXCAN into a recessive state.

NOTE

> The user is responsible for ensuring that the MSCAN is not active when power down mode is entered. The recommended procedure is to bring the MSCAN into Sleep mode before the STOP or WAI instruction (if CSWAI is set) is executed. Otherwise, the abort of an ongoing message can cause an error condition and impact other CAN bus devices.

In power down mode, all clocks are stopped and no registers can be accessed. If the MSCAN was not in sleep mode before power down mode became active, the module performs an internal recovery cycle after powering up. This causes some fixed delay before the module enters normal mode again.

14.4.5.7 Disabled Mode

The MSCAN is in disabled mode out of reset (CANE=0). All module clocks are stopped for power saving, however the register map can still be accessed as specified.

14.4.5.8 Programmable Wake-Up Function

The MSCAN can be programmed to wake up from sleep or power down mode as soon as CAN bus activity is detected (see control bit WUPE in MSCAN Control Register 0 (CANCTL0). The sensitivity to existing CAN bus action can be modified by applying a low-pass filter function to the RXCAN input line (see control bit WUPM in Section 14.3.2.2, “MSCAN Control Register 1 (CANCTL1)”).

This feature can be used to protect the MSCAN from wake-up due to short glitches on the CAN bus lines. Such glitches can result from—for example—electromagnetic interference within noisy environments.

14.4.6 Reset Initialization

The reset state of each individual bit is listed in Section 14.3.2, “Register Descriptions,” which details all the registers and their bit-fields.

14.4.7 Interrupts

This section describes all interrupts originated by the MSCAN. It documents the enable bits and generated flags. Each interrupt is listed and described separately.

14.4.7.1 Description of Interrupt Operation

The MSCAN supports four interrupt vectors (see Table 14-39), any of which can be individually masked (for details see Section 14.3.2.6, “MSCAN Receiver Interrupt Enable Register (CANRIER)” to Section 14.3.2.8, “MSCAN Transmitter Interrupt Enable Register (CANTIER)”).

Refer to the device overview section to determine the dedicated interrupt vector addresses.

Table 14-39. Interrupt Vectors

Interrupt Source	CCR Mask	Local Enable
Wake-Up Interrupt (WUPIF)	I bit	CANRIER (WUPIE)
Error Interrupts Interrupt (CSCIF, OVRIF)	I bit	CANRIER (CSCIE, OVRIE)
Receive Interrupt (RXF)	I bit	CANRIER (RXFIE)
Transmit Interrupts (TXE[2:0])	I bit	CANTIER (TXEIE[2:0])

14.4.7.2 Transmit Interrupt

At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. The TXEx flag of the empty message buffer is set.

14.4.7.3 Receive Interrupt

A message is successfully received and shifted into the foreground buffer (RxFG) of the receiver FIFO. This interrupt is generated immediately after receiving the EOF symbol. The RXF flag is set. If there are multiple messages in the receiver FIFO, the RXF flag is set as soon as the next message is shifted to the foreground buffer.

14.4.7.4 Wake-Up Interrupt

A wake-up interrupt is generated if activity on the CAN bus occurs during MSCAN sleep or power-down mode.

NOTE

This interrupt can only occur if the MSCAN was in sleep mode (SLPRQ = 1 and SLPAK = 1) before entering power down mode, the wake-up option is enabled (WUPE = 1), and the wake-up interrupt is enabled (WUPIE = 1).

14.4.7.5 Error Interrupt

An error interrupt is generated if an overrun of the receiver FIFO, error, warning, or bus-off condition occurrs. MSCAN Receiver Flag Register (CANRFLG) indicates one of the following conditions:

- **Overrun** — An overrun condition of the receiver FIFO as described in Section 14.4.2.3, "Receive Structures," occurred.
- **CAN Status Change** — The actual value of the transmit and receive error counters control the CAN bus state of the MSCAN. As soon as the error counters skip into a critical range (Tx/Rx-warning, Tx/Rx-error, bus-off) the MSCAN flags an error condition. The status change, which caused the error condition, is indicated by the TSTAT and RSTAT flags (see Section 14.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)" and Section 14.3.2.6, "MSCAN Receiver Interrupt Enable Register (CANRIER)").

14.4.7.6 Interrupt Acknowledge

Interrupts are directly associated with one or more status flags in either the MSCAN Receiver Flag Register (CANRFLG) or the MSCAN Transmitter Flag Register (CANTFLG). Interrupts are pending as long as one of the corresponding flags is set. The flags in CANRFLG and CANTFLG must be reset within the interrupt handler to handshake the interrupt. The flags are reset by writing a 1 to the corresponding bit position. A flag cannot be cleared if the respective condition prevails.

NOTE

It must be guaranteed that the CPU clears only the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

14.5 Initialization/Application Information

14.5.1 MSCAN initialization

The procedure to initially start up the MSCAN module out of reset is as follows:

1. Assert CANE
2. Write to the configuration registers in initialization mode
3. Clear INITRQ to leave initialization mode

If the configuration of registers which are only writable in initialization mode shall be changed:

1. Bring the module into sleep mode by setting SLPRQ and awaiting SLPAK to assert after the CAN bus becomes idle.
2. Enter initialization mode: assert INITRQ and await INITAK
3. Write to the configuration registers in initialization mode
4. Clear INITRQ to leave initialization mode and continue

14.5.2 Bus-Off Recovery

The bus-off recovery is user configurable. The bus-off state can either be left automatically or on user request.

For reasons of backwards compatibility, the MSCAN defaults to automatic recovery after reset. In this case, the MSCAN will become error active again after counting 128 occurrences of 11 consecutive recessive bits on the CAN bus (see the Bosch CAN 2.0 A/B specification for details).

If the MSCAN is configured for user request (BORM set in MSCAN Control Register 1 (CANCTL1)), the recovery from bus-off starts after both independent events have become true:

- 128 occurrences of 11 consecutive recessive bits on the CAN bus have been monitored
- BOHOLD in MSCAN Miscellaneous Register (CANMISC) has been cleared by the user

These two events may occur in any order.

Chapter 15
Serial Communication Interface (S12SCIV5)

Table 15-1. Revision History

Version Number	Revision Date	Effective Date	Author	Description of Changes
05.03	12/25/2008			remove redundancy comments in Figure1-2
05.04	08/05/2009			fix typo, SCIBDL reset value be 0x04, not 0x00
05.05	06/03/2010			fix typo, Table 15-4,SCICR1 Even parity should be PT=0 fix typo, on page 15-583,should be BKDIF,not BLDIF

15.1 Introduction

This block guide provides an overview of the serial communication interface (SCI) module.

The SCI allows asynchronous serial communications with peripheral devices and other CPUs.

15.1.1 Glossary

IR: InfraRed

IrDA: Infrared Design Associate

IRQ: Interrupt Request

LIN: Local Interconnect Network

LSB: Least Significant Bit

MSB: Most Significant Bit

NRZ: Non-Return-to-Zero

RZI: Return-to-Zero-Inverted

RXD: Receive Pin

SCI : Serial Communication Interface

TXD: Transmit Pin

15.1.2 Features

The SCI includes these distinctive features:

- Full-duplex or single-wire operation
- Standard mark/space non-return-to-zero (NRZ) format
- Selectable IrDA 1.4 return-to-zero-inverted (RZI) format with programmable pulse widths
- 13-bit baud rate selection
- Programmable 8-bit or 9-bit data format
- Separately enabled transmitter and receiver
- Programmable polarity for transmitter and receiver
- Programmable transmitter output parity
- Two receiver wakeup methods:
  - Idle line wakeup
  - Address mark wakeup
- Interrupt-driven operation with eight flags:
  - Transmitter empty
  - Transmission complete
  - Receiver full
  - Idle receiver input
  - Receiver overrun
  - Noise error
  - Framing error
  - Parity error
  - Receive wakeup on active edge
  - Transmit collision detect supporting LIN
  - Break Detect supporting LIN
- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection

15.1.3 Modes of Operation

The SCI functions the same in normal, special, and emulation modes. It has two low power modes, wait and stop modes.

- Run mode
- Wait mode
- Stop mode

15.1.4 Block Diagram

Figure 15-1 is a high level block diagram of the SCI module, showing the interaction of various function blocks.

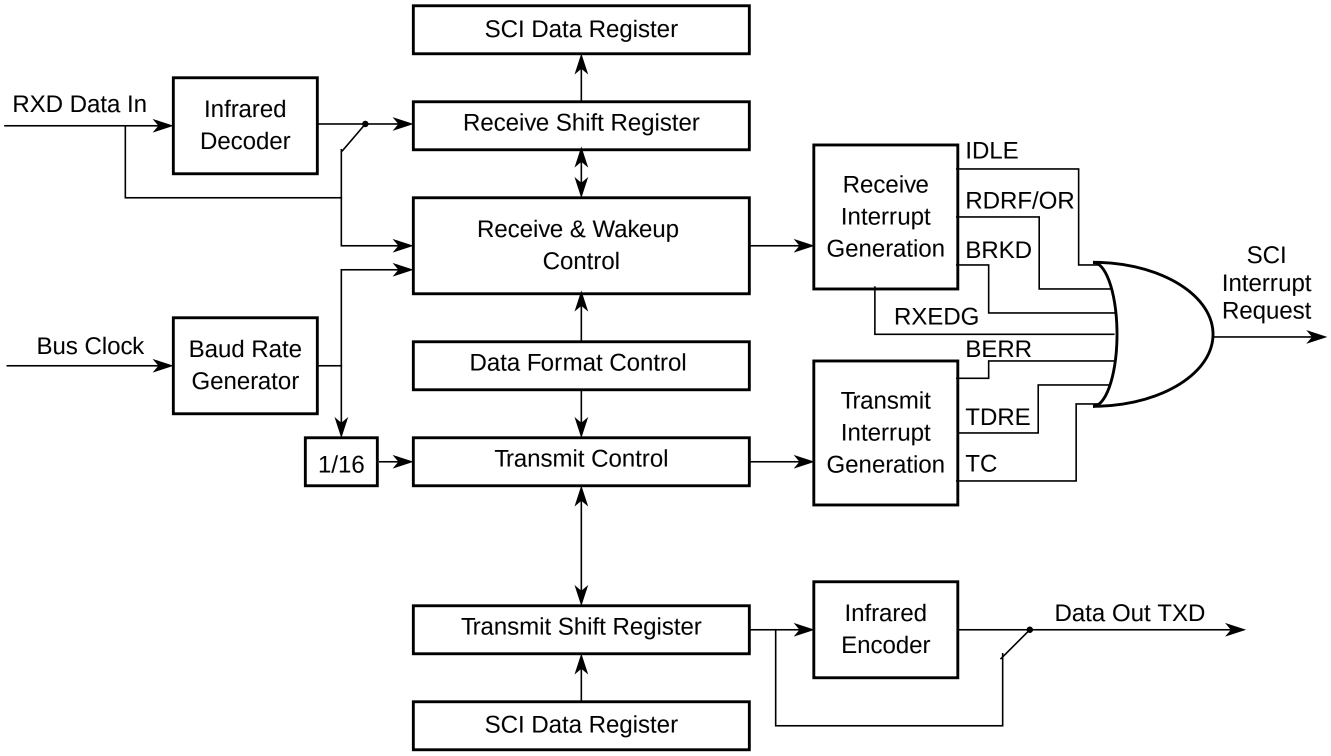


Figure 15-1. SCI Block Diagram

15.2 External Signal Description

The SCI module has a total of two external pins.

15.2.1 TXD — Transmit Pin

The TXD pin transmits SCI (standard or infrared) data. It will idle high in either mode and is high impedance anytime the transmitter is disabled.

15.2.2 RXD — Receive Pin

The RXD pin receives SCI (standard or infrared) data. An idle line is detected as a line high. This input is ignored when the receiver is disabled and should be terminated to a known voltage.

15.3 Memory Map and Register Definition

This section provides a detailed description of all the SCI registers.

15.3.1 Module Memory Map and Register Definition

The memory map for the SCI module is given below in Figure 15-2. The address listed for each register is the address offset. The total address for each register is the sum of the base address for the SCI module and the address offset for each register.

15.3.2 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Writes to a reserved register locations do not have any effect and reads of these locations return a zero. Details of register bit and field function follow the register diagrams, in bit order.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 SCIBDH[1]	R	IREN	TNP1	TNP0	SBR12	SBR11	SBR10	SBR9	SBR8
	W								
0x0001 SCIBDL[1]	R	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
	W								
0x0002 SCICR1[1]	R	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
	W								
0x0000 SCIASR1[2]	R	RXEDGIF	0	0	0	0	BERRV	BERRIF	BKDIF
	W		▨	▨	▨	▨			
0x0001 SCIACR1[2]	R	RXEDGIE	0	0	0	0	0	BERRIE	BKDIE
	W		▨	▨	▨	▨	▨		
0x0002 SCIACR2[2]	R	0	0	0	0	0	BERRM1	BERRM0	BKDFE
	W	▨	▨	▨	▨	▨			
0x0003 SCICR2	R	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
	W								
0x0004 SCISR1	R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0005 SCISR2	R	AMAP	0	0	TXPOL	RXPOL	BRK13	TXDIR	RAF
	W		▨	▨					▨
0x0006 SCIDRH	R	R8	T8	0	0	0	0	0	0
	W	▨		▨	▨	▨	▨	▨	▨
0x0007 SCIDRL	R	R7	R6	R5	R4	R3	R2	R1	R0
	W	T7	T6	T5	T4	T3	T2	T1	T0

1.These registers are accessible if the AMAP bit in the SCISR2 register is set to zero.

2,These registers are accessible if the AMAP bit in the SCISR2 register is set to one.

▨ = Unimplemented or Reserved

Figure 15-2. SCI Register Summary

15.3.2.1 SCI Baud Rate Registers (SCIBDH, SCIBDL)

Module Base + 0x0000

	7	6	5	4	3	2	1	0
R W	IREN	TNP1	TNP0	SBR12	SBR11	SBR10	SBR9	SBR8
Reset	0	0	0	0	0	0	0	0

Figure 15-3. SCI Baud Rate Register (SCIBDH)

Module Base + 0x0001

	7	6	5	4	3	2	1	0
R W	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
Reset	0	0	0	0	0	1	0	0

Figure 15-4. SCI Baud Rate Register (SCIBDL)

Read: Anytime, if AMAP = 0. If only SCIBDH is written to, a read will not return the correct data until SCIBDL is written to as well, following a write to SCIBDH.

Write: Anytime, if AMAP = 0.

NOTE

Those two registers are only visible in the memory map if AMAP = 0 (reset condition).

The SCI baud rate register is used by to determine the baud rate of the SCI, and to control the infrared modulation/demodulation submodule.

Table 15-2. SCIBDH and SCIBDL Field Descriptions

Field	Description
7 IREN	**Infrared Enable Bit** — This bit enables/disables the infrared modulation/demodulation submodule. 0 IR disabled 1 IR enabled
6:5 TNP[1:0]	**Transmitter Narrow Pulse Bits** — These bits enable whether the SCI transmits a 1/16, 3/16, 1/32 or 1/4 narrow pulse. See Table 15-3.
4:0 7:0 SBR[12:0]	**SCI Baud Rate Bits** — The baud rate for the SCI is determined by the bits in this register. The baud rate is calculated two different ways depending on the state of the IREN bit. The formulas for calculating the baud rate are: When IREN = 0 then, SCI baud rate = SCI bus clock / (16 x SBR[12:0]) When IREN = 1 then, SCI baud rate = SCI bus clock / (32 x SBR[12:1]) **Note:** The baud rate generator is disabled after reset and not started until the TE bit or the RE bit is set for the first time. The baud rate generator is disabled when (SBR[12:0] = 0 and IREN = 0) or (SBR[12:1] = 0 and IREN = 1). **Note:** Writing to SCIBDH has no effect without writing to SCIBDL, because writing to SCIBDH puts the data in a temporary location until SCIBDL is written to.

Table 15-3. IRSCI Transmit Pulse Width

TNP[1:0]	Narrow Pulse Width
11	1/4
10	1/32
01	1/16
00	3/16

15.3.2.2 SCI Control Register 1 (SCICR1)

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R W	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
Reset	0	0	0	0	0	0	0	0

Figure 15-5. SCI Control Register 1 (SCICR1)

Read: Anytime, if AMAP = 0.

Write: Anytime, if AMAP = 0.

NOTE

This register is only visible in the memory map if AMAP = 0 (reset condition).

Table 15-4. SCICR1 Field Descriptions

Field	Description
7 LOOPS	**Loop Select Bit** — LOOPS enables loop operation. In loop operation, the RXD pin is disconnected from the SCI and the transmitter output is internally connected to the receiver input. Both the transmitter and the receiver must be enabled to use the loop function. 0 Normal operation enabled 1 Loop operation enabled The receiver input is determined by the RSRC bit.
6 SCISWAI	**SCI Stop in Wait Mode Bit** — SCISWAI disables the SCI in wait mode. 0 SCI enabled in wait mode 1 SCI disabled in wait mode
5 RSRC	**Receiver Source Bit** — When LOOPS = 1, the RSRC bit determines the source for the receiver shift register input. See Table 15-5. 0 Receiver input internally connected to transmitter output 1 Receiver input connected externally to transmitter
4 M	**Data Format Mode Bit** — MODE 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
3 WAKE	**Wakeup Condition Bit** — WAKE determines which condition wakes up the SCI: a logic 1 (address mark) in the most significant bit position of a received data character or an idle condition on the RXD pin. 0 Idle line wakeup 1 Address mark wakeup

Table 15-4. SCICR1 Field Descriptions (continued)

Field	Description
2 ILT	**Idle Line Type Bit** — ILT determines when the receiver starts counting logic 1s as idle character bits. The counting begins either after the start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit may cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions. 0 Idle character bit count begins after start bit 1 Idle character bit count begins after stop bit
1 PE	**Parity Enable Bit** — PE enables the parity function. When enabled, the parity function inserts a parity bit in the most significant bit position. 0 Parity function disabled 1 Parity function enabled
0 PT	**Parity Type Bit** — PT determines whether the SCI generates and checks for even parity or odd parity. With even parity, an even number of 1s clears the parity bit and an odd number of 1s sets the parity bit. With odd parity, an odd number of 1s clears the parity bit and an even number of 1s sets the parity bit. 0 Even parity 1 Odd parity

Table 15-5. Loop Functions

LOOPS	RSRC	Function
0	x	Normal operation
1	0	Loop mode with transmitter output internally connected to receiver input
1	1	Single-wire mode with TXD pin connected to receiver input

15.3.2.3 SCI Alternative Status Register 1 (SCIASR1)

Module Base + 0x0000

	7	6	5	4	3	2	1	0
R	RXEDGIF	0	0	0	0	BERRV	BERRIF	BKDIF
W		▨	▨	▨	▨	▨		
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 15-6. SCI Alternative Status Register 1 (SCIASR1)

Read: Anytime, if AMAP = 1

Write: Anytime, if AMAP = 1

Table 15-6. SCIASR1 Field Descriptions

Field	Description
7 RXEDGIF	**Receive Input Active Edge Interrupt Flag** — RXEDGIF is asserted, if an active edge (falling if RXPOL = 0, rising if RXPOL = 1) on the RXD input occurs. RXEDGIF bit is cleared by writing a “1” to it. 0 No active receive on the receive input has occurred 1 An active edge on the receive input has occurred
2 BERRV	**Bit Error Value** — BERRV reflects the state of the RXD input when the bit error detect circuitry is enabled and a mismatch to the expected value happened. The value is only meaningful, if BERRIF = 1. 0 A low input was sampled, when a high was expected 1 A high input reassembled, when a low was expected
1 BERRIF	**Bit Error Interrupt Flag** — BERRIF is asserted, when the bit error detect circuitry is enabled and if the value sampled at the RXD input does not match the transmitted value. If the BERRIE interrupt enable bit is set an interrupt will be generated. The BERRIF bit is cleared by writing a “1” to it. 0 No mismatch detected 1 A mismatch has occurred
0 BKDIF	**Break Detect Interrupt Flag** — BKDIF is asserted, if the break detect circuitry is enabled and a break signal is received. If the BKDIE interrupt enable bit is set an interrupt will be generated. The BKDIF bit is cleared by writing a “1” to it. 0 No break signal was received 1 A break signal was received

15.3.2.4 SCI Alternative Control Register 1 (SCIACR1)

Module Base + 0x0001

	7	6	5	4	3	2	1	0
R	RXEDGIE	0	0	0	0	0	BERRIE	BKDIE
W		▨	▨	▨	▨	▨		
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 15-7. SCI Alternative Control Register 1 (SCIACR1)

Read: Anytime, if AMAP = 1

Write: Anytime, if AMAP = 1

Table 15-7. SCIACR1 Field Descriptions

Field	Description
7 RSEDGIE	**Receive Input Active Edge Interrupt Enable** — RXEDGIE enables the receive input active edge interrupt flag, RXEDGIF, to generate interrupt requests. 0 RXEDGIF interrupt requests disabled 1 RXEDGIF interrupt requests enabled
1 BERRIE	**Bit Error Interrupt Enable** — BERRIE enables the bit error interrupt flag, BERRIF, to generate interrupt requests. 0 BERRIF interrupt requests disabled 1 BERRIF interrupt requests enabled
0 BKDIE	**Break Detect Interrupt Enable** — BKDIE enables the break detect interrupt flag, BKDIF, to generate interrupt requests. 0 BKDIF interrupt requests disabled 1 BKDIF interrupt requests enabled

15.3.2.5 SCI Alternative Control Register 2 (SCIACR2)

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	BERRM1	BERRM0	BKDFE
W	▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 15-8. SCI Alternative Control Register 2 (SCIACR2)

Read: Anytime, if AMAP = 1

Write: Anytime, if AMAP = 1

Table 15-8. SCIACR2 Field Descriptions

Field	Description
2:1 BERRM[1:0]	**Bit Error Mode** — Those two bits determines the functionality of the bit error detect feature. See Table 15-9.
0 BKDFE	**Break Detect Feature Enable** — BKDFE enables the break detect circuitry. 0 Break detect circuit disabled 1 Break detect circuit enabled

Table 15-9. Bit Error Mode Coding

BERRM1	BERRM0	Function
0	0	Bit error detect circuit is disabled
0	1	Receive input sampling occurs during the 9th time tick of a transmitted bit (refer to Figure 15-19)
1	0	Receive input sampling occurs during the 13th time tick of a transmitted bit (refer to Figure 15-19)
1	1	Reserved

15.3.2.6 SCI Control Register 2 (SCICR2)

Module Base + 0x0003

	7	6	5	4	3	2	1	0
R W	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0

Figure 15-9. SCI Control Register 2 (SCICR2)

Read: Anytime

Write: Anytime

Table 15-10. SCICR2 Field Descriptions

Field	Description
7 TIE	**Transmitter Interrupt Enable Bit** — TIE enables the transmit data register empty flag, TDRE, to generate interrupt requests. 0 TDRE interrupt requests disabled 1 TDRE interrupt requests enabled
6 TCIE	**Transmission Complete Interrupt Enable Bit** — TCIE enables the transmission complete flag, TC, to generate interrupt requests. 0 TC interrupt requests disabled 1 TC interrupt requests enabled
5 RIE	**Receiver Full Interrupt Enable Bit** — RIE enables the receive data register full flag, RDRF, or the overrun flag, OR, to generate interrupt requests. 0 RDRF and OR interrupt requests disabled 1 RDRF and OR interrupt requests enabled
4 ILIE	**Idle Line Interrupt Enable Bit** — ILIE enables the idle line flag, IDLE, to generate interrupt requests. 0 IDLE interrupt requests disabled 1 IDLE interrupt requests enabled
3 TE	**Transmitter Enable Bit** — TE enables the SCI transmitter and configures the TXD pin as being controlled by the SCI. The TE bit can be used to queue an idle preamble. 0 Transmitter disabled 1 Transmitter enabled
2 RE	**Receiver Enable Bit** — RE enables the SCI receiver. 0 Receiver disabled 1 Receiver enabled
1 RWU	**Receiver Wakeup Bit** — Standby state 0 Normal operation. 1 RWU enables the wakeup function and inhibits further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing RWU.
0 SBK	**Send Break Bit** — Toggling SBK sends one break character (10 or 11 logic 0s, respectively 13 or 14 logics 0s if BRK13 is set). Toggling implies clearing the SBK bit before the break character has finished transmitting. As long as SBK is set, the transmitter continues to send complete break characters (10 or 11 bits, respectively 13 or 14 bits). 0 No break characters 1 Transmit break characters

15.3.2.7 SCI Status Register 1 (SCISR1)

The SCISR1 and SCISR2 registers provides inputs to the MCU for generation of SCI interrupts. Also, these registers can be polled by the MCU to check the status of these bits. The flag-clearing procedures require that the status register be read followed by a read or write to the SCI data register.It is permissible to execute other instructions between the two steps as long as it does not compromise the handling of I/O, but the order of operations is important for flag clearing.

Module Base + 0x0004

	7	6	5	4	3	2	1	0
R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	1	1	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 15-10. SCI Status Register 1 (SCISR1)

Read: Anytime

Write: Has no meaning or effect

Table 15-11. SCISR1 Field Descriptions

Field	Description
7 TDRE	**Transmit Data Register Empty Flag** — TDRE is set when the transmit shift register receives a byte from the SCI data register. When TDRE is 1, the transmit data register (SCIDRH/L) is empty and can receive a new value to transmit.Clear TDRE by reading SCI status register 1 (SCISR1), with TDRE set and then writing to SCI data register low (SCIDRL). 0 No byte transferred to transmit shift register 1 Byte transferred to transmit shift register; transmit data register empty
6 TC	**Transmit Complete Flag** — TC is set low when there is a transmission in progress or when a preamble or break character is loaded. TC is set high when the TDRE flag is set and no data, preamble, or break character is being transmitted.When TC is set, the TXD pin becomes idle (logic 1). Clear TC by reading SCI status register 1 (SCISR1) with TC set and then writing to SCI data register low (SCIDRL). TC is cleared automatically when data, preamble, or break is queued and ready to be sent. TC is cleared in the event of a simultaneous set and clear of the TC flag (transmission not complete). 0 Transmission in progress 1 No transmission in progress
5 RDRF	**Receive Data Register Full Flag** — RDRF is set when the data in the receive shift register transfers to the SCI data register. Clear RDRF by reading SCI status register 1 (SCISR1) with RDRF set and then reading SCI data register low (SCIDRL). 0 Data not available in SCI data register 1 Received data available in SCI data register
4 IDLE	**Idle Line Flag** — IDLE is set when 10 consecutive logic 1s (if M = 0) or 11 consecutive logic 1s (if M =1) appear on the receiver input. Once the IDLE flag is cleared, a valid frame must again set the RDRF flag before an idle condition can set the IDLE flag.Clear IDLE by reading SCI status register 1 (SCISR1) with IDLE set and then reading SCI data register low (SCIDRL). 0 Receiver input is either active now or has never become active since the IDLE flag was last cleared 1 Receiver input has become idle **Note:** When the receiver wakeup bit (RWU) is set, an idle line condition does not set the IDLE flag.

Table 15-11. SCISR1 Field Descriptions (continued)

Field	Description
3 OR	**Overrun Flag** — OR is set when software fails to read the SCI data register before the receive shift register receives the next frame. The OR bit is set immediately after the stop bit has been completely received for the second frame. The data in the shift register is lost, but the data already in the SCI data registers is not affected. Clear OR by reading SCI status register 1 (SCISR1) with OR set and then reading SCI data register low (SCIDRL). 0 No overrun 1 Overrun **Note:** OR flag may read back as set when RDRF flag is clear. This may happen if the following sequence of events occurs: 1. After the first frame is received, read status register SCISR1 (returns RDRF set and OR flag clear); 2. Receive second frame without reading the first frame in the data register (the second frame is not received and OR flag is set); 3. Read data register SCIDRL (returns first frame and clears RDRF flag in the status register); 4. Read status register SCISR1 (returns RDRF clear and OR set). Event 3 may be at exactly the same time as event 2 or any time after. When this happens, a dummy SCIDRL read following event 4 will be required to clear the OR flag if further frames are to be received.
2 NF	**Noise Flag** — NF is set when the SCI detects noise on the receiver input. NF bit is set during the same cycle as the RDRF flag but does not get set in the case of an overrun. Clear NF by reading SCI status register 1(SCISR1), and then reading SCI data register low (SCIDRL). 0 No noise 1 Noise
1 FE	**Framing Error Flag** — FE is set when a logic 0 is accepted as the stop bit. FE bit is set during the same cycle as the RDRF flag but does not get set in the case of an overrun. FE inhibits further data reception until it is cleared. Clear FE by reading SCI status register 1 (SCISR1) with FE set and then reading the SCI data register low (SCIDRL). 0 No framing error 1 Framing error
0 PF	**Parity Error Flag** — PF is set when the parity enable bit (PE) is set and the parity of the received data does not match the parity type bit (PT). PF bit is set during the same cycle as the RDRF flag but does not get set in the case of an overrun. Clear PF by reading SCI status register 1 (SCISR1), and then reading SCI data register low (SCIDRL). 0 No parity error 1 Parity error

15.3.2.8 SCI Status Register 2 (SCISR2)

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R	AMAP	0	0	TXPOL	RXPOL	BRK13	TXDIR	RAF
W		▨	▨					▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 15-11. SCI Status Register 2 (SCISR2)

Read: Anytime

Write: Anytime

Table 15-12. SCISR2 Field Descriptions

Field	Description
7 AMAP	**Alternative Map** — This bit controls which registers sharing the same address space are accessible. In the reset condition the SCI behaves as previous versions. Setting AMAP=1 allows the access to another set of control and status registers and hides the baud rate and SCI control Register 1. 0 The registers labelled SCIBDH (0x0000),SCIBDL (0x0001), SCICR1 (0x0002) are accessible 1 The registers labelled SCIASR1 (0x0000),SCIACR1 (0x0001), SCIACR2 (0x00002) are accessible
4 TXPOL	**Transmit Polarity** — This bit control the polarity of the transmitted data. In NRZ format, a one is represented by a mark and a zero is represented by a space for normal polarity, and the opposite for inverted polarity. In IrDA format, a zero is represented by short high pulse in the middle of a bit time remaining idle low for a one for normal polarity, and a zero is represented by short low pulse in the middle of a bit time remaining idle high for a one for inverted polarity. 0 Normal polarity 1 Inverted polarity
3 RXPOL	**Receive Polarity** — This bit control the polarity of the received data. In NRZ format, a one is represented by a mark and a zero is represented by a space for normal polarity, and the opposite for inverted polarity. In IrDA format, a zero is represented by short high pulse in the middle of a bit time remaining idle low for a one for normal polarity, and a zero is represented by short low pulse in the middle of a bit time remaining idle high for a one for inverted polarity. 0 Normal polarity 1 Inverted polarity
2 BRK13	**Break Transmit Character Length** — This bit determines whether the transmit break character is 10 or 11 bit respectively 13 or 14 bits long. The detection of a framing error is not affected by this bit. 0 Break character is 10 or 11 bit long 1 Break character is 13 or 14 bit long
1 TXDIR	**Transmitter Pin Data Direction in Single-Wire Mode** — This bit determines whether the TXD pin is going to be used as an input or output, in the single-wire mode of operation. This bit is only relevant in the single-wire mode of operation. 0 TXD pin to be used as an input in single-wire mode 1 TXD pin to be used as an output in single-wire mode
0 RAF	**Receiver Active Flag** — RAF is set when the receiver detects a logic 0 during the RT1 time period of the start bit search. RAF is cleared when the receiver detects an idle character. 0 No reception in progress 1 Reception in progress

15.3.2.9 SCI Data Registers (SCIDRH, SCIDRL)

Module Base + 0x0006

	7	6	5	4	3	2	1	0
R	R8	T8	0	0	0	0	0	0
W	▨		▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 15-12. SCI Data Registers (SCIDRH)

Module Base + 0x0007

	7	6	5	4	3	2	1	0
R	R7	R6	R5	R4	R3	R2	R1	R0
W	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0

Figure 15-13. SCI Data Registers (SCIDRL)

Read: Anytime; reading accesses SCI receive data register

Write: Anytime; writing accesses SCI transmit data register; writing to R8 has no effect

Table 15-13. SCIDRH and SCIDRL Field Descriptions

Field	Description
SCIDRH 7 R8	**Received Bit 8** — R8 is the ninth data bit received when the SCI is configured for 9-bit data format (M = 1).
SCIDRH 6 T8	**Transmit Bit 8** — T8 is the ninth data bit transmitted when the SCI is configured for 9-bit data format (M = 1).
SCIDRL 7:0 R[7:0] T[7:0]	**R7:R0** — Received bits seven through zero for 9-bit or 8-bit data formats **T7:T0** — Transmit bits seven through zero for 9-bit or 8-bit formats

NOTE

If the value of T8 is the same as in the previous transmission, T8 does not have to be rewritten.The same value is transmitted until T8 is rewritten

In 8-bit data format, only SCI data register low (SCIDRL) needs to be accessed.

When transmitting in 9-bit data format and using 8-bit write instructions, write first to SCI data register high (SCIDRH), then SCIDRL.

15.4 Functional Description

This section provides a complete functional description of the SCI block, detailing the operation of the design from the end user perspective in a number of subsections.

Figure 15-14 shows the structure of the SCI module. The SCI allows full duplex, asynchronous, serial communication between the CPU and remote devices, including other CPUs. The SCI transmitter and receiver operate independently, although they use the same baud rate generator. The CPU monitors the status of the SCI, writes the data to be transmitted, and processes received data.

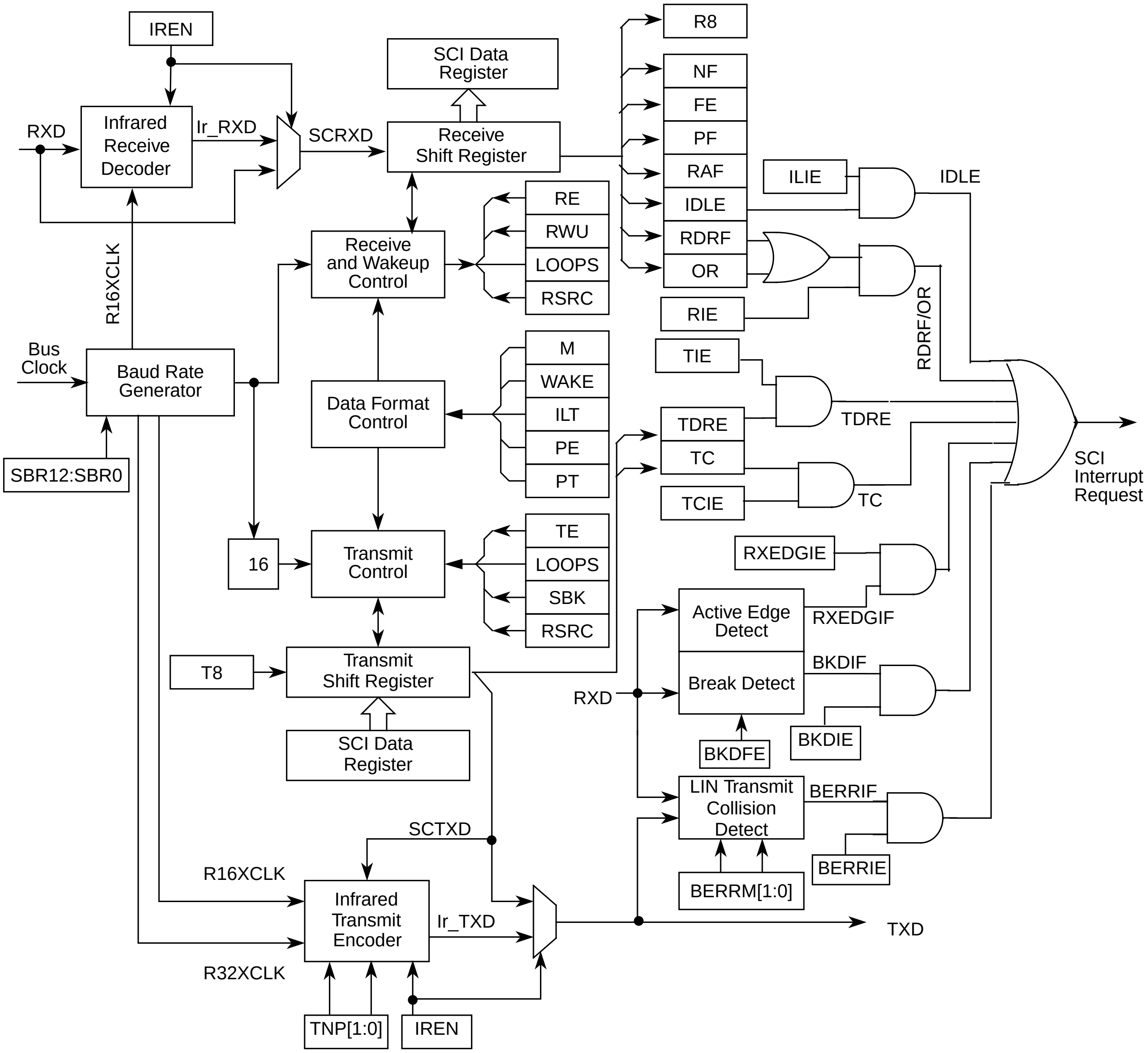


Figure 15-14. Detailed SCI Block Diagram

15.4.1 Infrared Interface Submodule

This module provides the capability of transmitting narrow pulses to an IR LED and receiving narrow pulses and transforming them to serial bits, which are sent to the SCI. The IrDA physical layer specification defines a half-duplex infrared communication link for exchange data. The full standard includes data rates up to 16 Mbits/s. This design covers only data rates between 2.4 Kbits/s and 115.2 Kbits/s.

The infrared submodule consists of two major blocks: the transmit encoder and the receive decoder. The SCI transmits serial bits of data which are encoded by the infrared submodule to transmit a narrow pulse for every zero bit. No pulse is transmitted for every one bit. When receiving data, the IR pulses should be detected using an IR photo diode and transformed to CMOS levels by the IR receive decoder (external from the MCU). The narrow pulses are then stretched by the infrared submodule to get back to a serial bit stream to be received by the SCI.The polarity of transmitted pulses and expected receive pulses can be inverted so that a direct connection can be made to external IrDA transceiver modules that uses active low pulses.

The infrared submodule receives its clock sources from the SCI. One of these two clocks are selected in the infrared submodule in order to generate either 3/16, 1/16, 1/32 or 1/4 narrow pulses during transmission. The infrared block receives two clock sources from the SCI, R16XCLK and R32XCLK, which are configured to generate the narrow pulse width during transmission. The R16XCLK and R32XCLK are internal clocks with frequencies 16 and 32 times the baud rate respectively. Both R16XCLK and R32XCLK clocks are used for transmitting data. The receive decoder uses only the R16XCLK clock.

15.4.1.1 Infrared Transmit Encoder

The infrared transmit encoder converts serial bits of data from transmit shift register to the TXD pin. A narrow pulse is transmitted for a zero bit and no pulse for a one bit. The narrow pulse is sent in the middle of the bit with a duration of 1/32, 1/16, 3/16 or 1/4 of a bit time. A narrow high pulse is transmitted for a zero bit when TXPOL is cleared, while a narrow low pulse is transmitted for a zero bit when TXPOL is set.

15.4.1.2 Infrared Receive Decoder

The infrared receive block converts data from the RXD pin to the receive shift register. A narrow pulse is expected for each zero received and no pulse is expected for each one received. A narrow high pulse is expected for a zero bit when RXPOL is cleared, while a narrow low pulse is expected for a zero bit when RXPOL is set. This receive decoder meets the edge jitter requirement as defined by the IrDA serial infrared physical layer specification.

15.4.2 LIN Support

This module provides some basic support for the LIN protocol. At first this is a break detect circuitry making it easier for the LIN software to distinguish a break character from an incoming data stream. As a further addition is supports a collision detection at the bit level as well as cancelling pending transmissions.

15.4.3 Data Format

The SCI uses the standard NRZ mark/space data format. When Infrared is enabled, the SCI uses RZI data format where zeroes are represented by light pulses and ones remain low. See Figure 15-15 below.

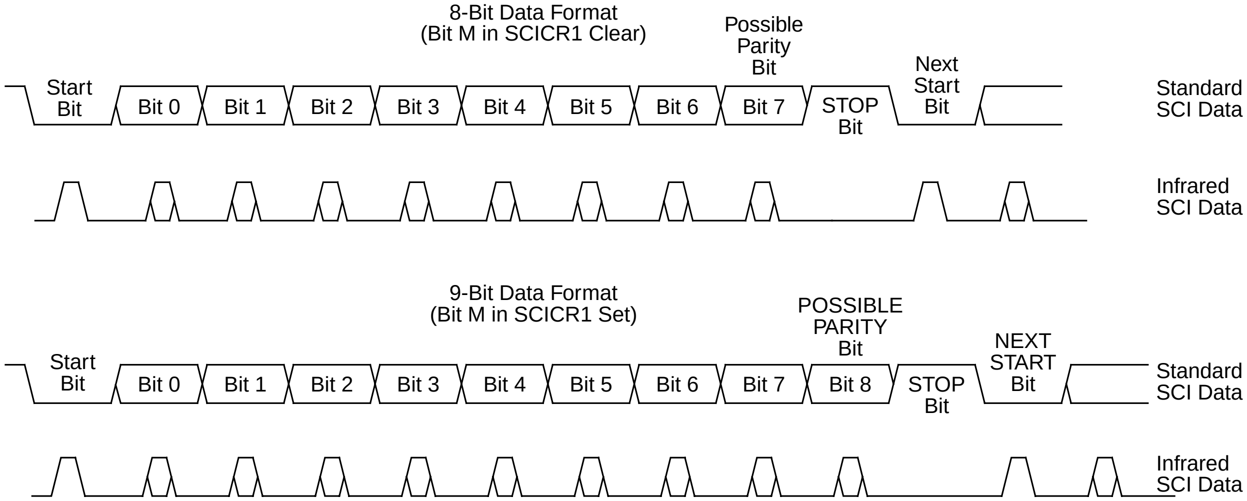


Figure 15-15. SCI Data Formats

Each data character is contained in a frame that includes a start bit, eight or nine data bits, and a stop bit. Clearing the M bit in SCI control register 1 configures the SCI for 8-bit data characters. A frame with eight data bits has a total of 10 bits. Setting the M bit configures the SCI for nine-bit data characters. A frame with nine data bits has a total of 11 bits.

Table 15-14. Example of 8-Bit Data Formats

Start Bit	Data Bits	Address Bits	Parity Bits	Stop Bit
1	8	0	0	1
1	7	0	1	1
1	7	1[1]	0	1

[1] The address bit identifies the frame as an address character. See Section 15.4.6.6, “Receiver Wakeup”.

When the SCI is configured for 9-bit data characters, the ninth data bit is the T8 bit in SCI data register high (SCIDRH). It remains unchanged after transmission and can be used repeatedly without rewriting it. A frame with nine data bits has a total of 11 bits.

Table 15-15. Example of 9-Bit Data Formats

Start Bit	Data Bits	Address Bits	Parity Bits	Stop Bit
1	9	0	0	1
1	8	0	1	1
1	8	1[1]	0	1

[1] The address bit identifies the frame as an address character. See Section 15.4.6.6, “Receiver Wakeup”.

15.4.4 Baud Rate Generation

A 13-bit modulus counter in the baud rate generator derives the baud rate for both the receiver and the transmitter. The value from 0 to 8191 written to the SBR12:SBR0 bits determines the bus clock divisor. The SBR bits are in the SCI baud rate registers (SCIBDH and SCIBDL). The baud rate clock is synchronized with the 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 one source of error:

- Integer division of the bus clock may not give the exact target frequency.

Table 15-16 lists some examples of achieving target baud rates with a bus clock frequency of 25 MHz.

When IREN = 0 then,

SCI baud rate = SCI bus clock / (16 * SCIBR[12:0])

Table 15-16. Baud Rates (Example: Bus Clock = 25 MHz)

Bits SBR[12:0]	Receiver Clock (Hz)	Transmitter Clock (Hz)	Target Baud Rate	Error (%)
41	609,756.1	38,109.8	38,400	.76
81	308,642.0	19,290.1	19,200	.47
163	153,374.2	9585.9	9,600	.16
326	76,687.1	4792.9	4,800	.15
651	38,402.5	2400.2	2,400	.01
1302	19,201.2	1200.1	1,200	.01
2604	9600.6	600.0	600	.00
5208	4800.0	300.0	300	.00

15.4.5 Transmitter

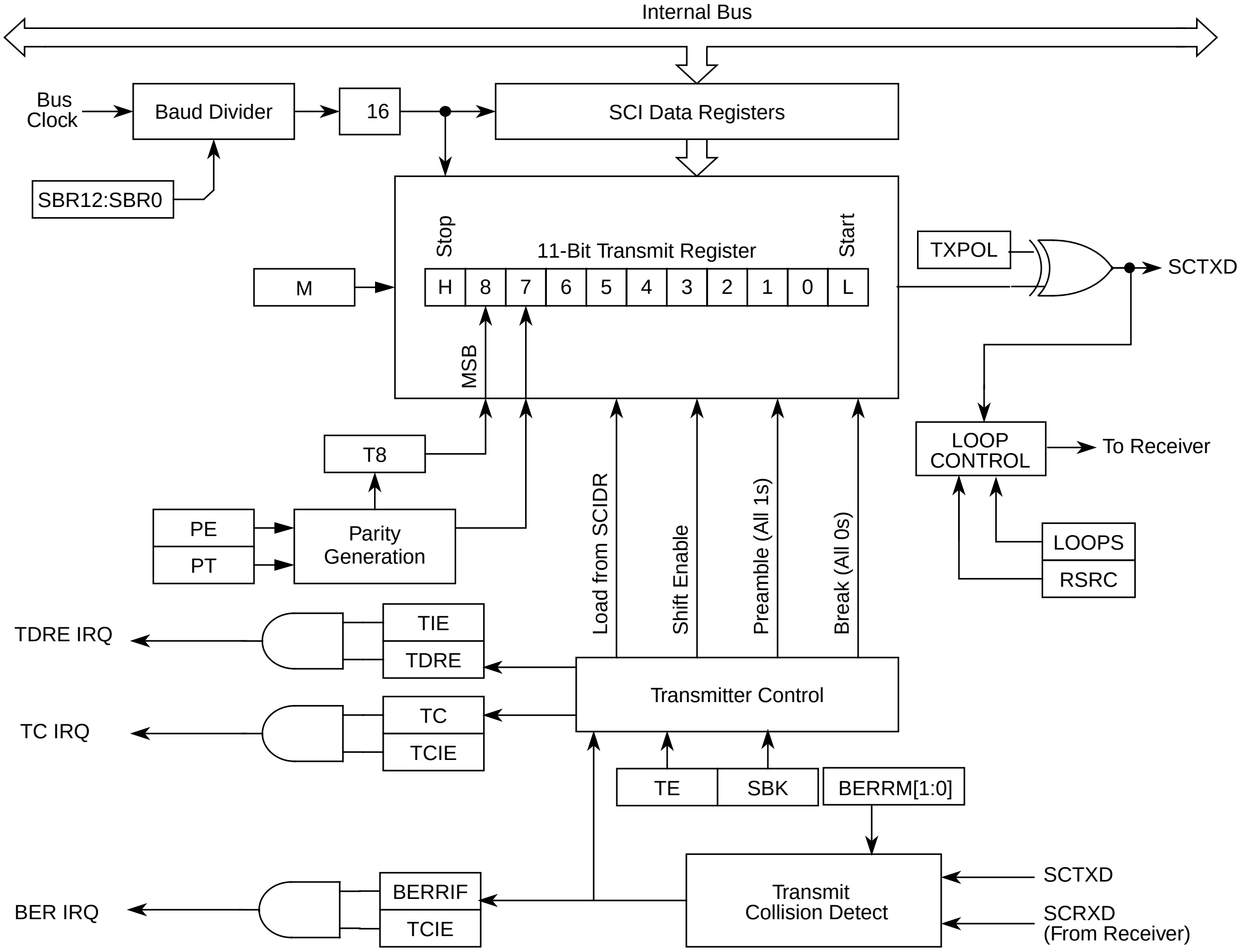


Figure 15-16. Transmitter Block Diagram

15.4.5.1 Transmitter Character Length

The SCI transmitter can accommodate either 8-bit or 9-bit data characters. The state of the M bit in SCI control register 1 (SCICR1) determines the length of data characters. When transmitting 9-bit data, bit T8 in SCI data register high (SCIDRH) is the ninth bit (bit 8).

15.4.5.2 Character Transmission

To transmit data, the MCU writes the data bits to the SCI data registers (SCIDRH/SCIDRL), which in turn are transferred to the transmitter shift register. The transmit shift register then shifts a frame out through the TXD pin, after it has prefaced them with a start bit and appended them with a stop bit. The SCI data registers (SCIDRH and SCIDRL) are the write-only buffers between the internal data bus and the transmit shift register.

The SCI also sets a flag, the transmit data register empty flag (TDRE), every time it transfers data from the buffer (SCIDRH/L) to the transmitter shift register.The transmit driver routine may respond to this flag by writing another byte to the Transmitter buffer (SCIDRH/SCIDRL), while the shift register is still shifting out the first byte.

To initiate an SCI transmission:

1. Configure the SCI:
   a) Select a baud rate. Write this value to the SCI baud registers (SCIBDH/L) to begin the baud rate generator. Remember that the baud rate generator is disabled when the baud rate is zero. Writing to the SCIBDH has no effect without also writing to SCIBDL.
   b) Write to SCICR1 to configure word length, parity, and other configuration bits (LOOPS,RSRC,M,WAKE,ILT,PE,PT).
   c) Enable the transmitter, interrupts, receive, and wake up as required, by writing to the SCICR2 register bits (TIE,TCIE,RIE,ILIE,TE,RE,RWU,SBK). A preamble or idle character will now be shifted out of the transmitter shift register.
2. Transmit Procedure for each byte:
   a) Poll the TDRE flag by reading the SCISR1 or responding to the TDRE interrupt. Keep in mind that the TDRE bit resets to one.
   b) If the TDRE flag is set, write the data to be transmitted to SCIDRH/L, where the ninth bit is written to the T8 bit in SCIDRH if the SCI is in 9-bit data format. A new transmission will not result until the TDRE flag has been cleared.
3. Repeat step 2 for each subsequent transmission.

NOTE

The TDRE flag is set when the shift register is loaded with the next data to be transmitted from SCIDRH/L, which happens, generally speaking, a little over half-way through the stop bit of the previous frame. Specifically, this transfer occurs 9/16ths of a bit time AFTER the start of the stop bit of the previous frame.

Writing the TE bit from 0 to a 1 automatically loads the transmit shift register with a preamble of 10 logic 1s (if M = 0) or 11 logic 1s (if M = 1). After the preamble shifts out, control logic transfers the data from the SCI data register into the transmit shift register. A logic 0 start bit automatically goes into the least significant bit position of the transmit shift register. A logic 1 stop bit goes into the most significant bit position.

Hardware supports odd or even parity. When parity is enabled, the most significant bit (MSB) of the data character is the parity bit.

The transmit data register empty flag, TDRE, in SCI status register 1 (SCISR1) becomes set when the SCI data register transfers a byte to the transmit shift register. The TDRE flag indicates that the SCI data register can accept new data from the internal data bus. If the transmit interrupt enable bit, TIE, in SCI control register 2 (SCICR2) is also set, the TDRE flag generates a transmitter interrupt request.

When the transmit shift register is not transmitting a frame, the TXD pin goes to the idle condition, logic 1. If at any time software clears the TE bit in SCI control register 2 (SCICR2), the transmitter enable signal goes low and the transmit signal goes idle.

If software clears TE while a transmission is in progress (TC = 0), the frame in the transmit shift register continues to shift out. To avoid accidentally cutting off the last frame in a message, always wait for TDRE to go high after the last frame before clearing TE.

To separate messages with preambles with minimum idle line time, use this sequence between messages:

1. Write the last byte of the first message to SCIDRH/L.
2. Wait for the TDRE flag to go high, indicating the transfer of the last frame to the transmit shift register.
3. Queue a preamble by clearing and then setting the TE bit.
4. Write the first byte of the second message to SCIDRH/L.

15.4.5.3 Break Characters

Writing a logic 1 to the send break bit, SBK, in SCI control register 2 (SCICR2) loads the transmit shift register with a break character. A break character contains all logic 0s and has no start, stop, or parity bit. Break character length depends on the M bit in SCI control register 1 (SCICR1). As long as SBK is at logic 1, transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the shift register finishes transmitting the last break character and then transmits at least one logic 1. The automatic logic 1 at the end of a break character guarantees the recognition of the start bit of the next frame.

The SCI recognizes a break character when there are 10 or 11(M = 0 or M = 1) consecutive zero received. Depending if the break detect feature is enabled or not receiving a break character has these effects on SCI registers.

If the break detect feature is disabled (BKDFE = 0):

- Sets the framing error flag, FE
- Sets the receive data register full flag, RDRF
- Clears the SCI data registers (SCIDRH/L)
- May set the overrun flag, OR, noise flag, NF, parity error flag, PE, or the receiver active flag, RAF (see 3.4.4 and 3.4.5 SCI Status Register 1 and 2)

If the break detect feature is enabled (BKDFE = 1) there are two scenarios[1]

The break is detected right from a start bit or is detected during a byte reception.

- Sets the break detect interrupt flag, BKDIF
- Does not change the data register full flag, RDRF or overrun flag OR
- Does not change the framing error flag FE, parity error flag PE.
- Does not clear the SCI data registers (SCIDRH/L)
- May set noise flag NF, or receiver active flag RAF.

---

[1] A Break character in this context are either 10 or 11 consecutive zero received bits

Figure 15-17 shows two cases of break detect. In trace RXD_1 the break symbol starts with the start bit, while in RXD_2 the break starts in the middle of a transmission. If BRKDFE = 1, in RXD_1 case there will be no byte transferred to the receive buffer and the RDRF flag will not be modified. Also no framing error or parity error will be flagged from this transfer. In RXD_2 case, however the break signal starts later during the transmission. At the expected stop bit position the byte received so far will be transferred to the receive buffer, the receive data register full flag will be set, a framing error and if enabled and appropriate a parity error will be set. Once the break is detected the BRKDIF flag will be set.

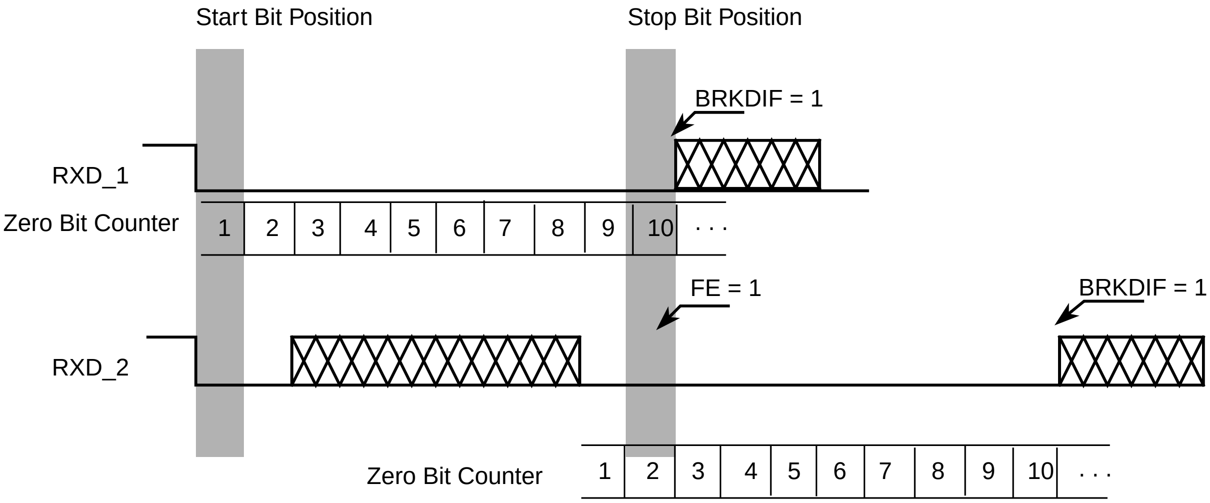


Figure 15-17. Break Detection if BRKDFE = 1 (M = 0)

15.4.5.4 Idle Characters

An idle character (or preamble) contains all logic 1s and has no start, stop, or parity bit. Idle character length depends on the M bit in SCI control register 1 (SCICR1). The preamble is a synchronizing idle character that begins the first transmission initiated after writing the TE bit from 0 to 1.

If the TE bit is cleared during a transmission, the TXD pin becomes idle after completion of the transmission in progress. Clearing and then setting the TE bit during a transmission queues an idle character to be sent after the frame currently being transmitted.

NOTE

When queueing an idle character, return the TE bit to logic 1 before the stop bit of the current frame shifts out through the TXD pin. Setting TE after the stop bit appears on TXD causes data previously written to the SCI data register to be lost. Toggle the TE bit for a queued idle character while the TDRE flag is set and immediately before writing the next byte to the SCI data register.

If the TE bit is clear and the transmission is complete, the SCI is not the master of the TXD pin

15.4.5.5 LIN Transmit Collision Detection

This module allows to check for collisions on the LIN bus.

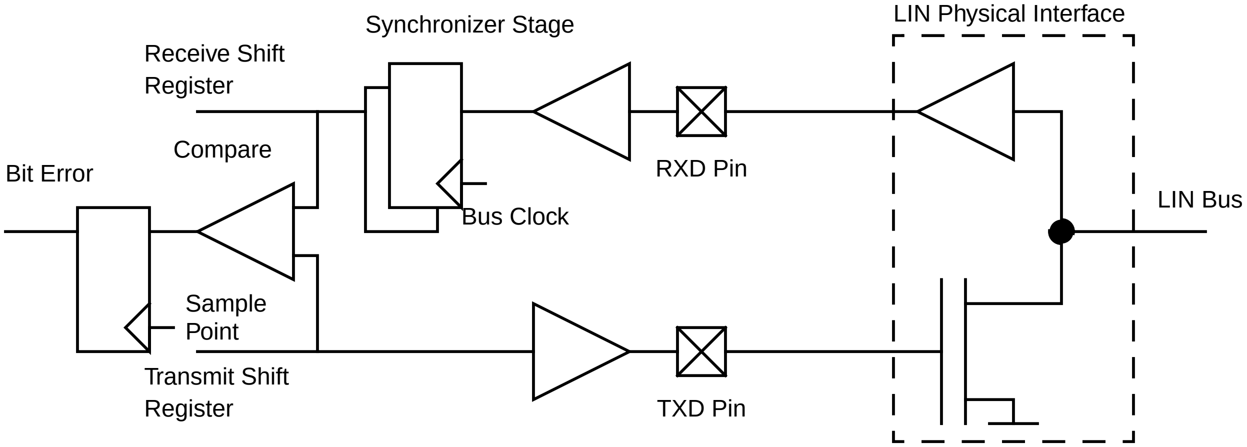


Figure 15-18. Collision Detect Principle

If the bit error circuit is enabled (BERRM[1:0] = 0:1 or = 1:0]), the error detect circuit will compare the transmitted and the received data stream at a point in time and flag any mismatch. The timing checks run when transmitter is active (not idle). As soon as a mismatch between the transmitted data and the received data is detected the following happens:

- The next bit transmitted will have a high level (TXPOL = 0) or low level (TXPOL = 1)
- The transmission is aborted and the byte in transmit buffer is discarded.
- the transmit data register empty and the transmission complete flag will be set
- The bit error interrupt flag, BERRIF, will be set.
- No further transmissions will take place until the BERRIF is cleared.

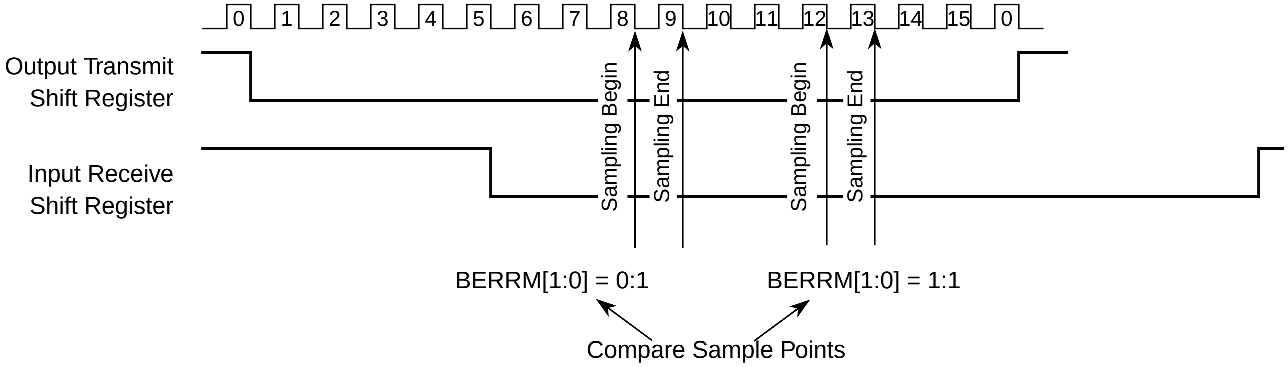


Figure 15-19. Timing Diagram Bit Error Detection

If the bit error detect feature is disabled, the bit error interrupt flag is cleared.

NOTE

The RXPOL and TXPOL bit should be set the same when transmission collision detect feature is enabled, otherwise the bit error interrupt flag may be set incorrectly.

15.4.6 Receiver

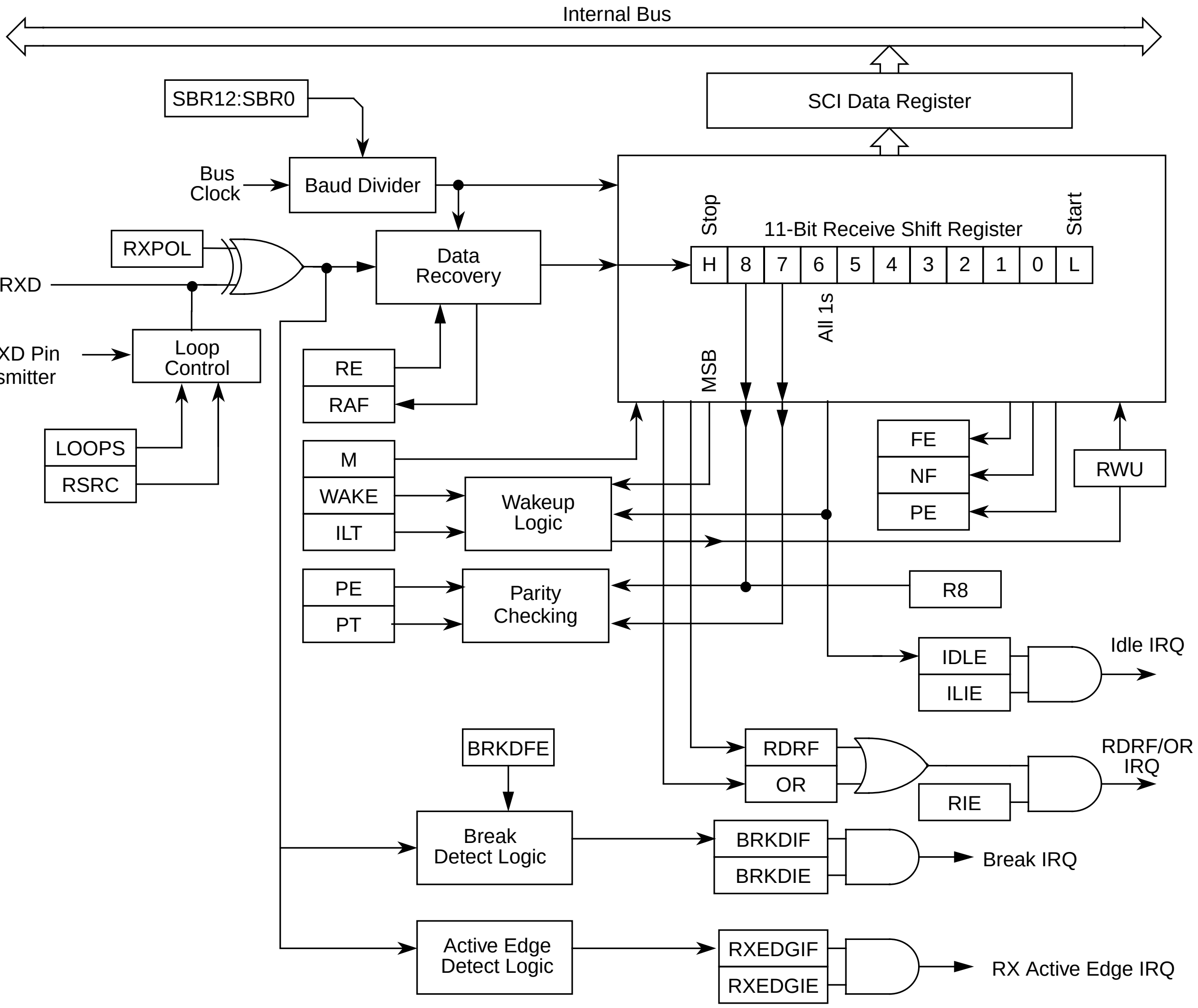


Figure 15-20. SCI Receiver Block Diagram

15.4.6.1 Receiver Character Length

The SCI receiver can accommodate either 8-bit or 9-bit data characters. The state of the M bit in SCI control register 1 (SCICR1) determines the length of data characters. When receiving 9-bit data, bit R8 in SCI data register high (SCIDRH) is the ninth bit (bit 8).

15.4.6.2 Character Reception

During an SCI reception, the receive shift register shifts a frame in from the RXD pin. The SCI data register 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 transfers to the SCI data register. The receive data register full flag, RDRF, in SCI status register 1 (SCISR1) becomes set, indicating that the received byte can be read. If the receive interrupt enable bit, RIE, in SCI control register 2 (SCICR2) is also set, the RDRF flag generates an RDRF interrupt request.

15.4.6.3 Data Sampling

The RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock (see Figure 15-21) is re-synchronized:

- After every start bit
- After the receiver detects a data bit change from logic 1 to logic 0 (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid logic 1 and the majority of the next RT8, RT9, and RT10 samples returns a valid logic 0)

To locate the start bit, data recovery logic does an asynchronous search for a logic 0 preceded by three logic 1s.When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

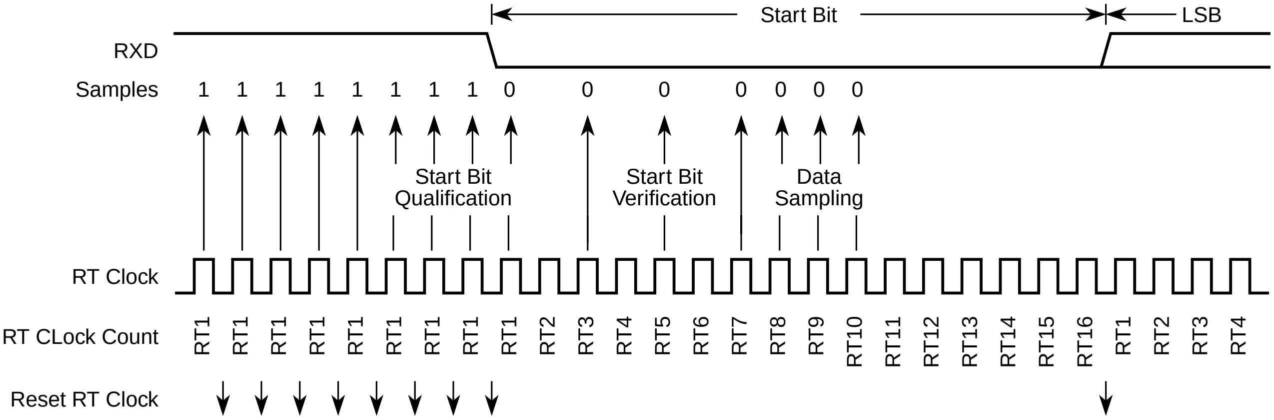


Figure 15-21. Receiver Data Sampling

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. Figure 15-17 summarizes the results of the start bit verification samples.

Table 15-17. 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. Table 15-18 summarizes the results of the data bit samples.

Table 15-18. Data Bit Recovery

RT8, RT9, and RT10 Samples	Data Bit Determination	Noise Flag
000	0	0
001	0	1
010	0	1
011	1	1
100	0	1
101	1	1
110	1	1
111	1	0

NOTE

The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic 1s following a successful start bit verification, the noise flag (NF) is set and the receiver assumes that the bit is a start bit (logic 0).

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 15-19 summarizes the results of the stop bit samples.

Table 15-19. 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 Figure 15-22 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.

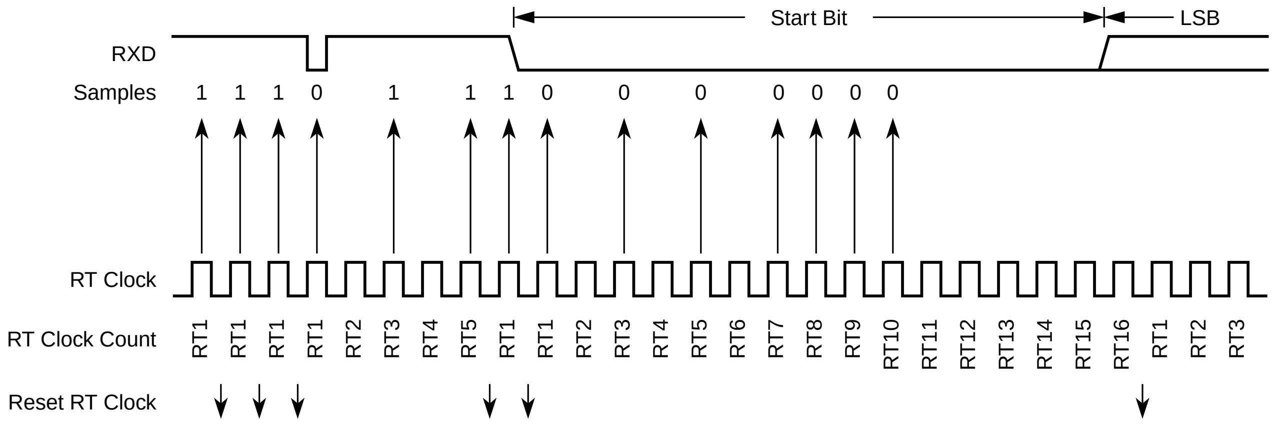


Figure 15-22. Start Bit Search Example 1

In Figure 15-23, 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.

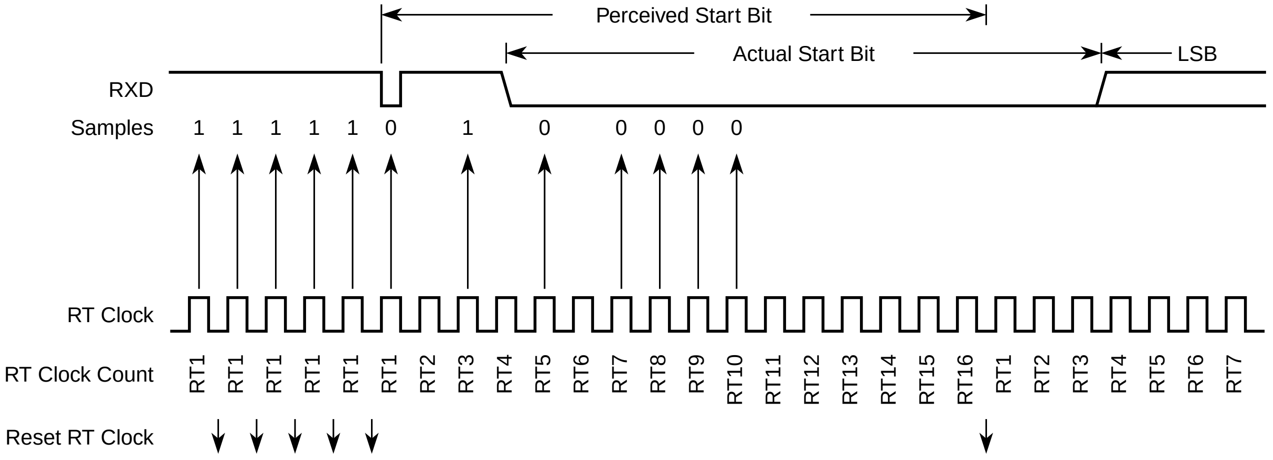


Figure 15-23. Start Bit Search Example 2

In Figure 15-24, 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.

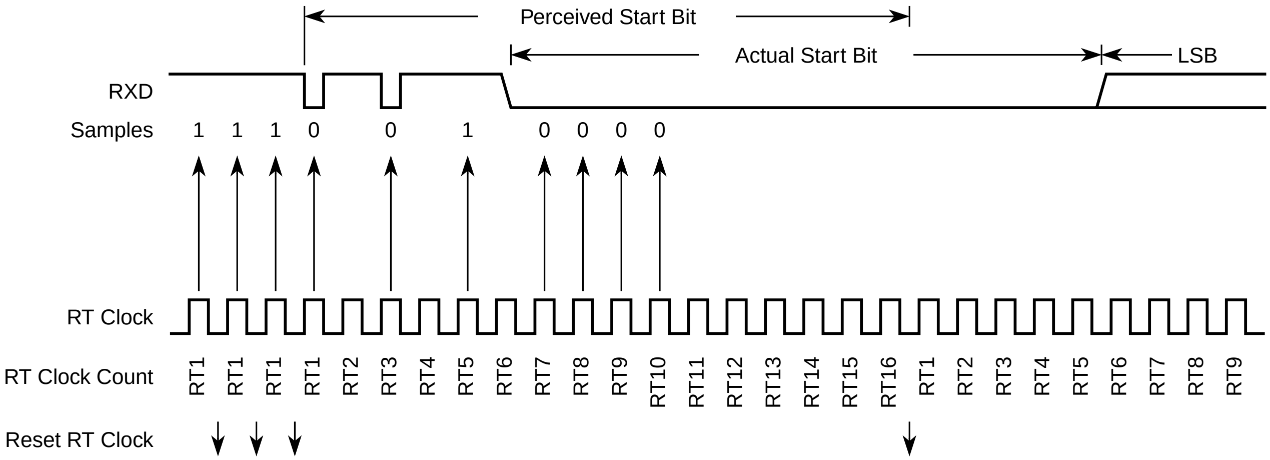


Figure 15-24. Start Bit Search Example 3

Figure 15-25 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.

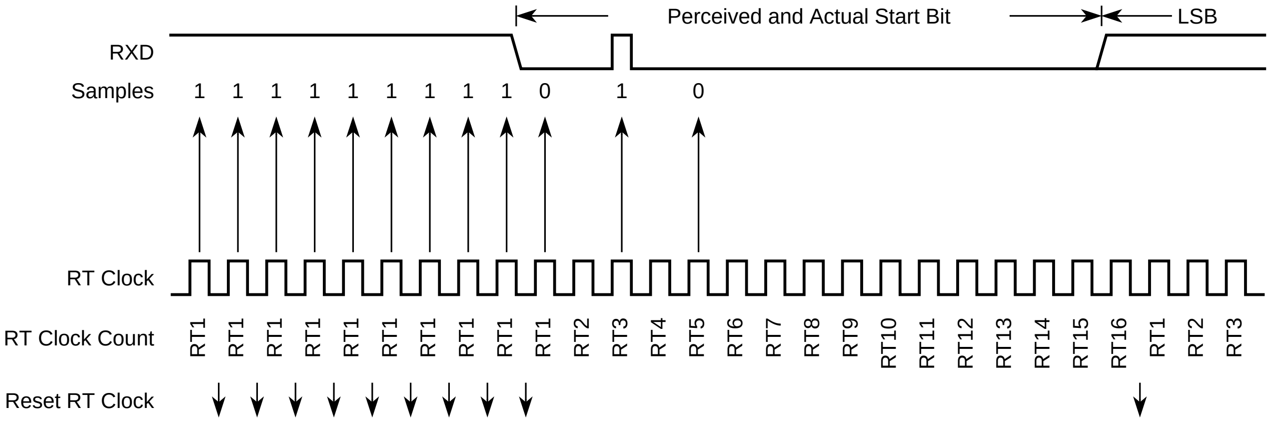


Figure 15-25. Start Bit Search Example 4

Figure 15-26 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.

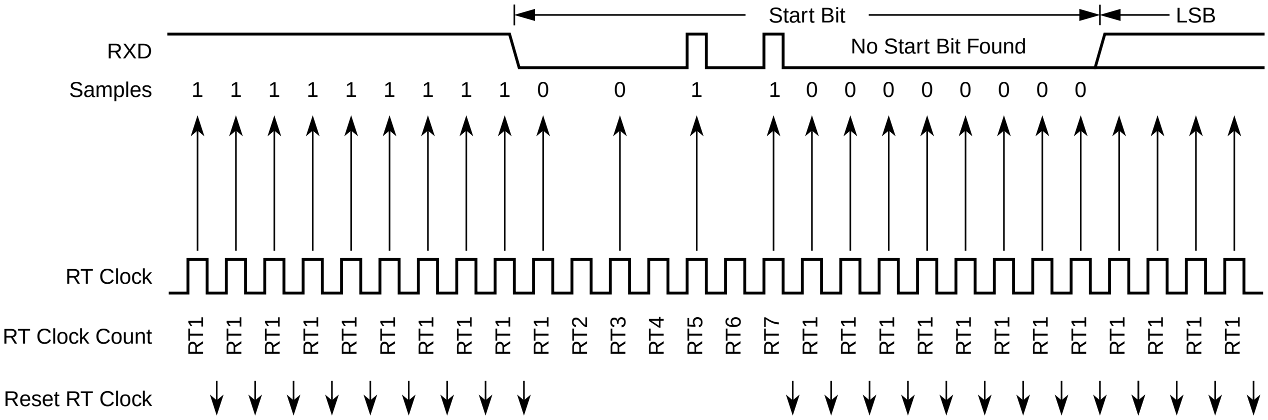


Figure 15-26. Start Bit Search Example 5

In Figure 15-27, 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.

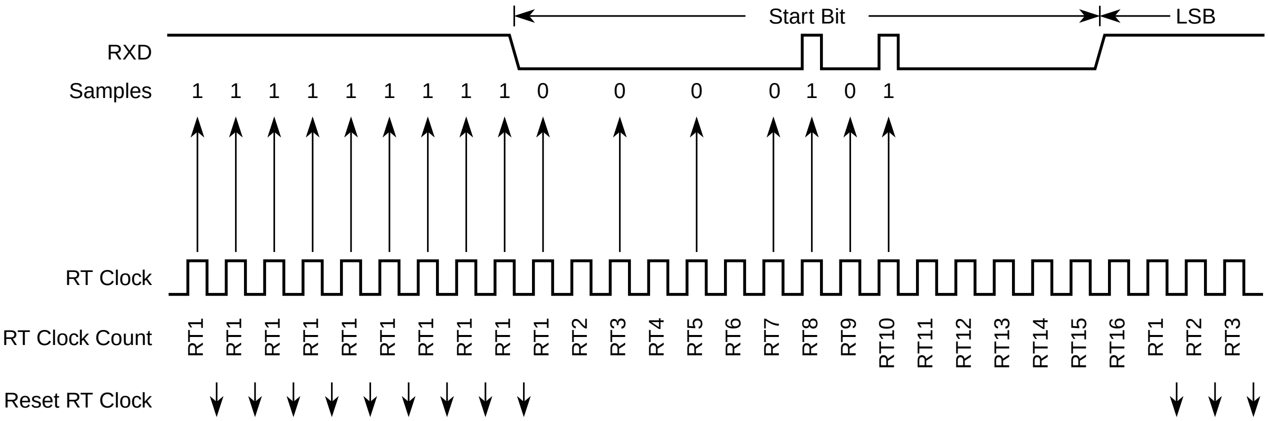


Figure 15-27. Start Bit Search Example 6

15.4.6.4 Framing Errors

If the data recovery logic does not detect a logic 1 where the stop bit should be in an incoming frame, it sets the framing error flag, FE, in SCI status register 1 (SCISR1). A break character also sets the FE flag because a break character has no stop bit. The FE flag is set at the same time that the RDRF flag is set.

15.4.6.5 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 (RT8, RT9, and RT10) to fall outside the actual stop bit. A noise error will occur if the RT8, RT9, and RT10 samples are not all the same logical values. A framing error will occur if the receiver clock is misaligned in such a way that the majority of the RT8, RT9, and RT10 stop bit samples are a logic zero.

As the receiver samples an incoming frame, it re-synchronizes the RT clock on any valid falling edge within the frame. Re synchronization within frames will correct a misalignment between transmitter bit times and receiver bit times.

15.4.6.5.1 Slow Data Tolerance

Figure 15-28 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.

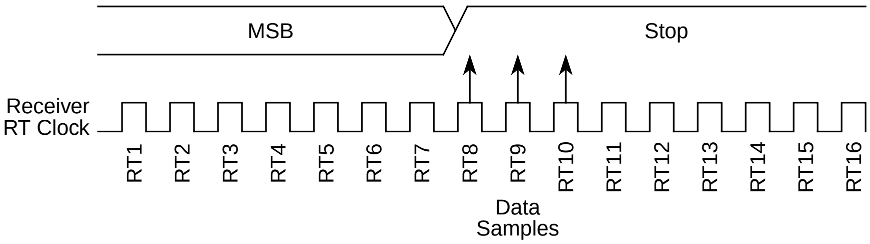


Figure 15-28. Slow Data

Let’s take RTr as receiver RT clock and RTt as transmitter RT clock.

For an 8-bit data character, it takes the receiver 9 bit times x 16 RTr cycles +7 RTr cycles = 151 RTr cycles to start data sampling of the stop bit.

With the misaligned character shown in Figure 15-28, the receiver counts 151 RTr cycles at the point when the count of the transmitting device is 9 bit times x 16 RTt cycles = 144 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 8-bit data character with no errors is:

$$((151 - 144) / 151) \times 100 = 4.63\%$$

For a 9-bit data character, it takes the receiver 10 bit times x 16 RTr cycles + 7 RTr cycles = 167 RTr cycles to start data sampling of the stop bit.

With the misaligned character shown in Figure 15-28, the receiver counts 167 RTr cycles at the point when the count of the transmitting device is 10 bit times x 16 RTt cycles = 160 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

$$((167 - 160) / 167) \text{ X } 100 = 4.19\%$$

15.4.6.5.2 Fast Data Tolerance

Figure 15-29 shows how much a fast received frame can be misaligned. The fast stop bit ends at RT10 instead of RT16 but is still sampled at RT8, RT9, and RT10.

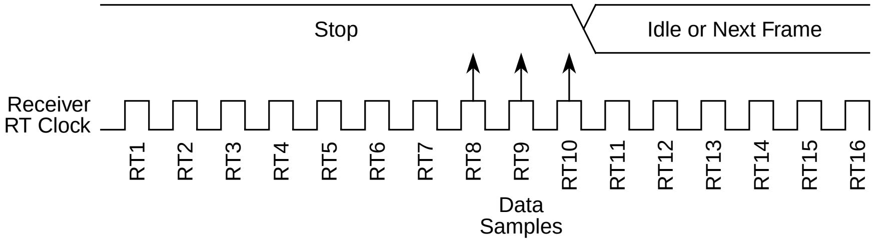


Figure 15-29. Fast Data

For an 8-bit data character, it takes the receiver 9 bit times x 16 RTr cycles + 10 RTr cycles = 154 RTr cycles to finish data sampling of the stop bit.

With the misaligned character shown in Figure 15-29, the receiver counts 154 RTr cycles at the point when the count of the transmitting device is 10 bit times x 16 RTt cycles = 160 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is:

((160 – 154) / 160) x 100 = 3.75%

For a 9-bit data character, it takes the receiver 10 bit times x 16 RTr cycles + 10 RTr cycles = 170 RTr cycles to finish data sampling of the stop bit.

With the misaligned character shown in Figure 15-29, the receiver counts 170 RTr cycles at the point when the count of the transmitting device is 11 bit times x 16 RTt cycles = 176 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

((176 – 170) /176) x 100 = 3.40%

15.4.6.6 Receiver Wakeup

To enable the SCI to 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, RWU, in SCI control register 2 (SCICR2) puts the receiver into standby state during which receiver interrupts are disabled.The SCI will still load the receive data into the SCIDRH/L registers, but it will not set the RDRF flag.

The transmitting device can address messages to selected receivers by including addressing information in the initial frame or frames of each message.

The WAKE bit in SCI control register 1 (SCICR1) determines how the SCI is brought out of the standby state to process an incoming message. The WAKE bit enables either idle line wakeup or address mark wakeup.

15.4.6.6.1 Idle Input line Wakeup (WAKE = 0)

In this wakeup method, an idle condition on the RXD pin clears the RWU bit and wakes up the SCI. The initial frame or frames of every message contain addressing information. All receivers evaluate the addressing information, and receivers for which the message is addressed process the frames that follow. Any receiver for which a message is not addressed can set its RWU bit and return to the standby state. The RWU bit remains set and the receiver remains on standby until another idle character appears on the RXD pin.

Idle line wakeup requires that messages be separated by at least one idle character and that no message contains idle characters.

The idle character that wakes a receiver does not set the receiver idle bit, IDLE, or the receive data register full flag, RDRF.

The idle line type bit, ILT, determines whether the receiver begins counting logic 1s as idle character bits after the start bit or after the stop bit. ILT is in SCI control register 1 (SCICR1).

15.4.6.6.2 Address Mark Wakeup (WAKE = 1)

In this wakeup method, a logic 1 in the most significant bit (MSB) position of a frame clears the RWU bit and wakes up the SCI. The logic 1 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. The RWU bit remains set and the receiver remains on standby until another address frame appears on the RXD pin.

The logic 1 MSB of an address frame clears the receiver’s RWU bit before the stop bit is received and sets the RDRF flag.

Address mark wakeup allows messages to contain idle characters but requires that the MSB be reserved for use in address frames.

NOTE

With the WAKE bit clear, setting the RWU bit after the RXD pin has been idle can cause the receiver to wake up immediately.

15.4.7 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. The SCI uses the TXD pin for both receiving and transmitting.

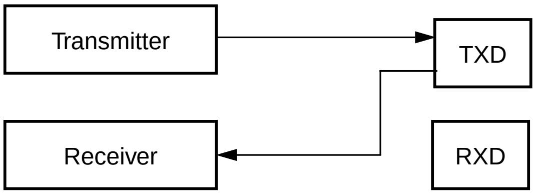


Figure 15-30. Single-Wire Operation (LOOPS = 1, RSRC = 1)

Enable single-wire operation by setting the LOOPS bit and the receiver source bit, RSRC, in SCI control register 1 (SCICR1). Setting the LOOPS bit disables the path from the RXD pin to the receiver. Setting the RSRC bit connects the TXD pin to the receiver. Both the transmitter and receiver must be enabled (TE = 1 and RE = 1).The TXDIR bit (SCISR2[1]) determines whether the TXD pin is going to be used as an input (TXDIR = 0) or an output (TXDIR = 1) in this mode of operation.

NOTE

In single-wire operation data from the TXD pin is inverted if RXPOL is set.

15.4.8 Loop Operation

In loop operation the transmitter output goes to the receiver input. The RXD pin is disconnected from the SCI.

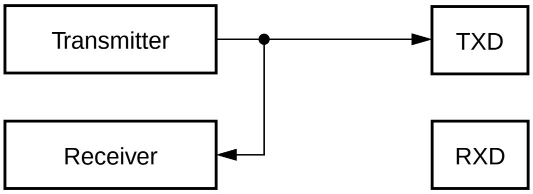


Figure 15-31. Loop Operation (LOOPS = 1, RSRC = 0)

Enable loop operation by setting the LOOPS bit and clearing the RSRC bit in SCI control register 1 (SCICR1). Setting the LOOPS bit disables the path from the RXD pin to the receiver. Clearing the RSRC bit connects the transmitter output to the receiver input. Both the transmitter and receiver must be enabled (TE = 1 and RE = 1).

NOTE

In loop operation data from the transmitter is not recognized by the receiver if RXPOL and TXPOL are not the same.

15.5 Initialization/Application Information

15.5.1 Reset Initialization

See Section 15.3.2, “Register Descriptions”.

15.5.2 Modes of Operation

15.5.2.1 Run Mode

Normal mode of operation.

To initialize a SCI transmission, see Section 15.4.5.2, “Character Transmission”.

15.5.2.2 Wait Mode

SCI operation in wait mode depends on the state of the SCISWAI bit in the SCI control register 1 (SCICR1).

- If SCISWAI is clear, the SCI operates normally when the CPU is in wait mode.
- If SCISWAI is set, SCI clock generation ceases and the SCI module enters a power-conservation state when the CPU is in wait mode. Setting SCISWAI does not affect the state of the receiver enable bit, RE, or the transmitter enable bit, TE.

  If SCISWAI 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 CPU out of wait mode. Exiting wait mode by reset aborts any transmission or reception in progress and resets the SCI.

15.5.2.3 Stop Mode

The SCI is inactive during stop mode for reduced power consumption. The STOP instruction does not affect the SCI register states, but the SCI bus clock will be disabled. The SCI operation resumes from where it left off after an external interrupt brings the CPU out of stop mode. Exiting stop mode by reset aborts any transmission or reception in progress and resets the SCI.

The receive input active edge detect circuit is still active in stop mode. An active edge on the receive input can be used to bring the CPU out of stop mode.

15.5.3 Interrupt Operation

This section describes the interrupt originated by the SCI block.The MCU must service the interrupt requests. Table 15-20 lists the eight interrupt sources of the SCI.

Table 15-20. SCI Interrupt Sources

Interrupt	Source	Local Enable	Description
TDRE	SCISR1[7]	TIE	Active high level. Indicates that a byte was transferred from SCIDRH/L to the transmit shift register.
TC	SCISR1[6]	TCIE	Active high level. Indicates that a transmit is complete.
RDRF	SCISR1[5]	RIE	Active high level. The RDRF interrupt indicates that received data is available in the SCI data register.
OR	SCISR1[3]		Active high level. This interrupt indicates that an overrun condition has occurred.
IDLE	SCISR1[4]	ILIE	Active high level. Indicates that receiver input has become idle.
RXEDGIF	SCIASR1[7]	RXEDGIE	Active high level. Indicates that an active edge (falling for RXPOL = 0, rising for RXPOL = 1) was detected.
BERRIF	SCIASR1[1]	BERRIE	Active high level. Indicates that a mismatch between transmitted and received data in a single wire application has happened.
BKDIF	SCIASR1[0]	BRKDIE	Active high level. Indicates that a break character has been received.

15.5.3.1 Description of Interrupt Operation

The SCI only originates interrupt requests. The following is a description of how the SCI makes a request and how the MCU should acknowledge that request. The interrupt vector offset and interrupt number are chip dependent. The SCI only has a single interrupt line (SCI Interrupt Signal, active high operation) and all the following interrupts, when generated, are ORed together and issued through that port.

15.5.3.1.1 TDRE Description

The TDRE interrupt is set high by the SCI when the transmit shift register receives a byte from the SCI data register. A TDRE interrupt indicates that the transmit data register (SCIDRH/L) is empty and that a new byte can be written to the SCIDRH/L for transmission.Clear TDRE by reading SCI status register 1 with TDRE set and then writing to SCI data register low (SCIDRL).

15.5.3.1.2 TC Description

The TC interrupt is set by the SCI when a transmission has been completed. Transmission is completed when all bits including the stop bit (if transmitted) have been shifted out and no data is queued to be transmitted. No stop bit is transmitted when sending a break character and the TC flag is set (providing there is no more data queued for transmission) when the break character has been shifted out. A TC interrupt indicates that there is no transmission in progress. TC is set high when the TDRE flag is set and no data, preamble, or break character is being transmitted. When TC is set, the TXD pin becomes idle (logic 1). Clear TC by reading SCI status register 1 (SCISR1) with TC set and then writing to SCI data register low (SCIDRL).TC is cleared automatically when data, preamble, or break is queued and ready to be sent.

15.5.3.1.3 RDRF Description

The RDRF interrupt is set when the data in the receive shift register transfers to the SCI data register. A RDRF interrupt indicates that the received data has been transferred to the SCI data register and that the byte can now be read by the MCU. The RDRF interrupt is cleared by reading the SCI status register one (SCISR1) and then reading SCI data register low (SCIDRL).

15.5.3.1.4 OR Description

The OR interrupt is set when software fails to read the SCI data register before the receive shift register receives the next frame. The newly acquired data in the shift register will be lost in this case, but the data already in the SCI data registers is not affected. The OR interrupt is cleared by reading the SCI status register one (SCISR1) and then reading SCI data register low (SCIDRL).

15.5.3.1.5 IDLE Description

The IDLE interrupt is set when 10 consecutive logic 1s (if M = 0) or 11 consecutive logic 1s (if M = 1) appear on the receiver input. Once the IDLE is cleared, a valid frame must again set the RDRF flag before an idle condition can set the IDLE flag. Clear IDLE by reading SCI status register 1 (SCISR1) with IDLE set and then reading SCI data register low (SCIDRL).

15.5.3.1.6 RXEDGIF Description

The RXEDGIF interrupt is set when an active edge (falling if RXPOL = 0, rising if RXPOL = 1) on the RXD pin is detected. Clear RXEDGIF by writing a “1” to the SCIASR1 SCI alternative status register 1.

15.5.3.1.7 BERRIF Description

The BERRIF interrupt is set when a mismatch between the transmitted and the received data in a single wire application like LIN was detected. Clear BERRIF by writing a “1” to the SCIASR1 SCI alternative status register 1. This flag is also cleared if the bit error detect feature is disabled.

15.5.3.1.8 BKDIF Description

The BKDIF interrupt is set when a break signal was received. Clear BKDIF by writing a “1” to the SCIASR1 SCI alternative status register 1. This flag is also cleared if break detect feature is disabled.

15.5.4 Recovery from Wait Mode

The SCI interrupt request can be used to bring the CPU out of wait mode.

15.5.5 Recovery from Stop Mode

An active edge on the receive input can be used to bring the CPU out of stop mode.

Chapter 16
Serial Peripheral Interface (S12SPIV4)

16.1 Introduction

The SPI module allows a duplex, synchronous, serial communication between the MCU and peripheral devices. Software can poll the SPI status flags or the SPI operation can be interrupt driven.

16.1.1 Glossary of Terms

SPI	Serial Peripheral Interface
$\overline{SS}$	Slave Select
SCK	Serial Clock
MOSI	Master Output, Slave Input
MISO	Master Input, Slave Output
MOMI	Master Output, Master Input
SISO	Slave Input, Slave Output

16.1.2 Features

The SPI includes these distinctive features:

- Master mode and slave mode
- Bidirectional mode
- Slave select output
- Mode fault error flag with CPU interrupt capability
- Double-buffered data register
- Serial clock with programmable polarity and phase
- Control of SPI operation during wait mode

16.1.3 Modes of Operation

The SPI functions in three modes: run, wait, and stop.

- Run mode

  This is the basic mode of operation.
- Wait mode

  SPI operation in wait mode is a configurable low power mode, controlled by the SPISWAI bit located in the SPICR2 register. In wait mode, if the SPISWAI bit is clear, the SPI operates like in run mode. If the SPISWAI bit is set, the SPI goes into a power conservative state, with the SPI clock generation turned off. If the SPI is configured as a master, any transmission in progress stops, but is resumed after CPU goes into run mode. If the SPI is configured as a slave, reception and transmission of a byte continues, so that the slave stays synchronized to the master.
- Stop mode

  The SPI is inactive in stop mode for reduced power consumption. If the SPI is configured as a master, any transmission in progress stops, but is resumed after CPU goes into run mode. If the SPI is configured as a slave, reception and transmission of a byte continues, so that the slave stays synchronized to the master.

This is a high level description only, detailed descriptions of operating modes are contained in Section 16.4.7, “Low Power Mode Options”.

16.1.4 Block Diagram

Figure 16-1 gives an overview on the SPI architecture. The main parts of the SPI are status, control and data registers, shifter logic, baud rate generator, master/slave control logic, and port control logic.

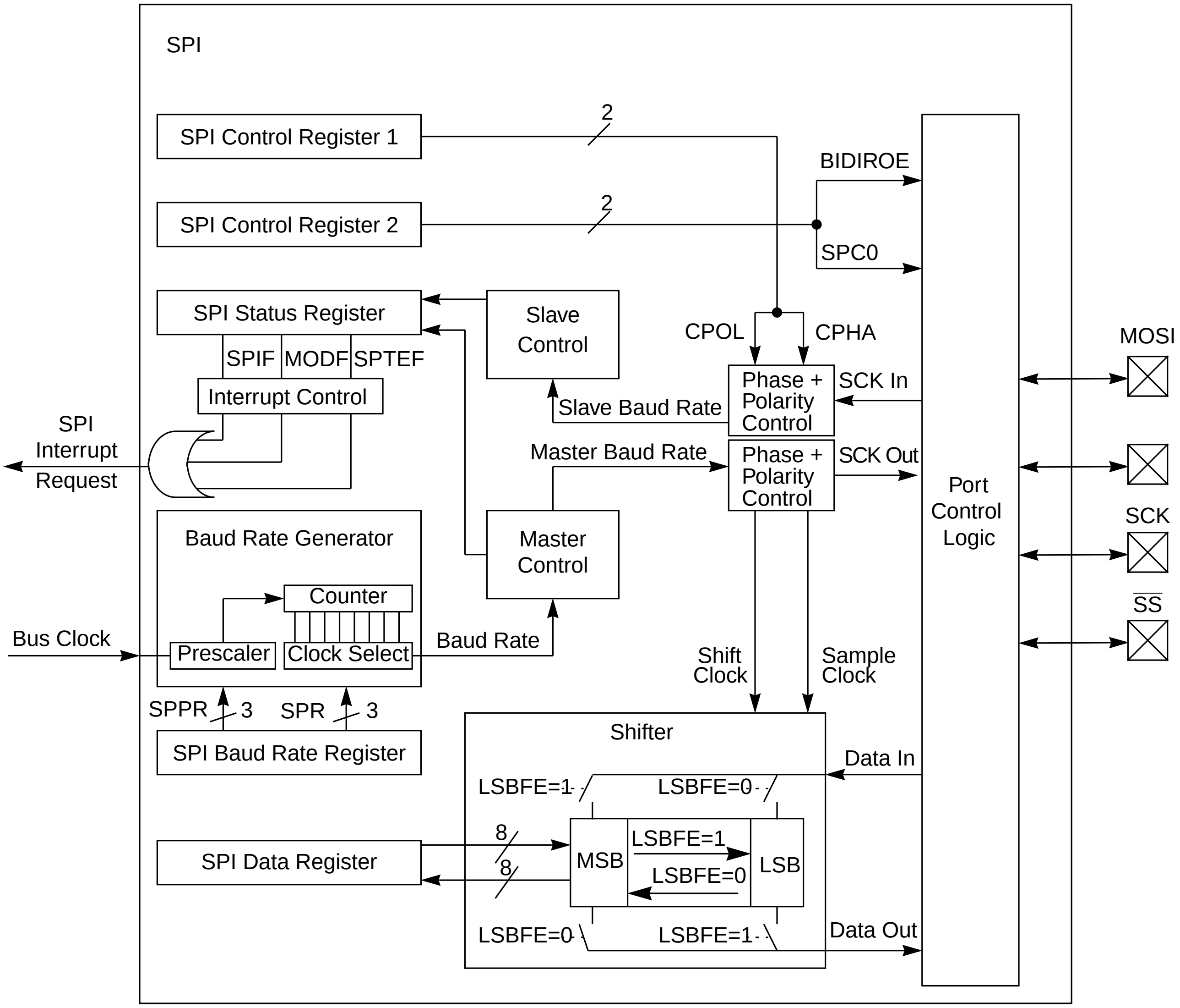


Figure 16-1. SPI Block Diagram

16.2 External Signal Description

This section lists the name and description of all ports including inputs and outputs that do, or may, connect off chip. The SPI module has a total of four external pins.

16.2.1 MOSI — Master Out/Slave In Pin

This pin is used to transmit data out of the SPI module when it is configured as a master and receive data when it is configured as slave.

16.2.2 MISO — Master In/Slave Out Pin

This pin is used to transmit data out of the SPI module when it is configured as a slave and receive data when it is configured as master.

16.2.3 $\overline{\text{SS}}$ — Slave Select Pin

This pin is used to output the select signal from the SPI module to another peripheral with which a data transfer is to take place when it is configured as a master and it is used as an input to receive the slave select signal when the SPI is configured as slave.

16.2.4 SCK — Serial Clock Pin

In master mode, this is the synchronous output clock. In slave mode, this is the synchronous input clock.

16.3 Memory Map and Register Definition

This section provides a detailed description of address space and registers used by the SPI.

16.3.1 Module Memory Map

The memory map for the SPI is given in Figure 16-2. The address listed for each register is the sum of a base address and an address offset. The base address is defined at the SoC level and the address offset is defined at the module level. Reads from the reserved bits return zeros and writes to the reserved bits have no effect.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 SPICR1	R	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
	W								
0x0001 SPICR2	R	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
	W	▨	▨	▨			▨		
0x0002 SPIBR	R	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
	W	▨				▨			
0x0003 SPISR	R	SPIF	0	SPTEF	MODF	0	0	0	0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0004 Reserved	R	▨	▨	▨	▨	▨	▨	▨	▨
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0005 SPIDR	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x0006 Reserved	R	▨	▨	▨	▨	▨	▨	▨	▨
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0007 Reserved	R	▨	▨	▨	▨	▨	▨	▨	▨
	W	▨	▨	▨	▨	▨	▨	▨	▨

▨ = Unimplemented or Reserved

Figure 16-2. SPI Register Summary

16.3.2 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

16.3.2.1 SPI Control Register 1 (SPICR1)

Module Base +0x0000

	7	6	5	4	3	2	1	0
R W	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
Reset	0	0	0	0	0	1	0	0

Figure 16-3. SPI Control Register 1 (SPICR1)

Read: Anytime

Write: Anytime

Table 16-1. SPICR1 Field Descriptions

Field	Description
7 SPIE	**SPI Interrupt Enable Bit** — This bit enables SPI interrupt requests, if SPIF or MODF status flag is set. 0 SPI interrupts disabled. 1 SPI interrupts enabled.
6 SPE	**SPI System Enable Bit** — This bit enables the SPI system and dedicates the SPI port pins to SPI system functions. If SPE is cleared, SPI is disabled and forced into idle state, status bits in SPISR register are reset. 0 SPI disabled (lower power consumption). 1 SPI enabled, port pins are dedicated to SPI functions.
5 SPTIE	**SPI Transmit Interrupt Enable** — This bit enables SPI interrupt requests, if SPTEF flag is set. 0 SPTEF interrupt disabled. 1 SPTEF interrupt enabled.
4 MSTR	**SPI Master/Slave Mode Select Bit** — This bit selects whether the SPI operates in master or slave mode. Switching the SPI from master to slave or vice versa forces the SPI system into idle state. 0 SPI is in slave mode. 1 SPI is in master mode.
3 CPOL	**SPI Clock Polarity Bit** — This bit selects an inverted or non-inverted SPI clock. To transmit data between SPI modules, the SPI modules must have identical CPOL values. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state. 0 Active-high clocks selected. In idle state SCK is low. 1 Active-low clocks selected. In idle state SCK is high.
2 CPHA	**SPI Clock Phase Bit** — This bit is used to select the SPI clock format. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state. 0 Sampling of data occurs at odd edges (1,3,5,...,15) of the SCK clock. 1 Sampling of data occurs at even edges (2,4,6,...,16) of the SCK clock.
1 SSOE	**Slave Select Output Enable** — The $\overline{SS}$ output feature is enabled only in master mode, if MODFEN is set, by asserting the SSOE as shown in Table 16-2. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state.
0 LSBFE	**LSB-First Enable** — This bit does not affect the position of the MSB and LSB in the data register. Reads and writes of the data register always have the MSB in bit 7. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state. 0 Data is transferred most significant bit first. 1 Data is transferred least significant bit first.

Table 16-2. $\overline{SS}$ Input / Output Selection

MODFEN	SSOE	Master Mode	Slave Mode
0	0	$\overline{SS}$ not used by SPI	$\overline{SS}$ input
0	1	$\overline{SS}$ not used by SPI	$\overline{SS}$ input
1	0	$\overline{SS}$ input with MODF feature	$\overline{SS}$ input
1	1	$\overline{SS}$ is slave select output	$\overline{SS}$ input

16.3.2.2 SPI Control Register 2 (SPICR2)

Module Base +0x0001

	7	6	5	4	3	2	1	0
R	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
W	▨	▨	▨			▨		
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 16-4. SPI Control Register 2 (SPICR2)

Read: Anytime

Write: Anytime; writes to the reserved bits have no effect

Table 16-3. SPICR2 Field Descriptions

Field	Description
4 MODFEN	**Mode Fault Enable Bit** — This bit allows the MODF failure to be detected. If the SPI is in master mode and MODFEN is cleared, then the $\overline{SS}$ port pin is not used by the SPI. In slave mode, the $\overline{SS}$ is available only as an input regardless of the value of MODFEN. For an overview on the impact of the MODFEN bit on the $\overline{SS}$ port pin configuration, refer to Table 16-4. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state. 0 $\overline{SS}$ port pin is not used by the SPI. 1 $\overline{SS}$ port pin with MODF feature.
3 BIDIROE	**Output Enable in the Bidirectional Mode of Operation** — This bit controls the MOSI and MISO output buffer of the SPI, when in bidirectional mode of operation (SPC0 is set). In master mode, this bit controls the output buffer of the MOSI port, in slave mode it controls the output buffer of the MISO port. In master mode, with SPC0 set, a change of this bit will abort a transmission in progress and force the SPI into idle state. 0 Output buffer disabled. 1 Output buffer enabled.
1 SPISWAI	**SPI Stop in Wait Mode Bit** — This bit is used for power conservation while in wait mode. 0 SPI clock operates normally in wait mode. 1 Stop SPI clock generation when in wait mode.
0 SPC0	**Serial Pin Control Bit 0** — This bit enables bidirectional pin configurations as shown in Table 16-4. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state.

Table 16-4. Bidirectional Pin Configurations

Pin Mode	SPC0	BIDIROE	MISO	MOSI
Master Mode of Operation				
Normal	0	X	Master In	Master Out
Bidirectional	1	0	MISO not used by SPI	Master In
		1		Master I/O
Slave Mode of Operation				
Normal	0	X	Slave Out	Slave In
Bidirectional	1	0	Slave In	MOSI not used by SPI
		1	Slave I/O	

16.3.2.3 SPI Baud Rate Register (SPIBR)

Module Base +0x0002

	7	6	5	4	3	2	1	0
R	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
W	▨				▨			
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 16-5. SPI Baud Rate Register (SPIBR)

Read: Anytime

Write: Anytime; writes to the reserved bits have no effect

Table 16-5. SPIBR Field Descriptions

Field	Description
6–4 SPPR[2:0]	**SPI Baud Rate Preselection Bits** — These bits specify the SPI baud rates as shown in Table 16-6. In master mode, a change of these bits will abort a transmission in progress and force the SPI system into idle state.
2–0 SPR[2:0]	**SPI Baud Rate Selection Bits** — These bits specify the SPI baud rates as shown in Table 16-6. In master mode, a change of these bits will abort a transmission in progress and force the SPI system into idle state.

The baud rate divisor equation is as follows:

$$\text{BaudRateDivisor} = (\text{SPPR} + 1) \bullet 2^{(\text{SPR} + 1)} \qquad \textit{Eqn. 16-1}$$

The baud rate can be calculated with the following equation:

$$\text{Baud Rate} = \text{BusClock} / \text{BaudRateDivisor} \qquad \textit{Eqn. 16-2}$$

NOTE

For maximum allowed baud rates, please refer to the SPI Electrical Specification in the Electricals chapter of this data sheet.

Table 16-6. Example SPI Baud Rate Selection (25 MHz Bus Clock)

SPPR2	SPPR1	SPPR0	SPR2	SPR1	SPR0	Baud Rate Divisor	Baud Rate
0	0	0	0	0	0	2	12.5 Mbit/s
0	0	0	0	0	1	4	6.25 Mbit/s
0	0	0	0	1	0	8	3.125 Mbit/s
0	0	0	0	1	1	16	1.5625 Mbit/s
0	0	0	1	0	0	32	781.25 kbit/s
0	0	0	1	0	1	64	390.63 kbit/s
0	0	0	1	1	0	128	195.31 kbit/s
0	0	0	1	1	1	256	97.66 kbit/s
0	0	1	0	0	0	4	6.25 Mbit/s
0	0	1	0	0	1	8	3.125 Mbit/s
0	0	1	0	1	0	16	1.5625 Mbit/s
0	0	1	0	1	1	32	781.25 kbit/s
0	0	1	1	0	0	64	390.63 kbit/s
0	0	1	1	0	1	128	195.31 kbit/s
0	0	1	1	1	0	256	97.66 kbit/s
0	0	1	1	1	1	512	48.83 kbit/s
0	1	0	0	0	0	6	4.16667 Mbit/s
0	1	0	0	0	1	12	2.08333 Mbit/s
0	1	0	0	1	0	24	1.04167 Mbit/s
0	1	0	0	1	1	48	520.83 kbit/s
0	1	0	1	0	0	96	260.42 kbit/s
0	1	0	1	0	1	192	130.21 kbit/s
0	1	0	1	1	0	384	65.10 kbit/s
0	1	0	1	1	1	768	32.55 kbit/s
0	1	1	0	0	0	8	3.125 Mbit/s
0	1	1	0	0	1	16	1.5625 Mbit/s
0	1	1	0	1	0	32	781.25 kbit/s
0	1	1	0	1	1	64	390.63 kbit/s
0	1	1	1	0	0	128	195.31 kbit/s
0	1	1	1	0	1	256	97.66 kbit/s
0	1	1	1	1	0	512	48.83 kbit/s
0	1	1	1	1	1	1024	24.41 kbit/s
1	0	0	0	0	0	10	2.5 Mbit/s
1	0	0	0	0	1	20	1.25 Mbit/s
1	0	0	0	1	0	40	625 kbit/s
1	0	0	0	1	1	80	312.5 kbit/s
1	0	0	1	0	0	160	156.25 kbit/s
1	0	0	1	0	1	320	78.13 kbit/s
1	0	0	1	1	0	640	39.06 kbit/s

Table 16-6. Example SPI Baud Rate Selection (25 MHz Bus Clock) (continued)

SPPR2	SPPR1	SPPR0	SPR2	SPR1	SPR0	Baud Rate Divisor	Baud Rate
1	0	0	1	1	1	1280	19.53 kbit/s
1	0	1	0	0	0	12	2.08333 Mbit/s
1	0	1	0	0	1	24	1.04167 Mbit/s
1	0	1	0	1	0	48	520.83 kbit/s
1	0	1	0	1	1	96	260.42 kbit/s
1	0	1	1	0	0	192	130.21 kbit/s
1	0	1	1	0	1	384	65.10 kbit/s
1	0	1	1	1	0	768	32.55 kbit/s
1	0	1	1	1	1	1536	16.28 kbit/s
1	1	0	0	0	0	14	1.78571 Mbit/s
1	1	0	0	0	1	28	892.86 kbit/s
1	1	0	0	1	0	56	446.43 kbit/s
1	1	0	0	1	1	112	223.21 kbit/s
1	1	0	1	0	0	224	111.61 kbit/s
1	1	0	1	0	1	448	55.80 kbit/s
1	1	0	1	1	0	896	27.90 kbit/s
1	1	0	1	1	1	1792	13.95 kbit/s
1	1	1	0	0	0	16	1.5625 Mbit/s
1	1	1	0	0	1	32	781.25 kbit/s
1	1	1	0	1	0	64	390.63 kbit/s
1	1	1	0	1	1	128	195.31 kbit/s
1	1	1	1	0	0	256	97.66 kbit/s
1	1	1	1	0	1	512	48.83 kbit/s
1	1	1	1	1	0	1024	24.41 kbit/s
1	1	1	1	1	1	2048	12.21 kbit/s

16.3.2.4 SPI Status Register (SPISR)

Module Base +0x0003

	7	6	5	4	3	2	1	0
R	SPIF	0	SPTEF	MODF	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	1	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 16-6. SPI Status Register (SPISR)

Read: Anytime

Write: Has no effect

Table 16-7. SPISR Field Descriptions

Field	Description
7 SPIF	**SPIF Interrupt Flag** — This bit is set after a received data byte has been transferred into the SPI data register. This bit is cleared by reading the SPISR register (with SPIF set) followed by a read access to the SPI data register. 0 Transfer not yet complete. 1 New data copied to SPIDR.
5 SPTEF	**SPI Transmit Empty Interrupt Flag** — If set, this bit indicates that the transmit data register is empty. To clear this bit and place data into the transmit data register, SPISR must be read with SPTEF = 1, followed by a write to SPIDR. Any write to the SPI data register without reading SPTEF = 1, is effectively ignored. 0 SPI data register not empty. 1 SPI data register empty.
4 MODF	**Mode Fault Flag** — This bit is set if the $\overline{SS}$ input becomes low while the SPI is configured as a master and mode fault detection is enabled, MODFEN bit of SPICR2 register is set. Refer to MODFEN bit description in Section 16.3.2.2, “SPI Control Register 2 (SPICR2)”. The flag is cleared automatically by a read of the SPI status register (with MODF set) followed by a write to the SPI control register 1. 0 Mode fault has not occurred. 1 Mode fault has occurred.

16.3.2.5 SPI Data Register (SPIDR)

Module Base +0x0005

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	2	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 16-7. SPI Data Register (SPIDR)

Read: Anytime; normally read only when SPIF is set

Write: Anytime

The SPI data register is both the input and output register for SPI data. A write to this register allows a data byte to be queued and transmitted. For an SPI configured as a master, a queued data byte is transmitted immediately after the previous transmission has completed. The SPI transmitter empty flag SPTEF in the SPISR register indicates when the SPI data register is ready to accept new data.

Received data in the SPIDR is valid when SPIF is set.

If SPIF is cleared and a byte has been received, the received byte is transferred from the receive shift register to the SPIDR and SPIF is set.

If SPIF is set and not serviced, and a second byte has been received, the second received byte is kept as valid byte in the receive shift register until the start of another transmission. The byte in the SPIDR does not change.

If SPIF is set and a valid byte is in the receive shift register, and SPIF is serviced before the start of a third transmission, the byte in the receive shift register is transferred into the SPIDR and SPIF remains set (see Figure 16-8).

If SPIF is set and a valid byte is in the receive shift register, and SPIF is serviced after the start of a third transmission, the byte in the receive shift register has become invalid and is not transferred into the SPIDR (see Figure 16-9).

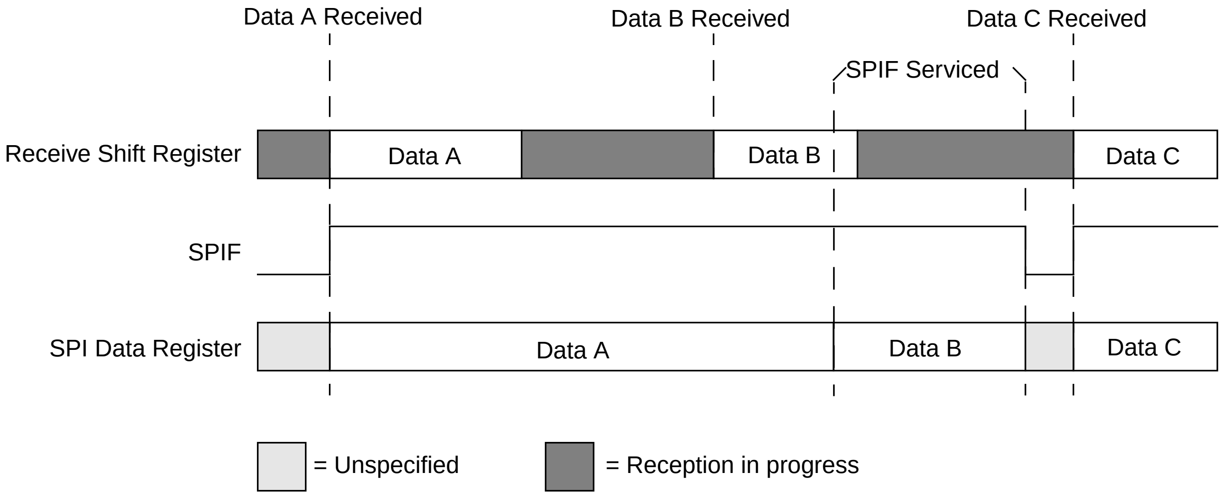


Figure 16-8. Reception with SPIF Serviced in Time

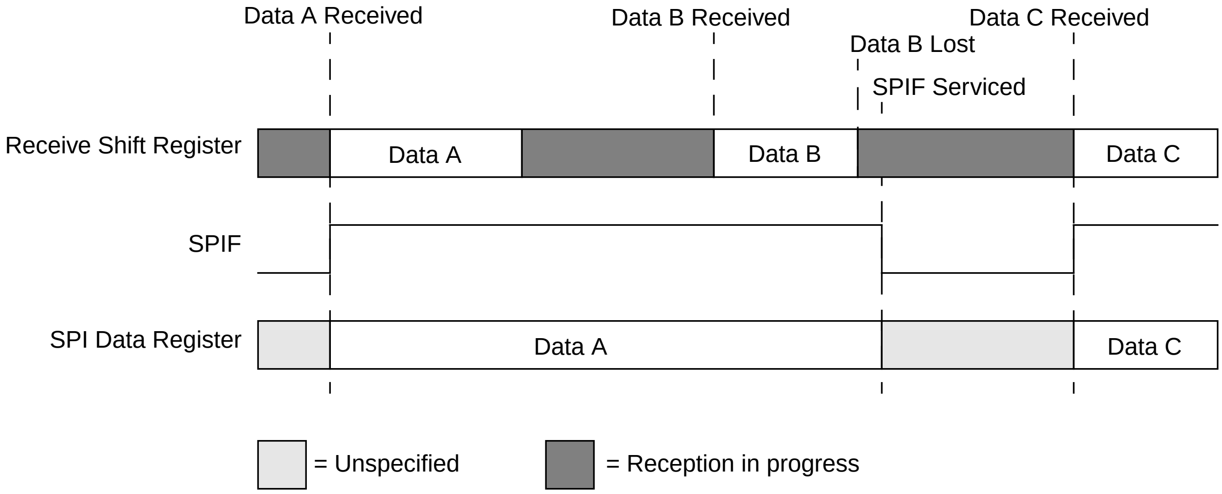


Figure 16-9. Reception with SPIF Serviced too Late

16.4 Functional Description

The SPI module allows a duplex, synchronous, serial communication between the MCU and peripheral devices. Software can poll the SPI status flags or SPI operation can be interrupt driven.

The SPI system is enabled by setting the SPI enable (SPE) bit in SPI control register 1. While SPE is set, the four associated SPI port pins are dedicated to the SPI function as:

- Slave select ($\overline{\mathrm{SS}}$)
- Serial clock (SCK)
- Master out/slave in (MOSI)
- Master in/slave out (MISO)

The main element of the SPI system is the SPI data register. The 8-bit data register in the master and the 8-bit data register in the slave are linked by the MOSI and MISO pins to form a distributed 16-bit register. When a data transfer operation is performed, this 16-bit register is serially shifted eight bit positions by the S-clock from the master, so data is exchanged between the master and the slave. Data written to the master SPI data register becomes the output data for the slave, and data read from the master SPI data register after a transfer operation is the input data from the slave.

A read of SPISR with SPTEF = 1 followed by a write to SPIDR puts data into the transmit data register. When a transfer is complete and SPIF is cleared, received data is moved into the receive data register. This 8-bit data register acts as the SPI receive data register for reads and as the SPI transmit data register for writes. A single SPI register address is used for reading data from the read data buffer and for writing data to the transmit data register.

The clock phase control bit (CPHA) and a clock polarity control bit (CPOL) in the SPI control register 1 (SPICR1) select one of four possible clock formats to be used by the SPI system. The CPOL bit simply selects a non-inverted or inverted clock. The CPHA bit is used to accommodate two fundamentally different protocols by sampling data on odd numbered SCK edges or on even numbered SCK edges (see Section 16.4.3, “Transmission Formats”).

The SPI can be configured to operate as a master or as a slave. When the MSTR bit in SPI control register1 is set, master mode is selected, when the MSTR bit is clear, slave mode is selected.

NOTE

A change of CPOL or MSTR bit while there is a received byte pending in the receive shift register will destroy the received byte and must be avoided.

16.4.1 Master Mode

The SPI operates in master mode when the MSTR bit is set. Only a master SPI module can initiate transmissions. A transmission begins by writing to the master SPI data register. If the shift register is empty, the byte immediately transfers to the shift register. The byte begins shifting out on the MOSI pin under the control of the serial clock.

- Serial clock

  The SPR2, SPR1, and SPR0 baud rate selection bits, in conjunction with the SPPR2, SPPR1, and SPPR0 baud rate preselection bits in the SPI baud rate register, control the baud rate generator and determine the speed of the transmission. The SCK pin is the SPI clock output. Through the SCK pin, the baud rate generator of the master controls the shift register of the slave peripheral.

- MOSI, MISO pin

  In master mode, the function of the serial data output pin (MOSI) and the serial data input pin (MISO) is determined by the SPC0 and BIDIROE control bits.

- $\overline{\text{SS}}$ pin

  If MODFEN and SSOE are set, the $\overline{\text{SS}}$ pin is configured as slave select output. The $\overline{\text{SS}}$ output becomes low during each transmission and is high when the SPI is in idle state.

  If MODFEN is set and SSOE is cleared, the $\overline{\text{SS}}$ pin is configured as input for detecting mode fault error. If the $\overline{\text{SS}}$ input becomes low this indicates a mode fault error where another master tries to drive the MOSI and SCK lines. In this case, the SPI immediately switches to slave mode, by clearing the MSTR bit and also disables the slave output buffer MISO (or SISO in bidirectional mode). So the result is that all outputs are disabled and SCK, MOSI, and MISO are inputs. If a transmission is in progress when the mode fault occurs, the transmission is aborted and the SPI is forced into idle state.

  This mode fault error also sets the mode fault (MODF) flag in the SPI status register (SPISR). If the SPI interrupt enable bit (SPIE) is set when the MODF flag becomes set, then an SPI interrupt sequence is also requested.

  When a write to the SPI data register in the master occurs, there is a half SCK-cycle delay. After the delay, SCK is started within the master. The rest of the transfer operation differs slightly, depending on the clock format specified by the SPI clock phase bit, CPHA, in SPI control register 1 (see Section 16.4.3, “Transmission Formats”).

NOTE

> A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0, or BIDIROE with SPC0 set, SPPR2-SPPR0 and SPR2-SPR0 in master mode will abort a transmission in progress and force the SPI into idle state. The remote slave cannot detect this, therefore the master must ensure that the remote slave is returned to idle state.

16.4.2 Slave Mode

The SPI operates in slave mode when the MSTR bit in SPI control register 1 is clear.

- Serial clock

  In slave mode, SCK is the SPI clock input from the master.

- MISO, MOSI pin

  In slave mode, the function of the serial data output pin (MISO) and serial data input pin (MOSI) is determined by the SPC0 bit and BIDIROE bit in SPI control register 2.

- $\overline{\text{SS}}$ pin

  The $\overline{\text{SS}}$ pin is the slave select input. Before a data transmission occurs, the $\overline{\text{SS}}$ pin of the slave SPI must be low. $\overline{\text{SS}}$ must remain low until the transmission is complete. If $\overline{\text{SS}}$ goes high, the SPI is forced into idle state.

  The $\overline{\text{SS}}$ input also controls the serial data output pin, if $\overline{\text{SS}}$ is high (not selected), the serial data output pin is high impedance, and, if $\overline{\text{SS}}$ is low, the first bit in the SPI data register is driven out of the serial data output pin. Also, if the slave is not selected ($\overline{\text{SS}}$ is high), then the SCK input is ignored and no internal shifting of the SPI shift register occurs.

  Although the SPI is capable of duplex operation, some SPI peripherals are capable of only receiving SPI data in a slave mode. For these simpler devices, there is no serial data out pin.

NOTE

> When peripherals with duplex capability are used, take care not to simultaneously enable two receivers whose serial outputs drive the same system slave's serial data output line.

As long as no more than one slave device drives the system slave's serial data output line, it is possible for several slaves to receive the same transmission from a master, although the master would not receive return information from all of the receiving slaves.

If the CPHA bit in SPI control register 1 is clear, odd numbered edges on the SCK input cause the data at the serial data input pin to be latched. Even numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

If the CPHA bit is set, even numbered edges on the SCK input cause the data at the serial data input pin to be latched. Odd numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

When CPHA is set, the first edge is used to get the first data bit onto the serial data output pin. When CPHA is clear and the $\overline{\text{SS}}$ input is low (slave selected), the first bit of the SPI data is driven out of the serial data output pin. After the eighth shift, the transfer is considered complete and the received data is transferred into the SPI data register. To indicate transfer is complete, the SPIF flag in the SPI status register is set.

NOTE

> A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0, or BIDIROE with SPC0 set in slave mode will corrupt a transmission in progress and must be avoided.

16.4.3 Transmission Formats

During an SPI transmission, data is transmitted (shifted out serially) and received (shifted in serially) simultaneously. The serial clock (SCK) synchronizes shifting and sampling of the information on the two serial data lines. A slave select line allows selection of an individual slave SPI device; slave devices that are not selected do not interfere with SPI bus activities. Optionally, on a master SPI device, the slave select line can be used to indicate multiple-master bus contention.

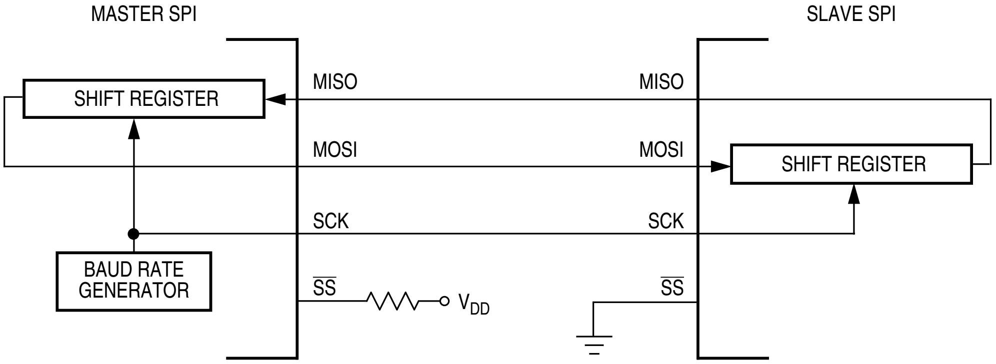


Figure 16-10. Master/Slave Transfer Block Diagram

16.4.3.1 Clock Phase and Polarity Controls

Using two bits in the SPI control register 1, software selects one of four combinations of serial clock phase and polarity.

The CPOL clock polarity control bit specifies an active high or low clock and has no significant effect on the transmission format.

The CPHA clock phase control bit selects one of two fundamentally different transmission formats.

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 transmissions to allow a master device to communicate with peripheral slaves having different requirements.

16.4.3.2 CPHA = 0 Transfer Format

The first edge on the SCK line is used to clock the first data bit of the slave into the master and the first data bit of the master into the slave. In some peripherals, the first bit of the slave's data is available at the slave's data out pin as soon as the slave is selected. In this format, the first SCK edge is issued a half cycle after $\overline{SS}$ has become low.

A half SCK cycle later, the second edge appears on the SCK line. When this second edge occurs, the value previously latched from the serial data input pin is shifted into the LSB or MSB of the shift register, depending on LSBFE bit.

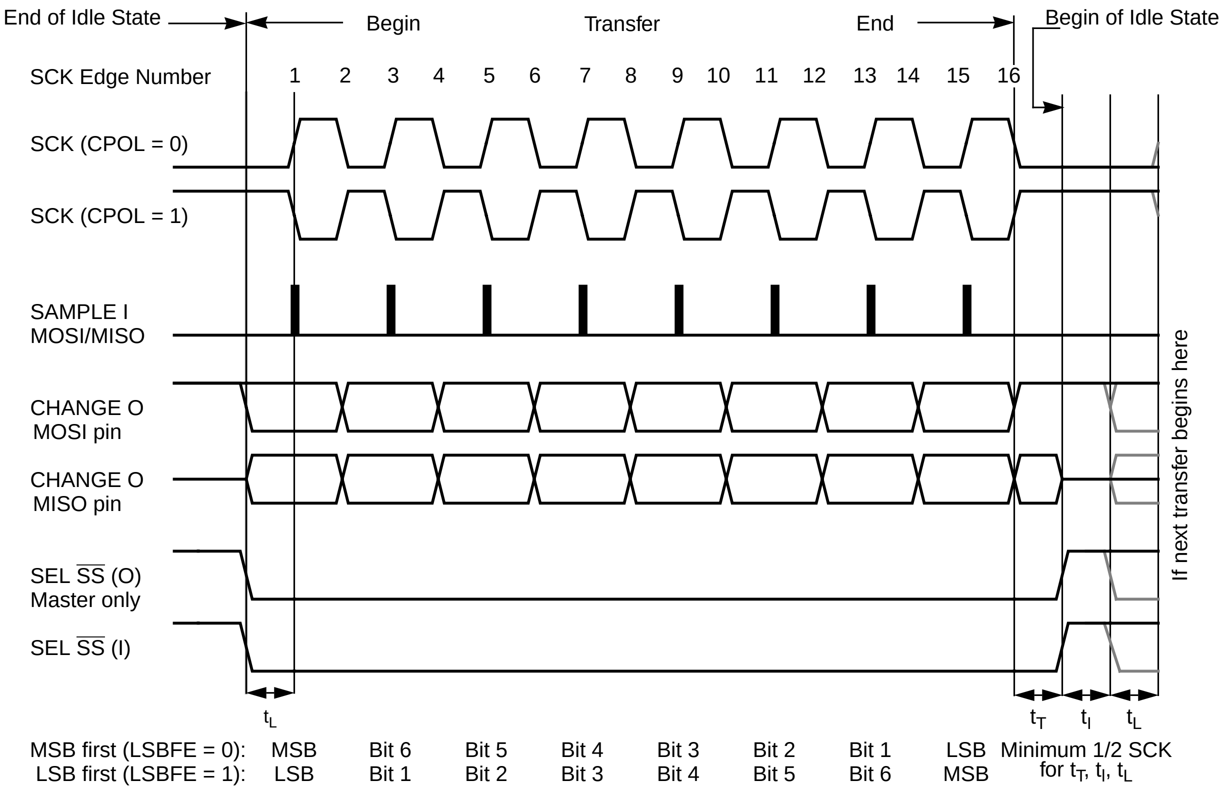
After this second edge, the next bit of the SPI master data is transmitted out of the serial data output pin of the master to the serial input pin on the slave. This process continues for a total of 16 edges on the SCK line, with data being latched on odd numbered edges and shifted on even numbered edges.

Data reception is double buffered. Data is shifted serially into the SPI shift register during the transfer and is transferred to the parallel SPI data register after the last bit is shifted in.

After the 16th (last) SCK edge:

- Data that was previously in the master SPI data register should now be in the slave data register and the data that was in the slave data register should be in the master.
- The SPIF flag in the SPI status register is set, indicating that the transfer is complete.

Figure 16-11 is a timing diagram of an SPI transfer where CPHA = 0. SCK waveforms are shown for CPOL = 0 and CPOL = 1. The diagram may be interpreted as a master or slave timing diagram because the SCK, MISO, and MOSI pins are connected directly 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. The $\overline{SS}$ pin of the master must be either high or reconfigured as a general-purpose output not affecting the SPI.



t_L = Minimum leading time before the first SCK edge
t_T = Minimum trailing time after the last SCK edge
t_I = Minimum idling time between transfers (minimum $\overline{SS}$ high time)
t_L, t_T, and t_I are guaranteed for the master mode and required for the slave mode.

Figure 16-11. SPI Clock Format 0 (CPHA = 0)

In slave mode, if the $\overline{\text{SS}}$ line is not deasserted between the successive transmissions then the content of the SPI data register is not transmitted; instead the last received byte is transmitted. If the $\overline{\text{SS}}$ line is deasserted for at least minimum idle time (half SCK cycle) between successive transmissions, then the content of the SPI data register is transmitted.

In master mode, with slave select output enabled the $\overline{\text{SS}}$ line is always deasserted and reasserted between successive transfers for at least minimum idle time.

16.4.3.3 CPHA = 1 Transfer Format

Some peripherals require the first SCK edge before the first data bit becomes available at the data out pin, the second edge clocks data into the system. In this format, the first SCK edge is issued by setting the CPHA bit at the beginning of the 8-cycle transfer operation.

The first edge of SCK occurs immediately after the half SCK clock cycle synchronization delay. This first edge commands the slave to transfer its first data bit to the serial data input pin of the master.

A half SCK cycle later, the second edge appears on the SCK pin. This is the latching edge for both the master and slave.

When the third edge occurs, the value previously latched from the serial data input pin is shifted into the LSB or MSB of the SPI shift register, depending on LSBFE bit. After this edge, the next bit of the master data is coupled out of the serial data output pin of the master to the serial input pin on the slave.

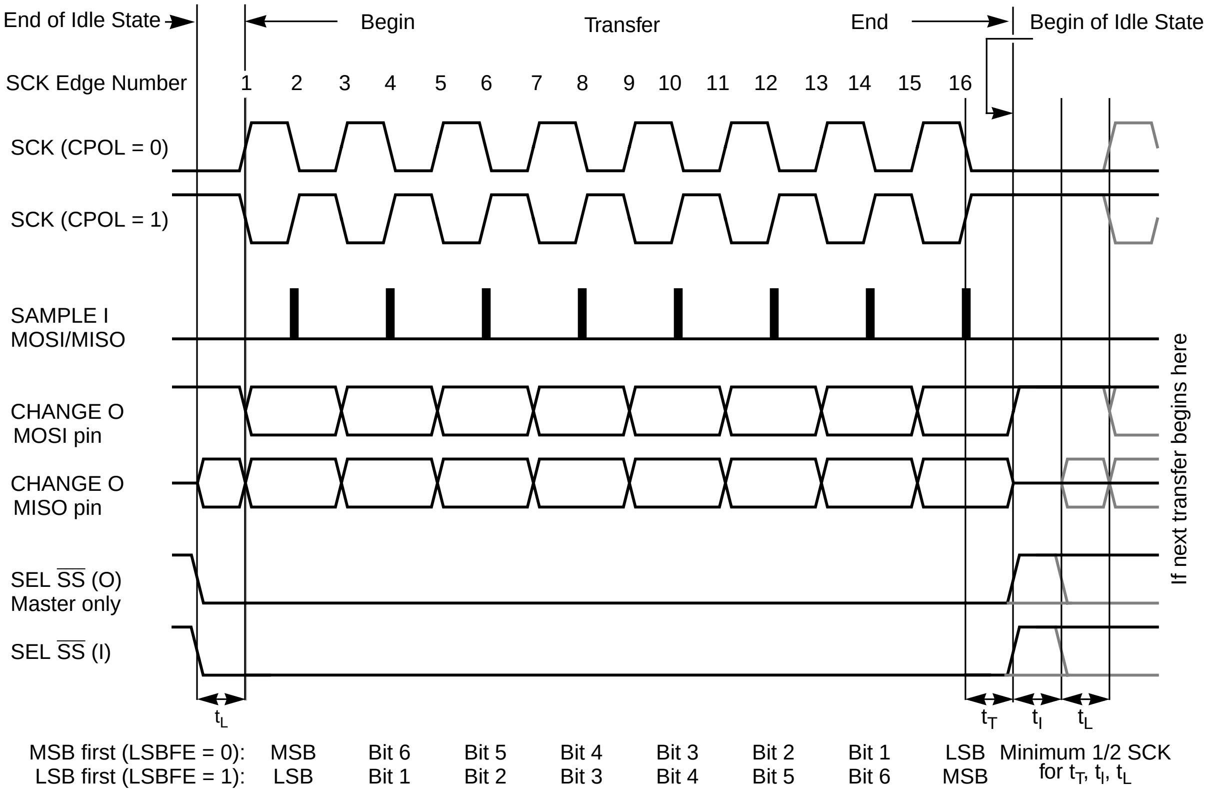
This process continues for a total of 16 edges on the SCK line with data being latched on even numbered edges and shifting taking place on odd numbered edges.

Data reception is double buffered, data is serially shifted into the SPI shift register during the transfer and is transferred to the parallel SPI data register after the last bit is shifted in.

After the 16th SCK edge:

- Data that was previously in the SPI data register of the master is now in the data register of the slave, and data that was in the data register of the slave is in the master.
- The SPIF flag bit in SPISR is set indicating that the transfer is complete.

Figure 16-12 shows two clocking variations for CPHA = 1. The diagram may be interpreted as a master or slave timing diagram because the SCK, MISO, and MOSI pins are connected directly 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. The $\overline{\text{SS}}$ line is the slave select input to the slave. The $\overline{\text{SS}}$ pin of the master must be either high or reconfigured as a general-purpose output not affecting the SPI.



t_L = Minimum leading time before the first SCK edge, not required for back-to-back transfers
t_T = Minimum trailing time after the last SCK edge
t_I = Minimum idling time between transfers (minimum $\overline{SS}$ high time), not required for back-to-back transfers

Figure 16-12. SPI Clock Format 1 (CPHA = 1)

The $\overline{SS}$ line can remain active low between successive transfers (can be tied low at all times). This format is sometimes preferred in systems having a single fixed master and a single slave that drive the MISO data line.

- Back-to-back transfers in master mode

  In master mode, if a transmission has completed and a new data byte is available in the SPI data register, this byte is sent out immediately without a trailing and minimum idle time.

The SPI interrupt request flag (SPIF) is common to both the master and slave modes. SPIF gets set one half SCK cycle after the last SCK edge.

16.4.4 SPI Baud Rate Generation

Baud rate generation consists of a series of divider stages. Six bits in the SPI baud rate register (SPPR2, SPPR1, SPPR0, SPR2, SPR1, and SPR0) determine the divisor to the SPI module clock which results in the SPI baud rate.

The SPI clock rate is determined by the product of the value in the baud rate preselection bits (SPPR2–SPPR0) and the value in the baud rate selection bits (SPR2–SPR0). The module clock divisor equation is shown in Equation 16-3.

$$\text{BaudRateDivisor} = (\text{SPPR} + 1) \bullet 2^{(\text{SPR} + 1)} \qquad \textit{Eqn. 16-3}$$

When all bits are clear (the default condition), the SPI module clock is divided by 2. When the selection bits (SPR2–SPR0) are 001 and the preselection bits (SPPR2–SPPR0) are 000, the module clock divisor becomes 4. When the selection bits are 010, the module clock divisor becomes 8, etc.

When the preselection bits are 001, the divisor determined by the selection bits is multiplied by 2. When the preselection bits are 010, the divisor is multiplied by 3, etc. See Table 16-6 for baud rate calculations for all bit conditions, based on a 25 MHz bus clock. The two sets of selects allows the clock to be divided by a non-power of two to achieve other baud rates such as divide by 6, divide by 10, etc.

The baud rate generator is activated only when the SPI is in master mode and a serial transfer is taking place. In the other cases, the divider is disabled to decrease I_{DD} current.

NOTE

For maximum allowed baud rates, please refer to the SPI Electrical Specification in the Electricals chapter of this data sheet.

16.4.5 Special Features

16.4.5.1 $\overline{\text{SS}}$ Output

The $\overline{\text{SS}}$ output feature automatically drives the $\overline{\text{SS}}$ pin low during transmission to select external devices and drives it high during idle to deselect external devices. When $\overline{\text{SS}}$ output is selected, the $\overline{\text{SS}}$ output pin is connected to the $\overline{\text{SS}}$ input pin of the external device.

The $\overline{\text{SS}}$ output is available only in master mode during normal SPI operation by asserting SSOE and MODFEN bit as shown in Table 16-2.

The mode fault feature is disabled while $\overline{\text{SS}}$ output is enabled.

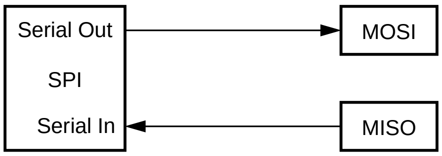
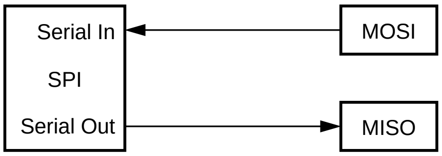
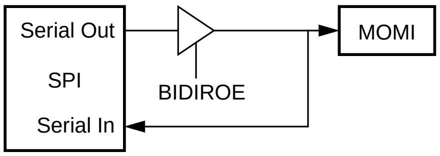
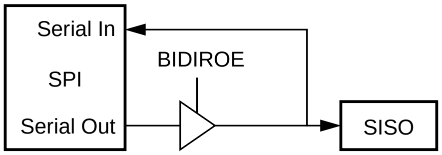
NOTE

Care must be taken when using the $\overline{\text{SS}}$ output feature in a multimaster system because the mode fault feature is not available for detecting system errors between masters.

16.4.5.2 Bidirectional Mode (MOMI or SISO)

The bidirectional mode is selected when the SPC0 bit is set in SPI control register 2 (see Table 16-8). In this mode, the SPI uses only one serial data pin for the interface with external device(s). The MSTR bit decides which pin to use. The MOSI pin becomes the serial data I/O (MOMI) pin for the master mode, and the MISO pin becomes serial data I/O (SISO) pin for the slave mode. The MISO pin in master mode and MOSI pin in slave mode are not used by the SPI.

Table 16-8. Normal Mode and Bidirectional Mode

When SPE = 1	Master Mode MSTR = 1	Slave Mode MSTR = 0
Normal Mode **SPC0 = 0**		
Bidirectional Mode **SPC0 = 1**		

The direction of each serial I/O pin depends on the BIDIROE bit. If the pin is configured as an output, serial data from the shift register is driven out on the pin. The same pin is also the serial input to the shift register.

- The SCK is output for the master mode and input for the slave mode.
- The $\overline{\text{SS}}$ is the input or output for the master mode, and it is always the input for the slave mode.
- The bidirectional mode does not affect SCK and $\overline{\text{SS}}$ functions.

NOTE

In bidirectional master mode, with mode fault enabled, both data pins MISO and MOSI can be occupied by the SPI, though MOSI is normally used for transmissions in bidirectional mode and MISO is not used by the SPI. If a mode fault occurs, the SPI is automatically switched to slave mode. In this case MISO becomes occupied by the SPI and MOSI is not used. This must be considered, if the MISO pin is used for another purpose.

16.4.6 Error Conditions

The SPI has one error condition:

- Mode fault error

16.4.6.1 Mode Fault Error

If the $\overline{SS}$ input becomes low while the SPI is configured as a master, it indicates a system error where more than one master may be trying to drive the MOSI and SCK lines simultaneously. This condition is not permitted in normal operation, the MODF bit in the SPI status register is set automatically, provided the MODFEN bit is set.

In the special case where the SPI is in master mode and MODFEN bit is cleared, the $\overline{SS}$ pin is not used by the SPI. In this special case, the mode fault error function is inhibited and MODF remains cleared. In case the SPI system is configured as a slave, the $\overline{SS}$ pin is a dedicated input pin. Mode fault error doesn't occur in slave mode.

If a mode fault error occurs, the SPI is switched to slave mode, with the exception that the slave output buffer is disabled. So SCK, MISO, and MOSI pins are forced to be high impedance inputs to avoid any possibility of conflict with another output driver. A transmission in progress is aborted and the SPI is forced into idle state.

If the mode fault error occurs in the bidirectional mode for a SPI system configured in master mode, output enable of the MOMI (MOSI in bidirectional mode) is cleared if it was set. No mode fault error occurs in the bidirectional mode for SPI system configured in slave mode.

The mode fault flag is cleared automatically by a read of the SPI status register (with MODF set) followed by a write to SPI control register 1. If the mode fault flag is cleared, the SPI becomes a normal master or slave again.

NOTE

If a mode fault error occurs and a received data byte is pending in the receive shift register, this data byte will be lost.

16.4.7 Low Power Mode Options

16.4.7.1 SPI in Run Mode

In run mode with the SPI system enable (SPE) bit in the SPI control register clear, the SPI system is in a low-power, disabled state. SPI registers remain accessible, but clocks to the core of this module are disabled.

16.4.7.2 SPI in Wait Mode

SPI operation in wait mode depends upon the state of the SPISWAI bit in SPI control register 2.

- If SPISWAI is clear, the SPI operates normally when the CPU is in wait mode
- If SPISWAI is set, SPI clock generation ceases and the SPI module enters a power conservation state when the CPU is in wait mode.
  - If SPISWAI is set and the SPI is configured for master, any transmission and reception in progress stops at wait mode entry. The transmission and reception resumes when the SPI exits wait mode.
  - If SPISWAI is set and the SPI is configured as a slave, any transmission and reception in progress continues if the SCK continues to be driven from the master. This keeps the slave synchronized to the master and the SCK.

    If the master transmits several bytes while the slave is in wait mode, the slave will continue to send out bytes consistent with the operation mode at the start of wait mode (i.e., if the slave is currently sending its SPIDR to the master, it will continue to send the same byte. Else if the slave is currently sending the last received byte from the master, it will continue to send each previous master byte).

NOTE

> Care must be taken when expecting data from a master while the slave is in wait or stop mode. Even though the shift register will continue to operate, the rest of the SPI is shut down (i.e., a SPIF interrupt will **not** be generated until exiting stop or wait mode). Also, the byte from the shift register will not be copied into the SPIDR register until after the slave SPI has exited wait or stop mode. In slave mode, a received byte pending in the receive shift register will be lost when entering wait or stop mode. An SPIF flag and SPIDR copy is generated only if wait mode is entered or exited during a tranmission. If the slave enters wait mode in idle mode and exits wait mode in idle mode, neither a SPIF nor a SPIDR copy will occur.

16.4.7.3 SPI in Stop Mode

Stop mode is dependent on the system. The SPI enters stop mode when the module clock is disabled (held high or low). If the SPI is in master mode and exchanging data when the CPU enters stop mode, the transmission is frozen until the CPU exits stop mode. After stop, data to and from the external SPI is exchanged correctly. In slave mode, the SPI will stay synchronized with the master.

The stop mode is not dependent on the SPISWAI bit.

16.4.7.4 Reset

The reset values of registers and signals are described in Section 16.3, “Memory Map and Register Definition”, which details the registers and their bit fields.

- If a data transmission occurs in slave mode after reset without a write to SPIDR, it will transmit garbage, or the byte last received from the master before the reset.
- Reading from the SPIDR after reset will always read a byte of zeros.

16.4.7.5 Interrupts

The SPI only originates interrupt requests when SPI is enabled (SPE bit in SPICR1 set). The following is a description of how the SPI makes a request and how the MCU should acknowledge that request. The interrupt vector offset and interrupt priority are chip dependent.

The interrupt flags MODF, SPIF, and SPTEF are logically ORed to generate an interrupt request.

16.4.7.5.1 MODF

MODF occurs when the master detects an error on the $\overline{\text{SS}}$ pin. The master SPI must be configured for the MODF feature (see Table 16-2). After MODF is set, the current transfer is aborted and the following bit is changed:

- MSTR = 0, The master bit in SPICR1 resets.

The MODF interrupt is reflected in the status register MODF flag. Clearing the flag will also clear the interrupt. This interrupt will stay active while the MODF flag is set. MODF has an automatic clearing process which is described in Section 16.3.2.4, “SPI Status Register (SPISR)”.

16.4.7.5.2 SPIF

SPIF occurs when new data has been received and copied to the SPI data register. After SPIF is set, it does not clear until it is serviced. SPIF has an automatic clearing process, which is described in Section 16.3.2.4, “SPI Status Register (SPISR)”.

16.4.7.5.3 SPTEF

SPTEF occurs when the SPI data register is ready to accept new data. After SPTEF is set, it does not clear until it is serviced. SPTEF has an automatic clearing process, which is described in Section 16.3.2.4, “SPI Status Register (SPISR)”.

Chapter 17
Periodic Interrupt Timer (S12PIT24B4CV1)

17.1 Introduction

The period interrupt timer (PIT) is an array of 24-bit timers that can be used to trigger peripheral modules or raise periodic interrupts. Refer to Figure 17-1 for a simplified block diagram.

17.1.1 Glossary

Acronyms and Abbreviations	
PIT	Periodic Interrupt Timer
ISR	Interrupt Service Routine
CCR	Condition Code Register
SoC	System on Chip
micro time bases	clock periods of the 16-bit timer modulus down-counters, which are generated by the 8-bit modulus down-counters.

17.1.2 Features

The PIT includes these features:

- Four timers implemented as modulus down-counters with independent time-out periods.
- Time-out periods selectable between 1 and 2^{24} bus clock cycles. Time-out equals m*n bus clock cycles with 1 <= m <= 256 and 1 <= n <= 65536.
- Timers that can be enabled individually.
- Four time-out interrupts.
- Four time-out trigger output signals available to trigger peripheral modules.
- Start of timer channels can be aligned to each other.

17.1.3 Modes of Operation

Refer to the SoC guide for a detailed explanation of the chip modes.

- Run mode

  This is the basic mode of operation.
- Wait mode

PIT operation in wait mode is controlled by the PITSWAI bit located in the PITCFLMT register. In wait mode, if the bus clock is globally enabled and if the PITSWAI bit is clear, the PIT operates like in run mode. In wait mode, if the PITSWAI bit is set, the PIT module is stalled.

- Stop mode

  In full stop mode or pseudo stop mode, the PIT module is stalled.

- Freeze mode

  PIT operation in freeze mode is controlled by the PITFRZ bit located in the PITCFLMT register. In freeze mode, if the PITFRZ bit is clear, the PIT operates like in run mode. In freeze mode, if the PITFRZ bit is set, the PIT module is stalled.

17.1.4 Block Diagram

Figure 17-1 shows a block diagram of the PIT.

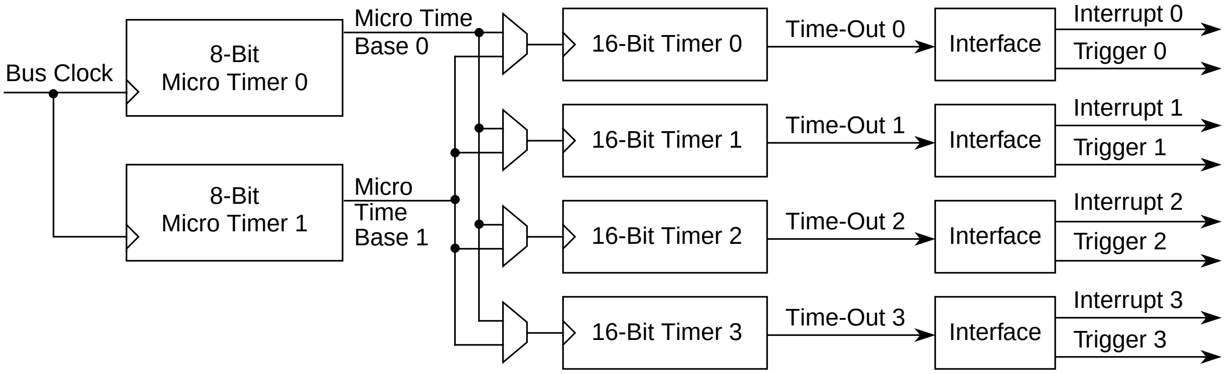


Figure 17-1. PIT Block Diagram

17.2 External Signal Description

The PIT module has no external pins.

17.3 Memory Map and Register Definition

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 PITCFLMT	R	PITE	PITSWAI	PITFRZ	0	0	0	0	0
	W				▨	▨	▨	PFLMT1	PFLMT0
0x0001 PITFLT	R	0	0	0	0	0	0	0	0
	W	▨	▨	▨	▨	PFLT3	PFLT2	PFLT1	PFLT0
0x0002 PITCE	R	0	0	0	0	PCE3	PCE2	PCE1	PCE0
	W	▨	▨	▨	▨				
0x0003 PITMUX	R	0	0	0	0	PMUX3	PMUX2	PMUX1	PMUX0
	W	▨	▨	▨	▨				
0x0004 PITINTE	R	0	0	0	0	PINTE3	PINTE2	PINTE1	PINTE0
	W	▨	▨	▨	▨				
0x0005 PITTF	R	0	0	0	0	PTF3	PTF2	PTF1	PTF0
	W	▨	▨	▨	▨				
0x0006 PITMTLD0	R	PMTLD7	PMTLD6	PMTLD5	PMTLD4	PMTLD3	PMTLD2	PMTLD1	PMTLD0
	W								
0x0007 PITMTLD1	R	PMTLD7	PMTLD6	PMTLD5	PMTLD4	PMTLD3	PMTLD2	PMTLD1	PMTLD0
	W								
0x0008 PITLD0 (High)	R	PLD15	PLD14	PLD13	PLD12	PLD11	PLD10	PLD9	PLD8
	W								
0x0009 PITLD0 (Low)	R	PLD7	PLD6	PLD5	PLD4	PLD3	PLD2	PLD1	PLD0
	W								
0x000A PITCNT0 (High)	R	PCNT15	PCNT14	PCNT13	PCNT12	PCNT11	PCNT10	PCNT9	PCNT8
	W								
0x000B PITCNT0 (Low)	R	PCNT7	PCNT6	PCNT5	PCNT4	PCNT3	PCNT2	PCNT1	PCNT0
	W								
0x000C PITLD1 (High)	R	PLD15	PLD14	PLD13	PLD12	PLD11	PLD10	PLD9	PLD8
	W								

▨ = Unimplemented or Reserved

Figure 17-2. PIT Register Summary (Sheet 1 of 2)

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x000D PITLD1 (Low)	R W	PLD7	PLD6	PLD5	PLD4	PLD3	PLD2	PLD1	PLD0
0x000E PITCNT1 (High)	R W	PCNT15	PCNT14	PCNT13	PCNT12	PCNT11	PCNT10	PCNT9	PCNT8
0x000F PITCNT1 (Low)	R W	PCNT7	PCNT6	PCNT5	PCNT4	PCNT3	PCNT2	PCNT1	PCNT0
0x0010 PITLD2 (High)	R W	PLD15	PLD14	PLD13	PLD12	PLD11	PLD10	PLD9	PLD8
0x0011 PITLD2 (Low)	R W	PLD7	PLD6	PLD5	PLD4	PLD3	PLD2	PLD1	PLD0
0x0012 PITCNT2 (High)	R W	PCNT15	PCNT14	PCNT13	PCNT12	PCNT11	PCNT10	PCNT9	PCNT8
0x0013 PITCNT2 (Low)	R W	PCNT7	PCNT6	PCNT5	PCNT4	PCNT3	PCNT2	PCNT1	PCNT0
0x0014 PITLD3 (High)	R W	PLD15	PLD14	PLD13	PLD12	PLD11	PLD10	PLD9	PLD8
0x0015 PITLD3 (Low)	R W	PLD7	PLD6	PLD5	PLD4	PLD3	PLD2	PLD1	PLD0
0x0016 PITCNT3 (High)	R W	PCNT15	PCNT14	PCNT13	PCNT12	PCNT11	PCNT10	PCNT9	PCNT8
0x0017 PITCNT3 (Low)	R W	PCNT7	PCNT6	PCNT5	PCNT4	PCNT3	PCNT2	PCNT1	PCNT0

▨ = Unimplemented or Reserved

Figure 17-2. PIT Register Summary (Sheet 2 of 2)

17.3.0.1 PIT Control and Force Load Micro Timer Register (PITCFLMT)

Module Base + 0x0000

	7	6	5	4	3	2	1	0
R	PITE	PITSWAI	PITFRZ	0	0	0	0	0
W				▨	▨	▨	PFLMT1	PFLMT0
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 17-3. PIT Control and Force Load Micro Timer Register (PITCFLMT)

Read: Anytime

Write: Anytime; writes to the reserved bits have no effect

Table 17-1. PITCFLMT Field Descriptions

Field	Description
7 PITE	**PIT Module Enable Bit** — This bit enables the PIT module. If PITE is cleared, the PIT module is disabled and flag bits in the PITTF register are cleared. When PITE is set, individually enabled timers (PCE set) start down-counting with the corresponding load register values. 0 PIT disabled (lower power consumption). 1 PIT is enabled.
6 PITSWAI	**PIT Stop in Wait Mode Bit** — This bit is used for power conservation while in wait mode. 0 PIT operates normally in wait mode 1 PIT clock generation stops and freezes the PIT module when in wait mode
5 PITFRZ	**PIT Counter Freeze while in Freeze Mode Bit** — When during debugging a breakpoint (freeze mode) is encountered it is useful in many cases to freeze the PIT counters to avoid e.g. interrupt generation. The PITFRZ bit controls the PIT operation while in freeze mode. 0 PIT operates normally in freeze mode 1 PIT counters are stalled when in freeze mode
1:0 PFLMT[1:0]	**PIT Force Load Bits for Micro Timer 1:0** — These bits have only an effect if the corresponding micro timer is active and if the PIT module is enabled (PITE set). Writing a one into a PFLMT bit loads the corresponding 8-bit micro timer load register into the 8-bit micro timer down-counter. Writing a zero has no effect. Reading these bits will always return zero. **Note:** A micro timer force load affects all timer channels that use the corresponding micro time base.

17.3.0.2 PIT Force Load Timer Register (PITFLT)

Module Base + 0x0001

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	PFLT3	PFLT2	PFLT1	PFLT0
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 17-4. PIT Force Load Timer Register (PITFLT)

Read: Anytime

Write: Anytime; writes to the reserved bits have no effect

Table 17-2. PITFLT Field Descriptions

Field	Description
3:0 PFLT[3:0]	**PIT Force Load Bits for Timer 3-0** — These bits have only an effect if the corresponding timer channel (PCE set) is enabled and if the PIT module is enabled (PITE set). Writing a one into a PFLT bit loads the corresponding 16-bit timer load register into the 16-bit timer down-counter. Writing a zero has no effect. Reading these bits will always return zero.

17.3.0.3 PIT Channel Enable Register (PITCE)

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R	0	0	0	0	PCE3	PCE2	PCE1	PCE0
W	▨	▨	▨	▨				
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 17-5. PIT Channel Enable Register (PITCE)

Read: Anytime

Write: Anytime; writes to the reserved bits have no effect

Table 17-3. PITCE Field Descriptions

Field	Description
3:0 PCE[3:0]	**PIT Enable Bits for Timer Channel 3:0** — These bits enable the PIT channels 3-0. If PCE is cleared, the PIT channel is disabled and the corresponding flag bit in the PITTF register is cleared. When PCE is set, and if the PIT module is enabled (PITE = 1) the 16-bit timer counter is loaded with the start count value and starts down-counting. 0 The corresponding PIT channel is disabled. 1 The corresponding PIT channel is enabled.

17.3.0.4 PIT Multiplex Register (PITMUX)

Module Base + 0x0003

	7	6	5	4	3	2	1	0
R	0	0	0	0	PMUX3	PMUX2	PMUX1	PMUX0
W	▨	▨	▨	▨				
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 17-6. PIT Multiplex Register (PITMUX)

Read: Anytime

Write: Anytime; writes to the reserved bits have no effect

Table 17-4. PITMUX Field Descriptions

Field	Description
3:0 PMUX[3:0]	**PIT Multiplex Bits for Timer Channel 3:0** — These bits select if the corresponding 16-bit timer is connected to micro time base 1 or 0. If PMUX is modified, the corresponding 16-bit timer is immediately switched to the other micro time base. 0 The corresponding 16-bit timer counts with micro time base 0. 1 The corresponding 16-bit timer counts with micro time base 1.

17.3.0.5 PIT Interrupt Enable Register (PITINTE)

Module Base + 0x0004

	7	6	5	4	3	2	1	0
R	0	0	0	0	PINTE3	PINTE2	PINTE1	PINTE0
W	▨	▨	▨	▨				
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 17-7. PIT Interrupt Enable Register (PITINTE)

Read: Anytime

Write: Anytime; writes to the reserved bits have no effect

Table 17-5. PITINTE Field Descriptions

Field	Description
3:0 PINTE[3:0]	**PIT Time-out Interrupt Enable Bits for Timer Channel 3:0** — These bits enable an interrupt service request whenever the time-out flag PTF of the corresponding PIT channel is set. When an interrupt is pending (PTF set) enabling the interrupt will immediately cause an interrupt. To avoid this, the corresponding PTF flag has to be cleared first. 0 Interrupt of the corresponding PIT channel is disabled. 1 Interrupt of the corresponding PIT channel is enabled.

17.3.0.6 PIT Time-Out Flag Register (PITTF)

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R	0	0	0	0	PTF3	PTF2	PTF1	PTF0
W	▨	▨	▨	▨				
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 17-8. PIT Time-Out Flag Register (PITTF)

Read: Anytime

Write: Anytime (write to clear); writes to the reserved bits have no effect

Table 17-6. PITTF Field Descriptions

Field	Description
3:0 PTF[3:0]	**PIT Time-out Flag Bits for Timer Channel 3:0** — PTF is set when the corresponding 16-bit timer modulus down-counter and the selected 8-bit micro timer modulus down-counter have counted to zero. The flag can be cleared by writing a one to the flag bit. Writing a zero has no effect. If flag clearing by writing a one and flag setting happen in the same bus clock cycle, the flag remains set. The flag bits are cleared if the PIT module is disabled or if the corresponding timer channel is disabled. 0 Time-out of the corresponding PIT channel has not yet occurred. 1 Time-out of the corresponding PIT channel has occurred.

17.3.0.7 PIT Micro Timer Load Register 0 to 1 (PITMTLD0–1)

Module Base + 0x0006

	7	6	5	4	3	2	1	0
R	PMTLD7	PMTLD6	PMTLD5	PMTLD4	PMTLD3	PMTLD2	PMTLD1	PMTLD0
W								
Reset	0	0	0	0	0	0	0	0

Figure 17-9. PIT Micro Timer Load Register 0 (PITMTLD0)

Module Base + 0x0007

	7	6	5	4	3	2	1	0
R	PMTLD7	PMTLD6	PMTLD5	PMTLD4	PMTLD3	PMTLD2	PMTLD1	PMTLD0
W								
Reset	0	0	0	0	0	0	0	0

Figure 17-10. PIT Micro Timer Load Register 1 (PITMTLD1)

Read: Anytime

Write: Anytime

Table 17-7. PITMTLD0–1 Field Descriptions

Field	Description
7:0 PMTLD[7:0]	**PIT Micro Timer Load Bits 7:0** — These bits set the 8-bit modulus down-counter load value of the micro timers. Writing a new value into the PITMTLD register will not restart the timer. When the micro timer has counted down to zero, the PMTLD register value will be loaded. The PFLMT bits in the PITCFLMT register can be used to immediately update the count register with the new value if an immediate load is desired.

17.3.0.8 PIT Load Register 0 to 3 (PITLD0–3)

Module Base + 0x0008, 0x0009

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PLD15	PLD14	PLD13	PLD12	PLD11	PLD10	PLD9	PLD8	PLD7	PLD6	PLD5	PLD4	PLD3	PLD2	PLD1	PLD0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-11. PIT Load Register 0 (PITLD0)

Module Base + 0x000C, 0x000D

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PLD15	PLD14	PLD13	PLD12	PLD11	PLD10	PLD9	PLD8	PLD7	PLD6	PLD5	PLD4	PLD3	PLD2	PLD1	PLD0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-12. PIT Load Register 1 (PITLD1)

Module Base + 0x0010, 0x0011

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PLD15	PLD14	PLD13	PLD12	PLD11	PLD10	PLD9	PLD8	PLD7	PLD6	PLD5	PLD4	PLD3	PLD2	PLD1	PLD0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-13. PIT Load Register 2 (PITLD2)

Module Base + 0x0014, 0x0015

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PLD15	PLD14	PLD13	PLD12	PLD11	PLD10	PLD9	PLD8	PLD7	PLD6	PLD5	PLD4	PLD3	PLD2	PLD1	PLD0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-14. PIT Load Register 3 (PITLD3)

Read: Anytime

Write: Anytime

Table 17-8. PITLD0–3 Field Descriptions

Field	Description
15:0 PLD[15:0]	**PIT Load Bits 15:0** — These bits set the 16-bit modulus down-counter load value. Writing a new value into the PITLD register must be a 16-bit access, to ensure data consistency. It will not restart the timer. When the timer has counted down to zero the PTF time-out flag will be set and the register value will be loaded. The PFLT bits in the PITFLT register can be used to immediately update the count register with the new value if an immediate load is desired.

17.3.0.9 PIT Count Register 0 to 3 (PITCNT0–3)

Module Base + 0x000A, 0x000B

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

Figure 17-15. PIT Count Register 0 (PITCNT0)

Module Base + 0x000E, 0x000F

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

Figure 17-16. PIT Count Register 1 (PITCNT1)

Module Base + 0x0012, 0x0013

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

Figure 17-17. PIT Count Register 2 (PITCNT2)

Module Base + 0x0016, 0x0017

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

Figure 17-18. PIT Count Register 3 (PITCNT3)

Read: Anytime

Write: Has no meaning or effect

Table 17-9. PITCNT0–3 Field Descriptions

Field	Description
15:0 PCNT[15:0]	**PIT Count Bits 15-0** — These bits represent the current 16-bit modulus down-counter value. The read access for the count register must take place in one clock cycle as a 16-bit access.

17.4 Functional Description

Figure 17-19 shows a detailed block diagram of the PIT module. The main parts of the PIT are status, control and data registers, two 8-bit down-counters, four 16-bit down-counters and an interrupt/trigger interface.

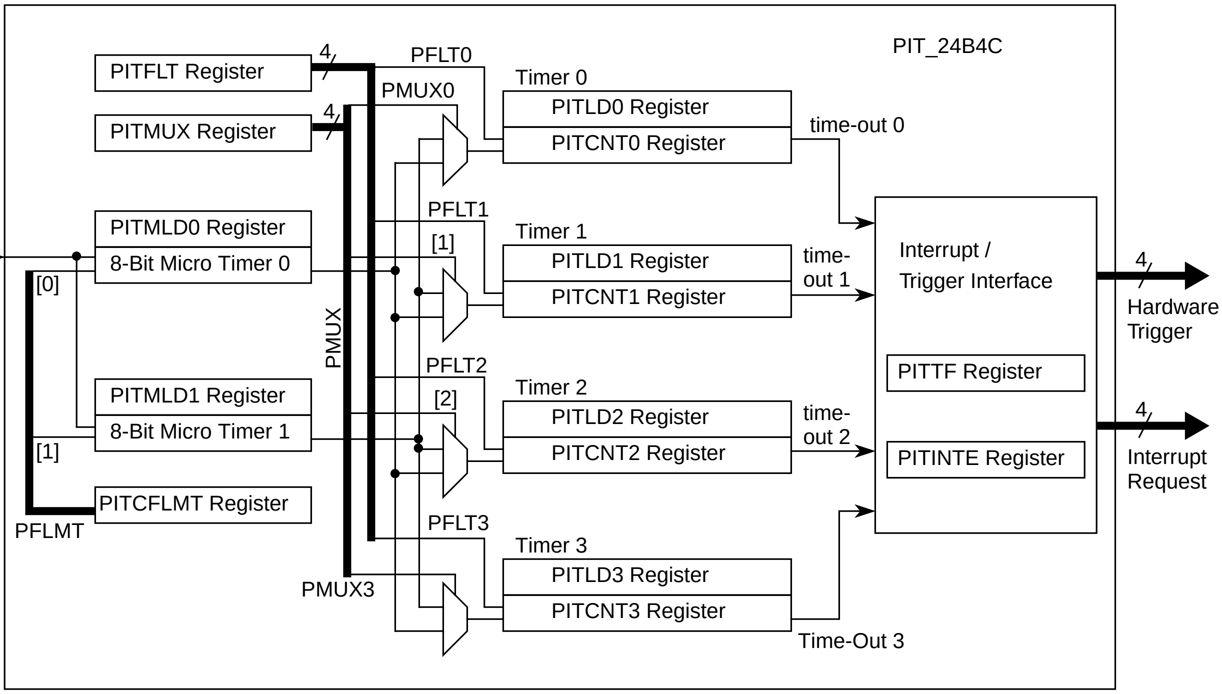


Figure 17-19. PIT Detailed Block Diagram

17.4.1 Timer

As shown in Figure 17-1and Figure 17-19, the 24-bit timers are built in a two-stage architecture with four 16-bit modulus down-counters and two 8-bit modulus down-counters. The 16-bit timers are clocked with two selectable micro time bases which are generated with 8-bit modulus down-counters. Each 16-bit timer is connected to micro time base 0 or 1 via the PMUX[3:0] bit setting in the PIT Multiplex (PITMUX) register.

A timer channel is enabled if the module enable bit PITE in the PIT control and force load micro timer (PITCFLMT) register is set and if the corresponding PCE bit in the PIT channel enable (PITCE) register is set. Two 8-bit modulus down-counters are used to generate two micro time bases. As soon as a micro time base is selected for an enabled timer channel, the corresponding micro timer modulus down-counter will load its start value as specified in the PITMTLD0 or PITMTLD1 register and will start down-counting. Whenever the micro timer down-counter has counted to zero the PITMTLD register is reloaded and the connected 16-bit modulus down-counters count one cycle.

Whenever a 16-bit timer counter and the connected 8-bit micro timer counter have counted to zero, the PITLD register is reloaded and the corresponding time-out flag PTF in the PIT time-out flag (PITTF) register is set, as shown in Figure 17-20. The time-out period is a function of the timer load (PITLD) and micro timer load (PITMTLD) registers and the bus clock f_{BUS}:

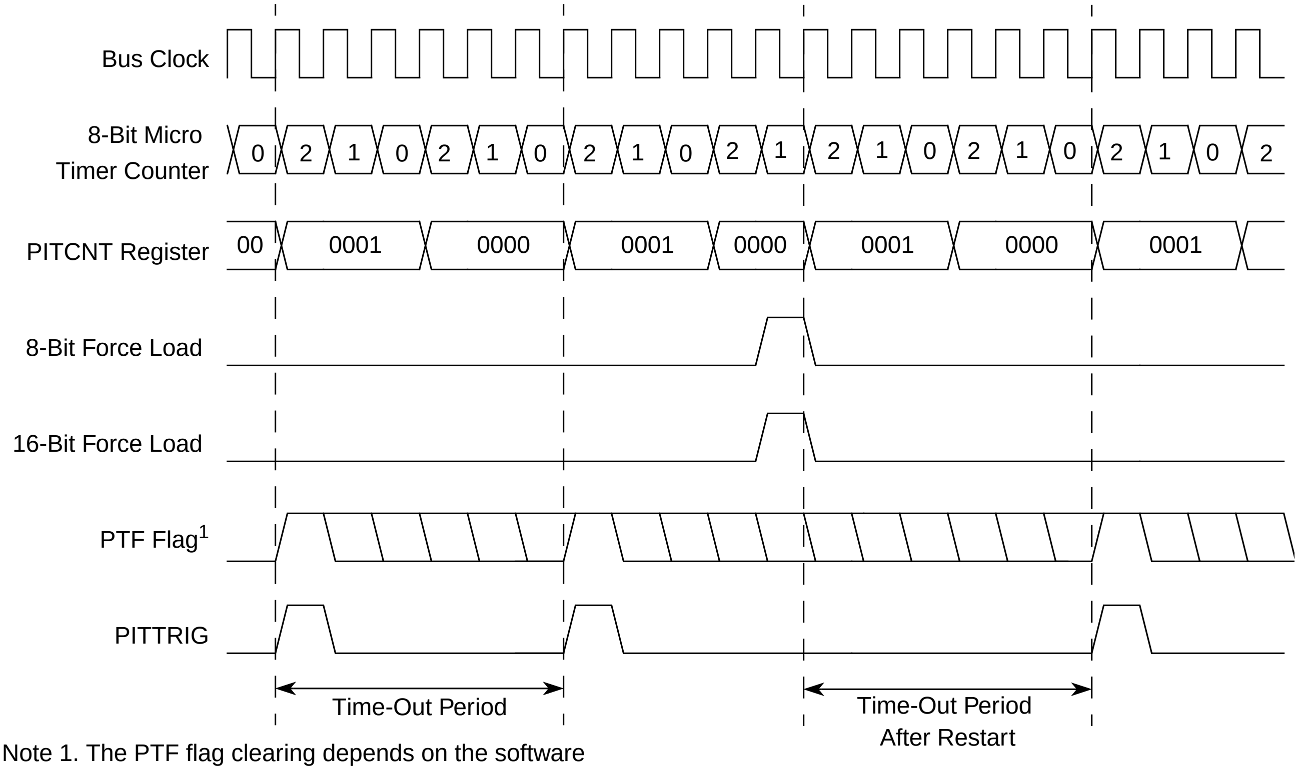
time-out period = (PITMTLD + 1) * (PITLD + 1) / f_{BUS}.

For example, for a 40 MHz bus clock, the maximum time-out period equals:

256 * 65536 * 25 ns = 419.43 ms.

The current 16-bit modulus down-counter value can be read via the PITCNT register. The micro timer down-counter values cannot be read.

The 8-bit micro timers can individually be restarted by writing a one to the corresponding force load micro timer PFLMT bits in the PIT control and force load micro timer (PITCFLMT) register. The 16-bit timers can individually be restarted by writing a one to the corresponding force load timer PFLT bits in the PIT forceload timer (PITFLT) register. If desired, any group of timers and micro timers can be restarted at the same time by using one 16-bit write to the adjacent PITCFLMT and PITFLT registers with the relevant bits set, as shown in Figure 17-20.



Note 1. The PTF flag clearing depends on the software

Figure 17-20. PIT Trigger and Flag Signal Timing

17.4.2 Interrupt Interface

Each time-out event can be used to trigger an interrupt service request. For each timer channel, an individual bit PINTE in the PIT interrupt enable (PITINTE) register exists to enable this feature. If PINTE is set, an interrupt service is requested whenever the corresponding time-out flag PTF in the PIT time-out flag (PITTF) register is set. The flag can be cleared by writing a one to the flag bit.

NOTE

> Be careful when resetting the PITE, PINTE or PITCE bits in case of pending PIT interrupt requests, to avoid spurious interrupt requests.

17.4.3 Hardware Trigger

The PIT module contains four hardware trigger signal lines PITTRIG[3:0], one for each timer channel. These signals can be connected on SoC level to peripheral modules enabling e.g. periodic ATD conversion (please refer to the SoC Guide for the mapping of PITTRIG[3:0] signals to peripheral modules).

Whenever a timer channel time-out is reached, the corresponding PTF flag is set and the corresponding trigger signal PITTRIG triggers a rising edge. The trigger feature requires a minimum time-out period of two bus clock cycles because the trigger is asserted high for at least one bus clock cycle. For load register values PITLD = 0x0001 and PITMTLD = 0x0002 the flag setting, trigger timing and a restart with force load is shown in Figure 17-20.

17.5 Initialization/Application Information

17.5.1 Startup

Set the configuration registers before the PITE bit in the PITCFLMT register is set. Before PITE is set, the configuration registers can be written in arbitrary order.

17.5.2 Shutdown

When the PITCE register bits, the PITINTE register bits or the PITE bit in the PITCFLMT register are cleared, the corresponding PIT interrupt flags are cleared. In case of a pending PIT interrupt request, a spurious interrupt can be generated. Two strategies, which avoid spurious interrupts, are recommended:

1. Reset the PIT interrupt flags only in an ISR. When entering the ISR, the I mask bit in the CCR is set automatically. The I mask bit must not be cleared before the PIT interrupt flags are cleared.
2. After setting the I mask bit with the SEI instruction, the PIT interrupt flags can be cleared. Then clear the I mask bit with the CLI instruction to re-enable interrupts.

17.5.3 Flag Clearing

A flag is cleared by writing a one to the flag bit. Always use store or move instructions to write a one in certain bit positions. Do not use the BSET instructions. Do not use any C-constructs that compile to BSET instructions. “BSET flag_register, #mask” must not be used for flag clearing because BSET is a read-modify-write instruction which writes back the “bit-wise or” of the flag_register and the mask into the flag_register. BSET would clear all flag bits that were set, independent from the mask.

For example, to clear flag bit 0 use: MOVB #$01,PITTF.

Chapter 18
Pulse-Width Modulator (S12PWM8B8CV1)

18.1 Introduction

The PWM definition is based on the HC12 PWM definitions. It contains the basic features from the HC11 with some of the enhancements incorporated on the HC12: center aligned output mode and four available clock sources.The PWM module has eight channels with independent control of left and center aligned outputs on each channel.

Each of the eight channels has a programmable period and duty cycle as well as a dedicated counter. A flexible clock select scheme allows a total of four different clock sources to be used with the counters. Each of the modulators can create independent continuous waveforms with software-selectable duty rates from 0% to 100%. The PWM outputs can be programmed as left aligned outputs or center aligned outputs.

18.1.1 Features

The PWM block includes these distinctive features:

- Eight independent PWM channels with programmable period and duty cycle
- Dedicated counter for each PWM channel
- Programmable PWM enable/disable for each channel
- Software selection of PWM duty pulse polarity for each channel
- Period and duty cycle are double buffered. Change takes effect when the end of the effective period is reached (PWM counter reaches zero) or when the channel is disabled.
- Programmable center or left aligned outputs on individual channels
- Eight 8-bit channel or four 16-bit channel PWM resolution
- Four clock sources (A, B, SA, and SB) provide for a wide range of frequencies
- Programmable clock select logic
- Emergency shutdown

18.1.2 Modes of Operation

There is a software programmable option for low power consumption in wait mode that disables the input clock to the prescaler.

In freeze mode there is a software programmable option to disable the input clock to the prescaler. This is useful for emulation.

18.1.3 Block Diagram

Figure 18-1 shows the block diagram for the 8-bit 8-channel PWM block.

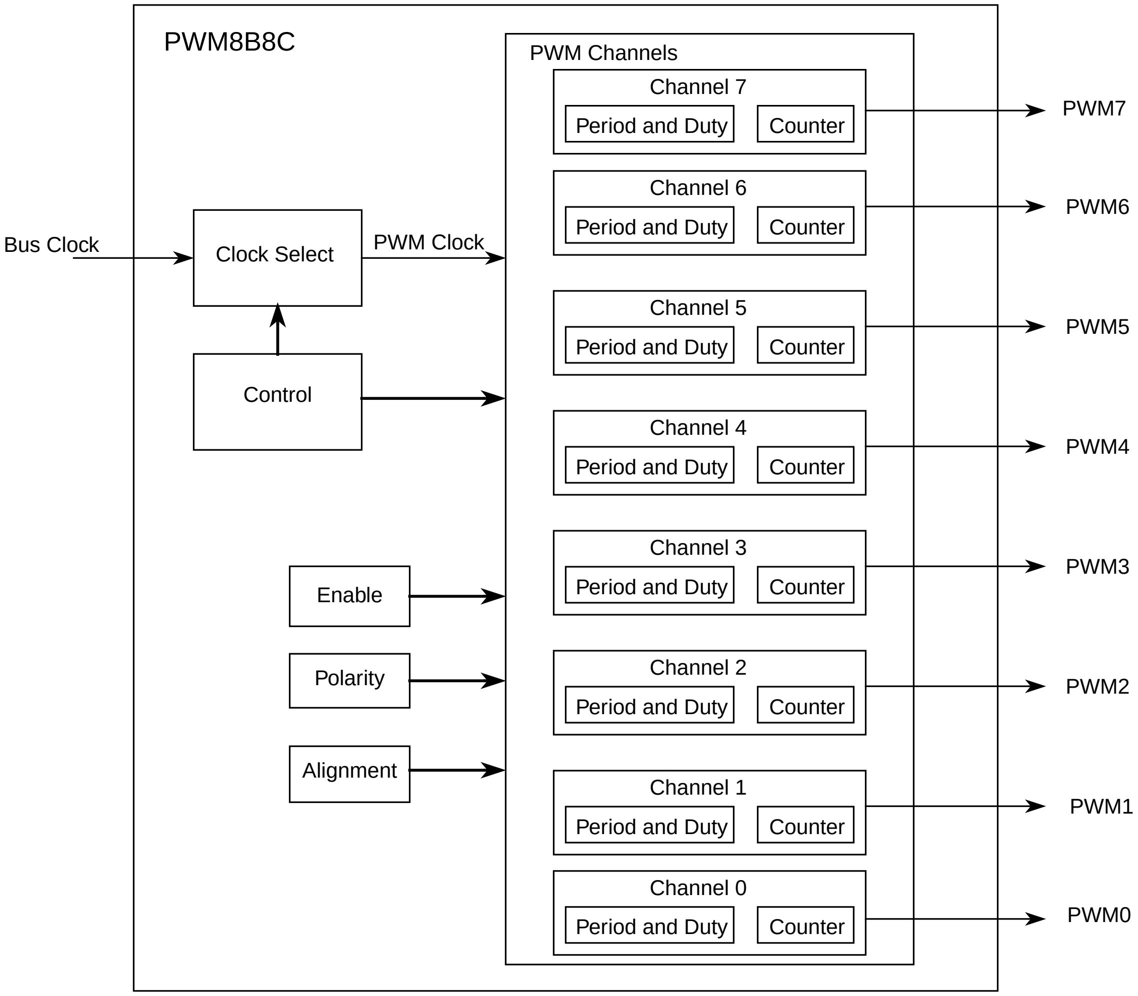


Figure 18-1. PWM Block Diagram

18.2 External Signal Description

The PWM module has a total of 8 external pins.

18.2.1 PWM7 — PWM Channel 7

This pin serves as waveform output of PWM channel 7 and as an input for the emergency shutdown feature.

18.2.2 PWM6 — PWM Channel 6

This pin serves as waveform output of PWM channel 6.

18.2.3 PWM5 — PWM Channel 5

This pin serves as waveform output of PWM channel 5.

18.2.4 PWM4 — PWM Channel 4

This pin serves as waveform output of PWM channel 4.

18.2.5 PWM3 — PWM Channel 3

This pin serves as waveform output of PWM channel 3.

18.2.6 PWM3 — PWM Channel 2

This pin serves as waveform output of PWM channel 2.

18.2.7 PWM3 — PWM Channel 1

This pin serves as waveform output of PWM channel 1.

18.2.8 PWM3 — PWM Channel 0

This pin serves as waveform output of PWM channel 0.

18.3 Memory Map and Register Definition

This section describes in detail all the registers and register bits in the PWM module.

The special-purpose registers and register bit functions that are not normally available to device end users, such as factory test control registers and reserved registers, are clearly identified by means of shading the appropriate portions of address maps and register diagrams. Notes explaining the reasons for restricting access to the registers and functions are also explained in the individual register descriptions.

18.3.1 Module Memory Map

This section describes the content of the registers in the PWM module. The base address of the PWM module is determined at the MCU level when the MCU is defined. The register decode map is fixed and begins at the first address of the module address offset. The figure below shows the registers associated with the PWM and their relative offset from the base address. The register detail description follows the order they appear in the register map.

Reserved bits within a register will always read as 0 and the write will be unimplemented. Unimplemented functions are indicated by shading the bit. .

NOTE

Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level.

18.3.2 Register Descriptions

This section describes in detail all the registers and register bits in the PWM module.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 PWME	R	PWME7	PWME6	PWME5	PWME4	PWME3	PWME2	PWME1	PWME0
	W								
0x0001 PWMPOL	R	PPOL7	PPOL6	PPOL5	PPOL4	PPOL3	PPOL2	PPOL1	PPOL0
	W								
0x0002 PWMCLK	R	PCLK7	PCLKL6	PCLK5	PCLK4	PCLK3	PCLK2	PCLK1	PCLK0
	W								
0x0003 PWMPRCLK	R	0	PCKB2	PCKB1	PCKB0	0	PCKA2	PCKA1	PCKA0
	W	▨				▨			
0x0004 PWMCAE	R	CAE7	CAE6	CAE5	CAE4	CAE3	CAE2	CAE1	CAE0
	W								
0x0005 PWMCTL	R	CON67	CON45	CON23	CON01	PSWAI	PFRZ	0	0
	W							▨	▨
0x0006 PWMTST[1]	R	0	0	0	0	0	0	0	0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0007 PWMPRSC[1]	R	0	0	0	0	0	0	0	0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0008 PWMSCLA	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x0009 PWMSCLB	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x000A PWMSCNTA[1]	R	0	0	0	0	0	0	0	0
	W	▨	▨	▨	▨	▨	▨	▨	▨

▨ = Unimplemented or Reserved

Figure 18-2. PWM Register Summary (Sheet 1 of 3)

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x000B PWMSCNTB[1]	R	0	0	0	0	0	0	0	0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x000C PWMCNT0	R	Bit 7	6	5	4	3	2	1	Bit 0
	W	0	0	0	0	0	0	0	0
0x000D PWMCNT1	R	Bit 7	6	5	4	3	2	1	Bit 0
	W	0	0	0	0	0	0	0	0
0x000E PWMCNT2	R	Bit 7	6	5	4	3	2	1	Bit 0
	W	0	0	0	0	0	0	0	0
0x000F PWMCNT3	R	Bit 7	6	5	4	3	2	1	Bit 0
	W	0	0	0	0	0	0	0	0
0x0010 PWMCNT4	R	Bit 7	6	5	4	3	2	1	Bit 0
	W	0	0	0	0	0	0	0	0
0x0011 PWMCNT5	R	Bit 7	6	5	4	3	2	1	Bit 0
	W	0	0	0	0	0	0	0	0
0x0012 PWMCNT6	R	Bit 7	6	5	4	3	2	1	Bit 0
	W	0	0	0	0	0	0	0	0
0x0013 PWMCNT7	R	Bit 7	6	5	4	3	2	1	Bit 0
	W	0	0	0	0	0	0	0	0
0x0014 PWMPER0	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x0015 PWMPER1	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x0016 PWMPER2	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x0017 PWMPER3	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x0018 PWMPER4	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x0019 PWMPER5	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								

▨ = Unimplemented or Reserved

Figure 18-2. PWM Register Summary (Sheet 2 of 3)

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x001A PWMPER6	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x001B PWMPER7	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x001C PWMDTY0	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x001D PWMDTY1	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x001E PWMDTY2	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x001F PWMDTY3	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x0010 PWMDTY4	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x0021 PWMDTY5	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x0022 PWMDTY6	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x0023 PWMDTY7	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x0024 PWMSDN	R	PWMIF	PWMIE	0	PWMLVL	0	PWM7IN	PWM7INL	PWM7ENA
	W			PWMRSTRT		▨	▨		

▨ = Unimplemented or Reserved

Figure 18-2. PWM Register Summary (Sheet 3 of 3)

---

[1] Intended for factory test purposes only.

18.3.2.1 PWM Enable Register (PWME)

Each PWM channel has an enable bit (PWMEx) to start its waveform output. When any of the PWMEx bits are set (PWMEx = 1), the associated PWM output is enabled immediately. However, the actual PWM waveform is not available on the associated PWM output until its clock source begins its next cycle due to the synchronization of PWMEx and the clock source.

NOTE

The first PWM cycle after enabling the channel can be irregular.

An exception to this is when channels are concatenated. Once concatenated mode is enabled (CONxx bits set in PWMCTL register), enabling/disabling the corresponding 16-bit PWM channel is controlled by the low order PWMEx bit.In this case, the high order bytes PWMEx bits have no effect and their corresponding PWM output lines are disabled.

While in run mode, if all eight PWM channels are disabled (PWME7–0 = 0), the prescaler counter shuts off for power savings.

Module Base + 0x0000

	7	6	5	4	3	2	1	0
R W	PWME7	PWME6	PWME5	PWME4	PWME3	PWME2	PWME1	PWME0
Reset	0	0	0	0	0	0	0	0

Figure 18-3. PWM Enable Register (PWME)

Read: Anytime

Write: Anytime

Table 18-1. PWME Field Descriptions

Field	Description
7 PWME7	**Pulse Width Channel 7 Enable** 0 Pulse width channel 7 is disabled. 1 Pulse width channel 7 is enabled. The pulse modulated signal becomes available at PWM output bit 7 when its clock source begins its next cycle.
6 PWME6	**Pulse Width Channel 6 Enable** 0 Pulse width channel 6 is disabled. 1 Pulse width channel 6 is enabled. The pulse modulated signal becomes available at PWM output bit6 when its clock source begins its next cycle. If CON67=1, then bit has no effect and PWM output line 6 is disabled.
5 PWME5	**Pulse Width Channel 5 Enable** 0 Pulse width channel 5 is disabled. 1 Pulse width channel 5 is enabled. The pulse modulated signal becomes available at PWM output bit 5 when its clock source begins its next cycle.
4 PWME4	**Pulse Width Channel 4 Enable** 0 Pulse width channel 4 is disabled. 1 Pulse width channel 4 is enabled. The pulse modulated signal becomes available at PWM, output bit 4 when its clock source begins its next cycle. If CON45 = 1, then bit has no effect and PWM output bit4 is disabled.
3 PWME3	**Pulse Width Channel 3 Enable** 0 Pulse width channel 3 is disabled. 1 Pulse width channel 3 is enabled. The pulse modulated signal becomes available at PWM, output bit 3 when its clock source begins its next cycle.
2 PWME2	**Pulse Width Channel 2 Enable** 0 Pulse width channel 2 is disabled. 1 Pulse width channel 2 is enabled. The pulse modulated signal becomes available at PWM, output bit 2 when its clock source begins its next cycle. If CON23 = 1, then bit has no effect and PWM output bit2 is disabled.

Table 18-1. PWME Field Descriptions (continued)

Field	Description
1 PWME1	**Pulse Width Channel 1 Enable** 0 Pulse width channel 1 is disabled. 1 Pulse width channel 1 is enabled. The pulse modulated signal becomes available at PWM, output bit 1 when its clock source begins its next cycle.
0 PWME0	**Pulse Width Channel 0 Enable** 0 Pulse width channel 0 is disabled. 1 Pulse width channel 0 is enabled. The pulse modulated signal becomes available at PWM, output bit 0 when its clock source begins its next cycle. If CON01 = 1, then bit has no effect and PWM output line0 is disabled.

18.3.2.2 PWM Polarity Register (PWMPOL)

The starting polarity of each PWM channel waveform is determined by the associated PPOLx bit in the PWMPOL register. If the polarity bit is one, the PWM channel output is high at the beginning of the cycle and then goes low when the duty count is reached. Conversely, if the polarity bit is zero, the output starts low and then goes high when the duty count is reached.

Module Base + 0x0001

	7	6	5	4	3	2	1	0
R W	PPOL7	PPOL6	PPOL5	PPOL4	PPOL3	PPOL2	PPOL1	PPOL0
Reset	0	0	0	0	0	0	0	0

Figure 18-4. PWM Polarity Register (PWMPOL)

Read: Anytime

Write: Anytime

NOTE

> PPOLx register bits can be written anytime. If the polarity is changed while a PWM signal is being generated, a truncated or stretched pulse can occur during the transition

Table 18-2. PWMPOL Field Descriptions

Field	Description
7–0 PPOL[7:0]	**Pulse Width Channel 7–0 Polarity Bits** 0 PWM channel 7–0 outputs are low at the beginning of the period, then go high when the duty count is reached. 1 PWM channel 7–0 outputs are high at the beginning of the period, then go low when the duty count is reached.

18.3.2.3 PWM Clock Select Register (PWMCLK)

Each PWM channel has a choice of two clocks to use as the clock source for that channel as described below.

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R W	PCLK7	PCLKL6	PCLK5	PCLK4	PCLK3	PCLK2	PCLK1	PCLK0
Reset	0	0	0	0	0	0	0	0

Figure 18-5. PWM Clock Select Register (PWMCLK)

Read: Anytime

Write: Anytime

NOTE

Register bits PCLK0 to PCLK7 can be written anytime. If a clock select is changed while a PWM signal is being generated, a truncated or stretched pulse can occur during the transition.

Table 18-3. PWMCLK Field Descriptions

Field	Description
7 PCLK7	**Pulse Width Channel 7 Clock Select** 0 Clock B is the clock source for PWM channel 7. 1 Clock SB is the clock source for PWM channel 7.
6 PCLK6	**Pulse Width Channel 6 Clock Select** 0 Clock B is the clock source for PWM channel 6. 1 Clock SB is the clock source for PWM channel 6.
5 PCLK5	**Pulse Width Channel 5 Clock Select** 0 Clock A is the clock source for PWM channel 5. 1 Clock SA is the clock source for PWM channel 5.
4 PCLK4	**Pulse Width Channel 4 Clock Select** 0 Clock A is the clock source for PWM channel 4. 1 Clock SA is the clock source for PWM channel 4.
3 PCLK3	**Pulse Width Channel 3 Clock Select** 0 Clock B is the clock source for PWM channel 3. 1 Clock SB is the clock source for PWM channel 3.
2 PCLK2	**Pulse Width Channel 2 Clock Select** 0 Clock B is the clock source for PWM channel 2. 1 Clock SB is the clock source for PWM channel 2.
1 PCLK1	**Pulse Width Channel 1 Clock Select** 0 Clock A is the clock source for PWM channel 1. 1 Clock SA is the clock source for PWM channel 1.
0 PCLK0	**Pulse Width Channel 0 Clock Select** 0 Clock A is the clock source for PWM channel 0. 1 Clock SA is the clock source for PWM channel 0.

18.3.2.4 PWM Prescale Clock Select Register (PWMPRCLK)

This register selects the prescale clock source for clocks A and B independently.

Module Base + 0x0003

	7	6	5	4	3	2	1	0
R	0	PCKB2	PCKB1	PCKB0	0	PCKA2	PCKA1	PCKA0
W	▨				▨			
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 18-6. PWM Prescale Clock Select Register (PWMPRCLK)

Read: Anytime

Write: Anytime

NOTE

PCKB2–0 and PCKA2–0 register bits can be written anytime. If the clock pre-scale is changed while a PWM signal is being generated, a truncated or stretched pulse can occur during the transition.

Table 18-4. PWMPRCLK Field Descriptions

Field	Description
6–4 PCKB[2:0]	**Prescaler Select for Clock B** — Clock B is one of two clock sources which can be used for channels 2, 3, 6, or 7. These three bits determine the rate of clock B, as shown in Table 18-5.
2–0 PCKA[2:0]	**Prescaler Select for Clock A** — Clock A is one of two clock sources which can be used for channels 0, 1, 4 or 5. These three bits determine the rate of clock A, as shown in Table 18-6.

Table 18-5. Clock B Prescaler Selects

PCKB2	PCKB1	PCKB0	Value of Clock B
0	0	0	Bus clock
0	0	1	Bus clock / 2
0	1	0	Bus clock / 4
0	1	1	Bus clock / 8
1	0	0	Bus clock / 16
1	0	1	Bus clock / 32
1	1	0	Bus clock / 64
1	1	1	Bus clock / 128

Table 18-6. Clock A Prescaler Selects

PCKA2	PCKA1	PCKA0	Value of Clock A
0	0	0	Bus clock
0	0	1	Bus clock / 2
0	1	0	Bus clock / 4
0	1	1	Bus clock / 8
1	0	0	Bus clock / 16
1	0	1	Bus clock / 32
1	1	0	Bus clock / 64
1	1	1	Bus clock / 128

18.3.2.5 PWM Center Align Enable Register (PWMCAE)

The PWMCAE register contains eight control bits for the selection of center aligned outputs or left aligned outputs for each PWM channel. If the CAEx bit is set to a one, the corresponding PWM output will be center aligned. If the CAEx bit is cleared, the corresponding PWM output will be left aligned. See Section 18.4.2.5, “Left Aligned Outputs” and Section 18.4.2.6, “Center Aligned Outputs” for a more detailed description of the PWM output modes.

Module Base + 0x0004

	7	6	5	4	3	2	1	0
R W	CAE7	CAE6	CAE5	CAE4	CAE3	CAE2	CAE1	CAE0
Reset	0	0	0	0	0	0	0	0

Figure 18-7. PWM Center Align Enable Register (PWMCAE)

Read: Anytime

Write: Anytime

NOTE

Write these bits only when the corresponding channel is disabled.

Table 18-7. PWMCAE Field Descriptions

Field	Description
7–0 CAE[7:0]	**Center Aligned Output Modes on Channels 7–0** 0 Channels 7–0 operate in left aligned output mode. 1 Channels 7–0 operate in center aligned output mode.

18.3.2.6 PWM Control Register (PWMCTL)

The PWMCTL register provides for various control of the PWM module.

Module Base + 0x0005

		7	6	5	4	3	2	1	0
R		CON67	CON45	CON23	CON01	PSWAI	PFRZ	0	0
W								▨	▨
Reset		0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 18-8. PWM Control Register (PWMCTL)

Read: Anytime

Write: Anytime

There are three control bits for concatenation, each of which is used to concatenate a pair of PWM channels into one 16-bit channel. When channels 6 and 7are concatenated, channel 6 registers become the high order bytes of the double byte channel. When channels 4 and 5 are concatenated, channel 4 registers become the high order bytes of the double byte channel. When channels 2 and 3 are concatenated, channel

2 registers become the high order bytes of the double byte channel. When channels 0 and 1 are concatenated, channel 0 registers become the high order bytes of the double byte channel.

See Section 18.4.2.7, "PWM 16-Bit Functions" for a more detailed description of the concatenation PWM Function.

NOTE

Change these bits only when both corresponding channels are disabled.

Table 18-8. PWMCTL Field Descriptions

Field	Description
7 CON67	**Concatenate Channels 6 and 7** 0 Channels 6 and 7 are separate 8-bit PWMs. 1 Channels 6 and 7 are concatenated to create one 16-bit PWM channel. Channel 6 becomes the high order byte and channel 7 becomes the low order byte. Channel 7 output pin is used as the output for this 16-bit PWM (bit 7 of port PWMP). Channel 7 clock select control-bit determines the clock source, channel 7 polarity bit determines the polarity, channel 7 enable bit enables the output and channel 7 center aligned enable bit determines the output mode.
6 CON45	**Concatenate Channels 4 and 5** 0 Channels 4 and 5 are separate 8-bit PWMs. 1 Channels 4 and 5 are concatenated to create one 16-bit PWM channel. Channel 4 becomes the high order byte and channel 5 becomes the low order byte. Channel 5 output pin is used as the output for this 16-bit PWM (bit 5 of port PWMP). Channel 5 clock select control-bit determines the clock source, channel 5 polarity bit determines the polarity, channel 5 enable bit enables the output and channel 5 center aligned enable bit determines the output mode.
5 CON23	**Concatenate Channels 2 and 3** 0 Channels 2 and 3 are separate 8-bit PWMs. 1 Channels 2 and 3 are concatenated to create one 16-bit PWM channel. Channel 2 becomes the high order byte and channel 3 becomes the low order byte. Channel 3 output pin is used as the output for this 16-bit PWM (bit 3 of port PWMP). Channel 3 clock select control-bit determines the clock source, channel 3 polarity bit determines the polarity, channel 3 enable bit enables the output and channel 3 center aligned enable bit determines the output mode.
4 CON01	**Concatenate Channels 0 and 1** 0 Channels 0 and 1 are separate 8-bit PWMs. 1 Channels 0 and 1 are concatenated to create one 16-bit PWM channel. Channel 0 becomes the high order byte and channel 1 becomes the low order byte. Channel 1 output pin is used as the output for this 16-bit PWM (bit 1 of port PWMP). Channel 1 clock select control-bit determines the clock source, channel 1 polarity bit determines the polarity, channel 1 enable bit enables the output and channel 1 center aligned enable bit determines the output mode.
3 PSWAI	**PWM Stops in Wait Mode —** Enabling this bit allows for lower power consumption in wait mode by disabling the input clock to the prescaler. 0 Allow the clock to the prescaler to continue while in wait mode. 1 Stop the input clock to the prescaler whenever the MCU is in wait mode.
2 PFREZ	**PWM Counters Stop in Freeze Mode** — In freeze mode, there is an option to disable the input clock to the prescaler by setting the PFRZ bit in the PWMCTL register. If this bit is set, whenever the MCU is in freeze mode, the input clock to the prescaler is disabled. This feature is useful during emulation as it allows the PWM function to be suspended. In this way, the counters of the PWM can be stopped while in freeze mode so that once normal program flow is continued, the counters are re-enabled to simulate real-time operations. Since the registers can still be accessed in this mode, to re-enable the prescaler clock, either disable the PFRZ bit or exit freeze mode. 0 Allow PWM to continue while in freeze mode. 1 Disable PWM input clock to the prescaler whenever the part is in freeze mode. This is useful for emulation.

18.3.2.7 Reserved Register (PWMTST)

This register is reserved for factory testing of the PWM module and is not available in normal modes.

Module Base + 0x0006

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 18-9. Reserved Register (PWMTST)

Read: Always read $00 in normal modes

Write: Unimplemented in normal modes

NOTE

Writing to this register when in special modes can alter the PWM functionality.

18.3.2.8 Reserved Register (PWMPRSC)

This register is reserved for factory testing of the PWM module and is not available in normal modes.

Module Base + 0x0007

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 18-10. Reserved Register (PWMPRSC)

Read: Always read $00 in normal modes

Write: Unimplemented in normal modes

NOTE

Writing to this register when in special modes can alter the PWM functionality.

18.3.2.9 PWM Scale A Register (PWMSCLA)

PWMSCLA is the programmable scale value used in scaling clock A to generate clock SA. Clock SA is generated by taking clock A, dividing it by the value in the PWMSCLA register and dividing that by two.

Clock SA = Clock A / (2 * PWMSCLA)

NOTE

> When PWMSCLA = $00, PWMSCLA value is considered a full scale value of 256. Clock A is thus divided by 512.

Any value written to this register will cause the scale counter to load the new scale value (PWMSCLA).

Module Base + 0x0008

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 18-11. PWM Scale A Register (PWMSCLA)

Read: Anytime

Write: Anytime (causes the scale counter to load the PWMSCLA value)

18.3.2.10 PWM Scale B Register (PWMSCLB)

PWMSCLB is the programmable scale value used in scaling clock B to generate clock SB. Clock SB is generated by taking clock B, dividing it by the value in the PWMSCLB register and dividing that by two.

Clock SB = Clock B / (2 * PWMSCLB)

NOTE

> When PWMSCLB = $00, PWMSCLB value is considered a full scale value of 256. Clock B is thus divided by 512.

Any value written to this register will cause the scale counter to load the new scale value (PWMSCLB).

Module Base + 0x0009

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 18-12. PWM Scale B Register (PWMSCLB)

Read: Anytime

Write: Anytime (causes the scale counter to load the PWMSCLB value).

18.3.2.11 Reserved Registers (PWMSCNTx)

The registers PWMSCNTA and PWMSCNTB are reserved for factory testing of the PWM module and are not available in normal modes.

Module Base + 0x000A, 0x000B

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 18-13. Reserved Registers (PWMSCNTx)

Read: Always read $00 in normal modes

Write: Unimplemented in normal modes

NOTE

Writing to these registers when in special modes can alter the PWM functionality.

18.3.2.12 PWM Channel Counter Registers (PWMCNTx)

Each channel has a dedicated 8-bit up/down counter which runs at the rate of the selected clock source. The counter can be read at any time without affecting the count or the operation of the PWM channel. In left aligned output mode, the counter counts from 0 to the value in the period register - 1. In center aligned output mode, the counter counts from 0 up to the value in the period register and then back down to 0.

Any value written to the counter causes the counter to reset to $00, the counter direction to be set to up, the immediate load of both duty and period registers with values from the buffers, and the output to change according to the polarity bit. The counter is also cleared at the end of the effective period (see Section 18.4.2.5, “Left Aligned Outputs” and Section 18.4.2.6, “Center Aligned Outputs” for more details). When the channel is disabled (PWMEx = 0), the PWMCNTx register does not count. When a channel becomes enabled (PWMEx = 1), the associated PWM counter starts at the count in the PWMCNTx register. For more detailed information on the operation of the counters, see Section 18.4.2.4, “PWM Timer Counters”.

In concatenated mode, writes to the 16-bit counter by using a 16-bit access or writes to either the low or high order byte of the counter will reset the 16-bit counter. Reads of the 16-bit counter must be made by 16-bit access to maintain data coherency.

NOTE

Writing to the counter while the channel is enabled can cause an irregular PWM cycle to occur.

Module Base + 0x000C = PWMCNT0, 0x000D = PWMCNT1, 0x000E = PWMCNT2, 0x000F = PWMCNT3
Module Base + 0x0010 = PWMCNT4, 0x0011 = PWMCNT5, 0x0012 = PWMCNT6, 0x0013 = PWMCNT7

	7	6	5	4	3	2	1	0
R	Bit 7	6	5	4	3	2	1	Bit 0
W	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0

Figure 18-14. PWM Channel Counter Registers (PWMCNTx)

Read: Anytime

Write: Anytime (any value written causes PWM counter to be reset to $00).

18.3.2.13 PWM Channel Period Registers (PWMPERx)

There is a dedicated period register for each channel. The value in this register determines the period of the associated PWM channel.

The period registers for each channel are double buffered so that if they change while the channel is enabled, the change will NOT take effect until one of the following occurs:

- The effective period ends
- The counter is written (counter resets to $00)
- The channel is disabled

In this way, the output of the PWM will always be either the old waveform or the new waveform, not some variation in between. If the channel is not enabled, then writes to the period register will go directly to the latches as well as the buffer.

NOTE

> Reads of this register return the most recent value written. Reads do not necessarily return the value of the currently active period due to the double buffering scheme.

See Section 18.4.2.3, “PWM Period and Duty” for more information.

To calculate the output period, take the selected clock source period for the channel of interest (A, B, SA, or SB) and multiply it by the value in the period register for that channel:

- Left aligned output (CAEx = 0)
- PWMx Period = Channel Clock Period * PWMPERx Center Aligned Output (CAEx = 1)

  PWMx Period = Channel Clock Period * (2 * PWMPERx)

For boundary case programming values, please refer to Section 18.4.2.8, “PWM Boundary Cases”.

Module Base + 0x0014 = PWMPER0, 0x0015 = PWMPER1, 0x0016 = PWMPER2, 0x0017 = PWMPER3
Module Base + 0x0018 = PWMPER4, 0x0019 = PWMPER5, 0x001A = PWMPER6, 0x001B = PWMPER7

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	1	1	1	1	1	1	1	1

Figure 18-15. PWM Channel Period Registers (PWMPERx)

Read: Anytime

Write: Anytime

18.3.2.14 PWM Channel Duty Registers (PWMDTYx)

There is a dedicated duty register for each channel. The value in this register determines the duty of the associated PWM channel. The duty value is compared to the counter and if it is equal to the counter value a match occurs and the output changes state.

The duty registers for each channel are double buffered so that if they change while the channel is enabled, the change will NOT take effect until one of the following occurs:

- The effective period ends
- The counter is written (counter resets to $00)
- The channel is disabled

In this way, the output of the PWM will always be either the old duty waveform or the new duty waveform, not some variation in between. If the channel is not enabled, then writes to the duty register will go directly to the latches as well as the buffer.

NOTE

> Reads of this register return the most recent value written. Reads do not necessarily return the value of the currently active duty due to the double buffering scheme.

See Section 18.4.2.3, “PWM Period and Duty” for more information.

NOTE

> Depending on the polarity bit, the duty registers will contain the count of either the high time or the low time. If the polarity bit is one, the output starts high and then goes low when the duty count is reached, so the duty registers contain a count of the high time. If the polarity bit is zero, the output starts low and then goes high when the duty count is reached, so the duty registers contain a count of the low time.

To calculate the output duty cycle (high time as a% of period) for a particular channel:

- Polarity = 0 (PPOL x =0)

  Duty Cycle = [(PWMPERx-PWMDTYx)/PWMPERx] * 100%
- Polarity = 1 (PPOLx = 1)

  Duty Cycle = [PWMDTYx / PWMPERx] * 100%

For boundary case programming values, please refer to Section 18.4.2.8, “PWM Boundary Cases”.

Module Base + 0x001C = PWMDTY0, 0x001D = PWMDTY1, 0x001E = PWMDTY2, 0x001F = PWMDTY3
Module Base + 0x0020 = PWMDTY4, 0x0021 = PWMDTY5, 0x0022 = PWMDTY6, 0x0023 = PWMDTY7

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	1	1	1	1	1	1	1	1

Figure 18-16. PWM Channel Duty Registers (PWMDTYx)

Read: Anytime

Write: Anytime

18.3.2.15 PWM Shutdown Register (PWMSDN)

The PWMSDN register provides for the shutdown functionality of the PWM module in the emergency cases. For proper operation, channel 7 must be driven to the active level for a minimum of two bus clocks.

Module Base + 0x0024

	7	6	5	4	3	2	1	0
R	PWMIF	PWMIE	0	PWMLVL	0	PWM7IN	PWM7INL	PWM7ENA
W			PWMRSTRT		▨	▨		
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 18-17. PWM Shutdown Register (PWMSDN)

Read: Anytime

Write: Anytime

Table 18-9. PWMSDN Field Descriptions

Field	Description
7 PWMIF	**PWM Interrupt Flag** — Any change from passive to asserted (active) state or from active to passive state will be flagged by setting the PWMIF flag = 1. The flag is cleared by writing a logic 1 to it. Writing a 0 has no effect. 0 No change on PWM7IN input. 1 Change on PWM7IN input
6 PWMIE	**PWM Interrupt Enable** — If interrupt is enabled an interrupt to the CPU is asserted. 0 PWM interrupt is disabled. 1 PWM interrupt is enabled.
5 PWMRSTRT	**PWM Restart** — The PWM can only be restarted if the PWM channel input 7 is de-asserted. After writing a logic 1 to the PWMRSTRT bit (trigger event) the PWM channels start running after the corresponding counter passes next "counter == 0" phase. Also, if the PWM7ENA bit is reset to 0, the PWM do not start before the counter passes $00. The bit is always read as "0".
4 PWMLVL	**PWM Shutdown Output Level** If active level as defined by the PWM7IN input, gets asserted all enabled PWM channels are immediately driven to the level defined by PWMLVL. 0 PWM outputs are forced to 0 1 Outputs are forced to 1.
2 PWM7IN	**PWM Channel 7 Input Status** — This reflects the current status of the PWM7 pin.
1 PWM7INL	**PWM Shutdown Active Input Level for Channel 7** — If the emergency shutdown feature is enabled (PWM7ENA = 1), this bit determines the active level of the PWM7channel. 0 Active level is low 1 Active level is high
0 PWM7ENA	**PWM Emergency Shutdown Enable** — If this bit is logic 1, the pin associated with channel 7 is forced to input and the emergency shutdown feature is enabled. All the other bits in this register are meaningful only if PWM7ENA = 1. 0 PWM emergency feature disabled. 1 PWM emergency feature is enabled.

18.4 Functional Description

18.4.1 PWM Clock Select

There are four available clocks: clock A, clock B, clock SA (scaled A), and clock SB (scaled B). These four clocks are based on the bus clock.

Clock A and B can be software selected to be 1, 1/2, 1/4, 1/8,..., 1/64, 1/128 times the bus clock. Clock SA uses clock A as an input and divides it further with a reloadable counter. Similarly, clock SB uses clock B as an input and divides it further with a reloadable counter. The rates available for clock SA are software selectable to be clock A divided by 2, 4, 6, 8,..., or 512 in increments of divide by 2. Similar rates are available for clock SB. Each PWM channel has the capability of selecting one of two clocks, either the pre-scaled clock (clock A or B) or the scaled clock (clock SA or SB).

The block diagram in Figure 18-18 shows the four different clocks and how the scaled clocks are created.

18.4.1.1 Prescale

The input clock to the PWM prescaler is the bus clock. It can be disabled whenever the part is in freeze mode by setting the PFRZ bit in the PWMCTL register. If this bit is set, whenever the MCU is in freeze mode (freeze mode signal active) the input clock to the prescaler is disabled. This is useful for emulation in order to freeze the PWM. The input clock can also be disabled when all eight PWM channels are disabled (PWME7-0 = 0). This is useful for reducing power by disabling the prescale counter.

Clock A and clock B are scaled values of the input clock. The value is software selectable for both clock A and clock B and has options of 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, or 1/128 times the bus clock. The value selected for clock A is determined by the PCKA2, PCKA1, PCKA0 bits in the PWMPRCLK register. The value selected for clock B is determined by the PCKB2, PCKB1, PCKB0 bits also in the PWMPRCLK register.

18.4.1.2 Clock Scale

The scaled A clock uses clock A as an input and divides it further with a user programmable value and then divides this by 2. The scaled B clock uses clock B as an input and divides it further with a user programmable value and then divides this by 2. The rates available for clock SA are software selectable to be clock A divided by 2, 4, 6, 8,..., or 512 in increments of divide by 2. Similar rates are available for clock SB.

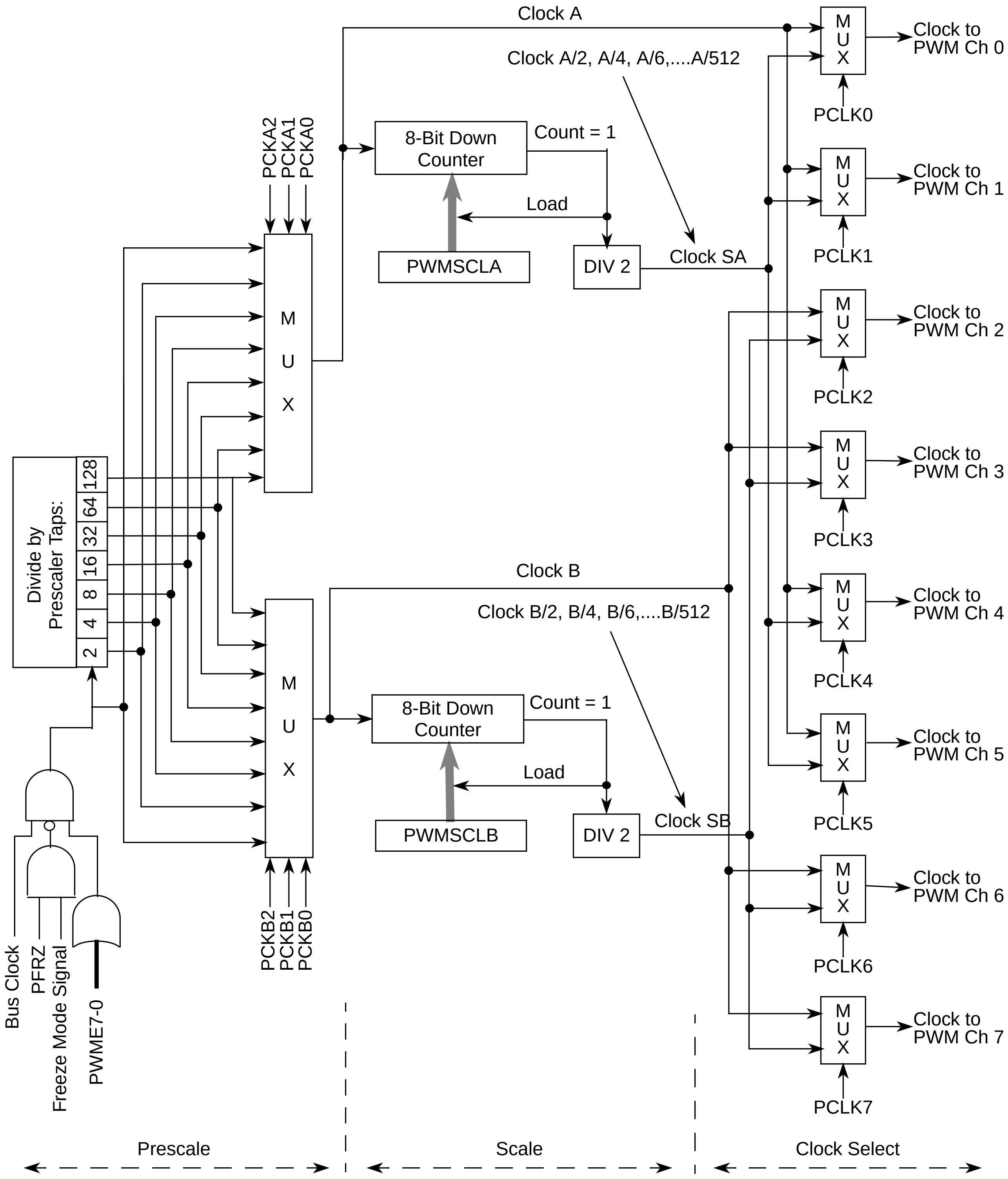


Figure 18-18. PWM Clock Select Block Diagram

Clock A is used as an input to an 8-bit down counter. This down counter loads a user programmable scale value from the scale register (PWMSCLA). When the down counter reaches one, a pulse is output and the 8-bit counter is re-loaded. The output signal from this circuit is further divided by two. This gives a greater range with only a slight reduction in granularity. Clock SA equals clock A divided by two times the value in the PWMSCLA register.

NOTE

Clock SA = Clock A / (2 * PWMSCLA)

When PWMSCLA = $00, PWMSCLA value is considered a full scale value of 256. Clock A is thus divided by 512.

Similarly, clock B is used as an input to an 8-bit down counter followed by a divide by two producing clock SB. Thus, clock SB equals clock B divided by two times the value in the PWMSCLB register.

NOTE

Clock SB = Clock B / (2 * PWMSCLB)

When PWMSCLB = $00, PWMSCLB value is considered a full scale value of 256. Clock B is thus divided by 512.

As an example, consider the case in which the user writes $FF into the PWMSCLA register. Clock A for this case will be E divided by 4. A pulse will occur at a rate of once every 255x4 E cycles. Passing this through the divide by two circuit produces a clock signal at an E divided by 2040 rate. Similarly, a value of $01 in the PWMSCLA register when clock A is E divided by 4 will produce a clock at an E divided by 8 rate.

Writing to PWMSCLA or PWMSCLB causes the associated 8-bit down counter to be re-loaded. Otherwise, when changing rates the counter would have to count down to $01 before counting at the proper rate. Forcing the associated counter to re-load the scale register value every time PWMSCLA or PWMSCLB is written prevents this.

NOTE

Writing to the scale registers while channels are operating can cause irregularities in the PWM outputs.

18.4.1.3 Clock Select

Each PWM channel has the capability of selecting one of two clocks. For channels 0, 1, 4, and 5 the clock choices are clock A or clock SA. For channels 2, 3, 6, and 7 the choices are clock B or clock SB. The clock selection is done with the PCLKx control bits in the PWMCLK register.

NOTE

Changing clock control bits while channels are operating can cause irregularities in the PWM outputs.

18.4.2 PWM Channel Timers

The main part of the PWM module are the actual timers. Each of the timer channels has a counter, a period register and a duty register (each are 8-bit). The waveform output period is controlled by a match between the period register and the value in the counter. The duty is controlled by a match between the duty register and the counter value and causes the state of the output to change during the period. The starting polarity of the output is also selectable on a per channel basis. Shown below in Figure 18-19 is the block diagram for the PWM timer.

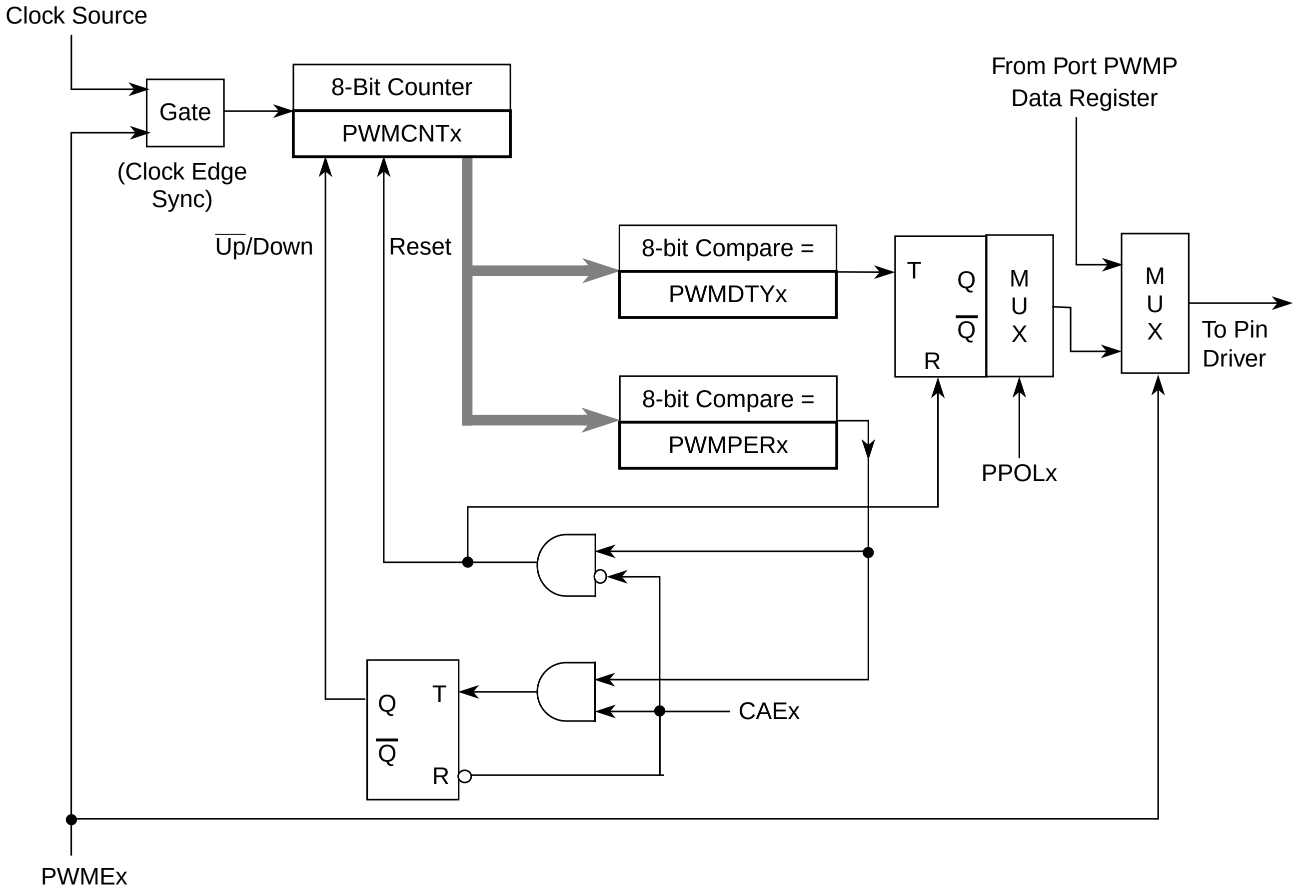


Figure 18-19. PWM Timer Channel Block Diagram

18.4.2.1 PWM Enable

Each PWM channel has an enable bit (PWMEx) to start its waveform output. When any of the PWMEx bits are set (PWMEx = 1), the associated PWM output signal is enabled immediately. However, the actual PWM waveform is not available on the associated PWM output until its clock source begins its next cycle due to the synchronization of PWMEx and the clock source. An exception to this is when channels are concatenated. Refer to Section 18.4.2.7, “PWM 16-Bit Functions” for more detail.

NOTE

The first PWM cycle after enabling the channel can be irregular.

On the front end of the PWM timer, the clock is enabled to the PWM circuit by the PWMEx bit being high. There is an edge-synchronizing circuit to guarantee that the clock will only be enabled or disabled at an edge. When the channel is disabled (PWMEx = 0), the counter for the channel does not count.

18.4.2.2 PWM Polarity

Each channel has a polarity bit to allow starting a waveform cycle with a high or low signal. This is shown on the block diagram as a mux select of either the Q output or the $\overline{Q}$ output of the PWM output flip flop. When one of the bits in the PWMPOL register is set, the associated PWM channel output is high at the beginning of the waveform, then goes low when the duty count is reached. Conversely, if the polarity bit is zero, the output starts low and then goes high when the duty count is reached.

18.4.2.3 PWM Period and Duty

Dedicated period and duty registers exist for each channel and are double buffered so that if they change while the channel is enabled, the change will NOT take effect until one of the following occurs:

- The effective period ends
- The counter is written (counter resets to $00)
- The channel is disabled

In this way, the output of the PWM will always be either the old waveform or the new waveform, not some variation in between. If the channel is not enabled, then writes to the period and duty registers will go directly to the latches as well as the buffer.

A change in duty or period can be forced into effect “immediately” by writing the new value to the duty and/or period registers and then writing to the counter. This forces the counter to reset and the new duty and/or period values to be latched. In addition, since the counter is readable, it is possible to know where the count is with respect to the duty value and software can be used to make adjustments

NOTE

When forcing a new period or duty into effect immediately, an irregular PWM cycle can occur.

Depending on the polarity bit, the duty registers will contain the count of either the high time or the low time.

18.4.2.4 PWM Timer Counters

Each channel has a dedicated 8-bit up/down counter which runs at the rate of the selected clock source (see Section 18.4.1, “PWM Clock Select” for the available clock sources and rates). The counter compares to two registers, a duty register and a period register as shown in Figure 18-19. When the PWM counter matches the duty register, the output flip-flop changes state, causing the PWM waveform to also change state. A match between the PWM counter and the period register behaves differently depending on what output mode is selected as shown in Figure 18-19 and described in Section 18.4.2.5, “Left Aligned Outputs” and Section 18.4.2.6, “Center Aligned Outputs”.

Each channel counter can be read at anytime without affecting the count or the operation of the PWM channel.

Any value written to the counter causes the counter to reset to $00, the counter direction to be set to up, the immediate load of both duty and period registers with values from the buffers, and the output to change according to the polarity bit. When the channel is disabled (PWMEx = 0), the counter stops. When a channel becomes enabled (PWMEx = 1), the associated PWM counter continues from the count in the PWMCNTx register. This allows the waveform to continue where it left off when the channel is re-enabled. When the channel is disabled, writing "0" to the period register will cause the counter to reset on the next selected clock.

NOTE

If the user wants to start a new "clean" PWM waveform without any "history" from the old waveform, the user must write to channel counter (PWMCNTx) prior to enabling the PWM channel (PWMEx = 1).

Generally, writes to the counter are done prior to enabling a channel in order to start from a known state. However, writing a counter can also be done while the PWM channel is enabled (counting). The effect is similar to writing the counter when the channel is disabled, except that the new period is started immediately with the output set according to the polarity bit.

NOTE

Writing to the counter while the channel is enabled can cause an irregular PWM cycle to occur.

The counter is cleared at the end of the effective period (see Section 18.4.2.5, "Left Aligned Outputs" and Section 18.4.2.6, "Center Aligned Outputs" for more details).

Table 18-10. PWM Timer Counter Conditions

Counter Clears ($00)	Counter Counts	Counter Stops
When PWMCNTx register written to any value	When PWM channel is enabled (PWMEx = 1). Counts from last value in PWMCNTx.	When PWM channel is disabled (PWMEx = 0)
Effective period ends		

18.4.2.5 Left Aligned Outputs

The PWM timer provides the choice of two types of outputs, left aligned or center aligned. They are selected with the CAEx bits in the PWMCAE register. If the CAEx bit is cleared (CAEx = 0), the corresponding PWM output will be left aligned.

In left aligned output mode, the 8-bit counter is configured as an up counter only. It compares to two registers, a duty register and a period register as shown in the block diagram in Figure 18-19. When the PWM counter matches the duty register the output flip-flop changes state causing the PWM waveform to also change state. A match between the PWM counter and the period register resets the counter and the output flip-flop, as shown in Figure 18-19, as well as performing a load from the double buffer period and duty register to the associated registers, as described in Section 18.4.2.3, "PWM Period and Duty". The counter counts from 0 to the value in the period register – 1.

NOTE

Changing the PWM output mode from left aligned to center aligned output (or vice versa) while channels are operating can cause irregularities in the PWM output. It is recommended to program the output mode before enabling the PWM channel.

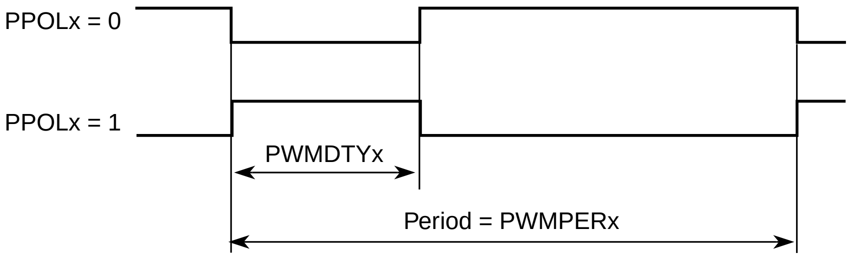


Figure 18-20. PWM Left Aligned Output Waveform

To calculate the output frequency in left aligned output mode for a particular channel, take the selected clock source frequency for the channel (A, B, SA, or SB) and divide it by the value in the period register for that channel.

- PWMx Frequency = Clock (A, B, SA, or SB) / PWMPERx
- PWMx Duty Cycle (high time as a% of period):
  - Polarity = 0 (PPOLx = 0)
- Duty Cycle = [(PWMPERx-PWMDTYx)/PWMPERx] * 100%
  - Polarity = 1 (PPOLx = 1)

Duty Cycle = [PWMDTYx / PWMPERx] * 100%

As an example of a left aligned output, consider the following case:

Clock Source = E, where E = 10 MHz (100 ns period)
PPOLx = 0
PWMPERx = 4
PWMDTYx = 1
PWMx Frequency = 10 MHz/4 = 2.5 MHz
PWMx Period = 400 ns
PWMx Duty Cycle = 3/4 *100% = 75%

The output waveform generated is shown in Figure 18-21.

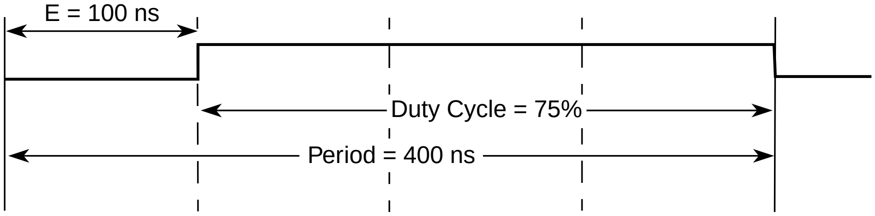


Figure 18-21. PWM Left Aligned Output Example Waveform

18.4.2.6 Center Aligned Outputs

For center aligned output mode selection, set the CAEx bit (CAEx = 1) in the PWMCAE register and the corresponding PWM output will be center aligned.

The 8-bit counter operates as an up/down counter in this mode and is set to up whenever the counter is equal to $00. The counter compares to two registers, a duty register and a period register as shown in the block diagram in Figure 18-19. When the PWM counter matches the duty register, the output flip-flop changes state, causing the PWM waveform to also change state. A match between the PWM counter and the period register changes the counter direction from an up-count to a down-count. When the PWM counter decrements and matches the duty register again, the output flip-flop changes state causing the PWM output to also change state. When the PWM counter decrements and reaches zero, the counter direction changes from a down-count back to an up-count and a load from the double buffer period and duty registers to the associated registers is performed, as described in Section 18.4.2.3, “PWM Period and Duty”. The counter counts from 0 up to the value in the period register and then back down to 0. Thus the effective period is PWMPERx*2.

NOTE

Changing the PWM output mode from left aligned to center aligned output (or vice versa) while channels are operating can cause irregularities in the PWM output. It is recommended to program the output mode before enabling the PWM channel.

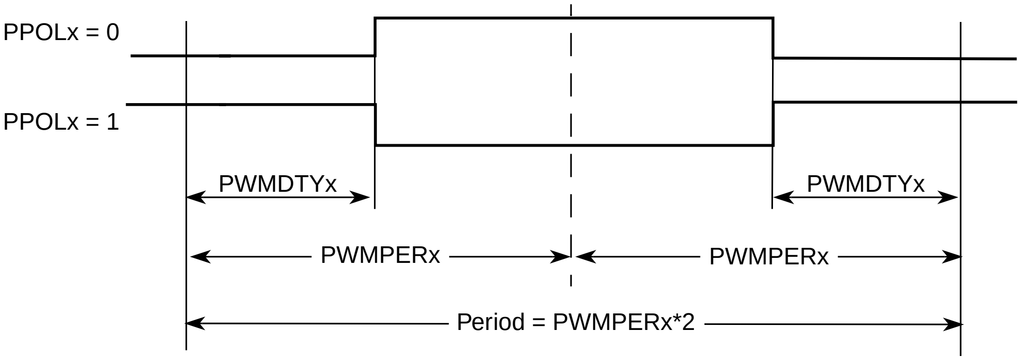


Figure 18-22. PWM Center Aligned Output Waveform

To calculate the output frequency in center aligned output mode for a particular channel, take the selected clock source frequency for the channel (A, B, SA, or SB) and divide it by twice the value in the period register for that channel.

- PWMx Frequency = Clock (A, B, SA, or SB) / (2*PWMPERx)
- PWMx Duty Cycle (high time as a% of period):
  - Polarity = 0 (PPOLx = 0)

    Duty Cycle = [(PWMPERx-PWMDTYx)/PWMPERx] * 100%
  - Polarity = 1 (PPOLx = 1)

    Duty Cycle = [PWMDTYx / PWMPERx] * 100%

As an example of a center aligned output, consider the following case:

Clock Source = E, where E = 10 MHz (100 ns period)
PPOLx = 0
PWMPERx = 4
PWMDTYx = 1
PWMx Frequency = 10 MHz/8 = 1.25 MHz
PWMx Period = 800 ns
PWMx Duty Cycle = 3/4 *100% = 75%

Shown in Figure 18-23 is the output waveform generated.

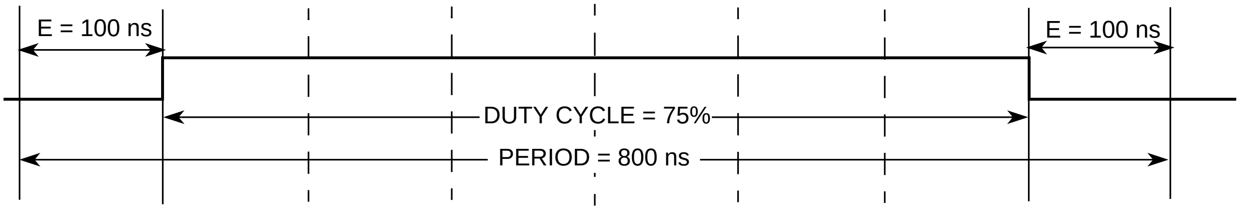


Figure 18-23. PWM Center Aligned Output Example Waveform

18.4.2.7 PWM 16-Bit Functions

The PWM timer also has the option of generating 8-channels of 8-bits or 4-channels of 16-bits for greater PWM resolution. This 16-bit channel option is achieved through the concatenation of two 8-bit channels.

The PWMCTL register contains four control bits, each of which is used to concatenate a pair of PWM channels into one 16-bit channel. Channels 6 and 7 are concatenated with the CON67 bit, channels 4 and 5 are concatenated with the CON45 bit, channels 2 and 3 are concatenated with the CON23 bit, and channels 0 and 1 are concatenated with the CON01 bit.

NOTE

Change these bits only when both corresponding channels are disabled.

When channels 6 and 7 are concatenated, channel 6 registers become the high order bytes of the double byte channel, as shown in Figure 18-24. Similarly, when channels 4 and 5 are concatenated, channel 4 registers become the high order bytes of the double byte channel. When channels 2 and 3 are concatenated, channel 2 registers become the high order bytes of the double byte channel. When channels 0 and 1 are concatenated, channel 0 registers become the high order bytes of the double byte channel.

When using the 16-bit concatenated mode, the clock source is determined by the low order 8-bit channel clock select control bits. That is channel 7 when channels 6 and 7 are concatenated, channel 5 when channels 4 and 5 are concatenated, channel 3 when channels 2 and 3 are concatenated, and channel 1 when channels 0 and 1 are concatenated. The resulting PWM is output to the pins of the corresponding low order 8-bit channel as also shown in Figure 18-24. The polarity of the resulting PWM output is controlled by the PPOLx bit of the corresponding low order 8-bit channel as well.

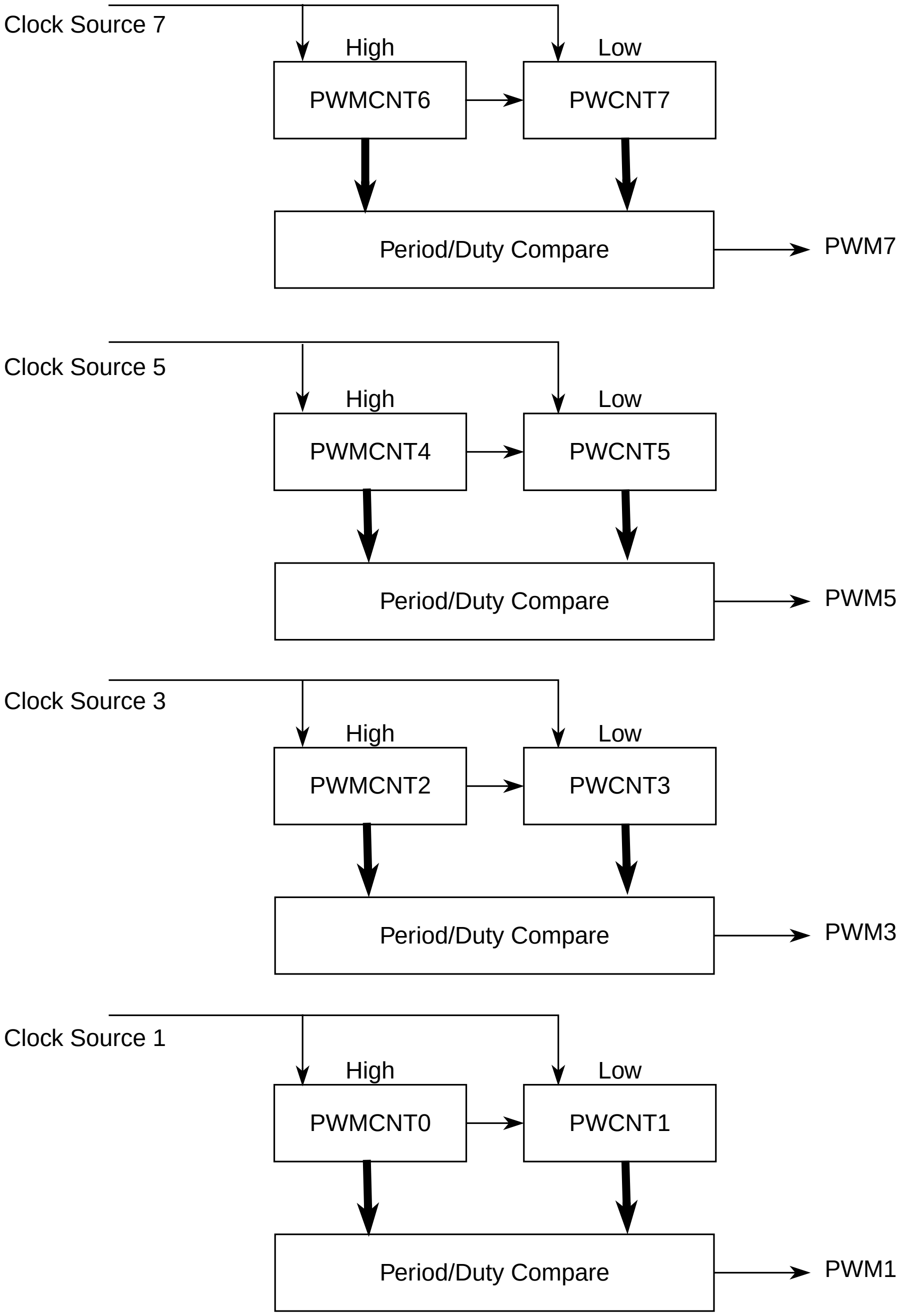


Figure 18-24. PWM 16-Bit Mode

Once concatenated mode is enabled (CONxx bits set in PWMCTL register), enabling/disabling the corresponding 16-bit PWM channel is controlled by the low order PWMEx bit. In this case, the high order bytes PWMEx bits have no effect and their corresponding PWM output is disabled.

In concatenated mode, writes to the 16-bit counter by using a 16-bit access or writes to either the low or high order byte of the counter will reset the 16-bit counter. Reads of the 16-bit counter must be made by 16-bit access to maintain data coherency.

Either left aligned or center aligned output mode can be used in concatenated mode and is controlled by the low order CAEx bit. The high order CAEx bit has no effect.

Table 18-11 is used to summarize which channels are used to set the various control bits when in 16-bit mode.

Table 18-11. 16-bit Concatenation Mode Summary

CONxx	PWMEx	PPOLx	PCLKx	CAEx	PWMx Output
CON67	PWME7	PPOL7	PCLK7	CAE7	PWM7
CON45	PWME5	PPOL5	PCLK5	CAE5	PWM5
CON23	PWME3	PPOL3	PCLK3	CAE3	PWM3
CON01	PWME1	PPOL1	PCLK1	CAE1	PWM1

18.4.2.8 PWM Boundary Cases

Table 18-12 summarizes the boundary conditions for the PWM regardless of the output mode (left aligned or center aligned) and 8-bit (normal) or 16-bit (concatenation).

Table 18-12. PWM Boundary Cases

PWMDTYx	PWMPERx	PPOLx	PWMx Output
\$00 (indicates no duty)	>\$00	1	Always low
\$00 (indicates no duty)	>\$00	0	Always high
XX	\$00[1] (indicates no period)	1	Always high
XX	\$00[1] (indicates no period)	0	Always low
>= PWMPERx	XX	1	Always high
>= PWMPERx	XX	0	Always low

[1] Counter = \$00 and does not count.

18.5 Resets

The reset state of each individual bit is listed within the Section 18.3.2, “Register Descriptions” which details the registers and their bit-fields. All special functions or modes which are initialized during or just following reset are described within this section.

- The 8-bit up/down counter is configured as an up counter out of reset.
- All the channels are disabled and all the counters do not count.

18.6 Interrupts

The PWM module has only one interrupt which is generated at the time of emergency shutdown, if the corresponding enable bit (PWMIE) is set. This bit is the enable for the interrupt. The interrupt flag PWMIF is set whenever the input level of the PWM7 channel changes while PWM7ENA = 1 or when PWMENA is being asserted while the level at PWM7 is active.

In stop mode or wait mode (with the PSWAI bit set), the emergency shutdown feature will drive the PWM outputs to their shutdown output levels but the PWMIF flag will not be set.

A description of the registers involved and affected due to this interrupt is explained in Section 18.3.2.15, “PWM Shutdown Register (PWMSDN)”.

The PWM block only generates the interrupt and does not service it. The interrupt signal name is PWM interrupt signal.

Chapter 19
Enhanced Capture Timer (ECT16B8CV3)

Table 19-1. Revision History

Revision Number	Revision Date	Sections Affected	Description of Changes
V03.06	05 Aug 2009	19.3.2.15/19-693 19.3.2.16/19-695 19.3.2.24/19-701 19.3.2.29/19-706 19.4.1.1.2/19-717	update register PACTL bit4 PEDGE PT7 to IC7 update register PAFLG bit0 PAIF PT7 to IC7,update bit1 PAOVF PT3 to IC3 update register ICSYS bit3 TFMOD PTx to ICx update register PBFLG bit1 PBOVF PT1 to IC1 update IC Queue Mode description.
V03.07	26 Aug 2009	19.3.2.2/19-680 19.3.2.3/19-680 19.3.2.4/19-681	- Add description, ?a counter overflow when TTOV[7] is set?, to be the condition of channel 7 override event. - Phrase the description of OC7M to make it more explicit
V03.08	04 May 2010	19.3.2.8/19-684 19.3.2.11/19-687	- Add Table 19-11 - TCRE descripgion part,add Note and Figure 19-17

19.1 Introduction

The HCS12 enhanced capture timer module has the features of the HCS12 standard timer module enhanced by additional features in order to enlarge the field of applications, in particular for automotive ABS applications.

This design specification describes the standard timer as well as the additional features.

The basic timer consists of a 16-bit, software-programmable counter driven by a prescaler. This timer can be used for many purposes, including input waveform measurements while simultaneously generating an output waveform. Pulse widths can vary from microseconds to many seconds.

A full access for the counter registers or the input capture/output compare registers will take place in one clock cycle. Accessing high byte and low byte separately for all of these registers will not yield the same result as accessing them in one word.

19.1.1 Features

- 16-bit buffer register for four input capture (IC) channels.

- Four 8-bit pulse accumulators with 8-bit buffer registers associated with the four buffered IC channels. Configurable also as two 16-bit pulse accumulators.
- 16-bit modulus down-counter with 8-bit prescaler.
- Four user-selectable delay counters for input noise immunity increase.

19.1.2 Modes of Operation

- Stop — Timer and modulus counter are off since clocks are stopped.
- Freeze — Timer and modulus counter keep on running, unless the TSFRZ bit in the TSCR1 register is set to one.
- Wait — Counters keep on running, unless the TSWAI bit in the TSCR1 register is set to one.
- Normal — Timer and modulus counter keep on running, unless the TEN bit in the TSCR1 register or the MCEN bit in the MCCTL register are cleared.

19.1.3 Block Diagram

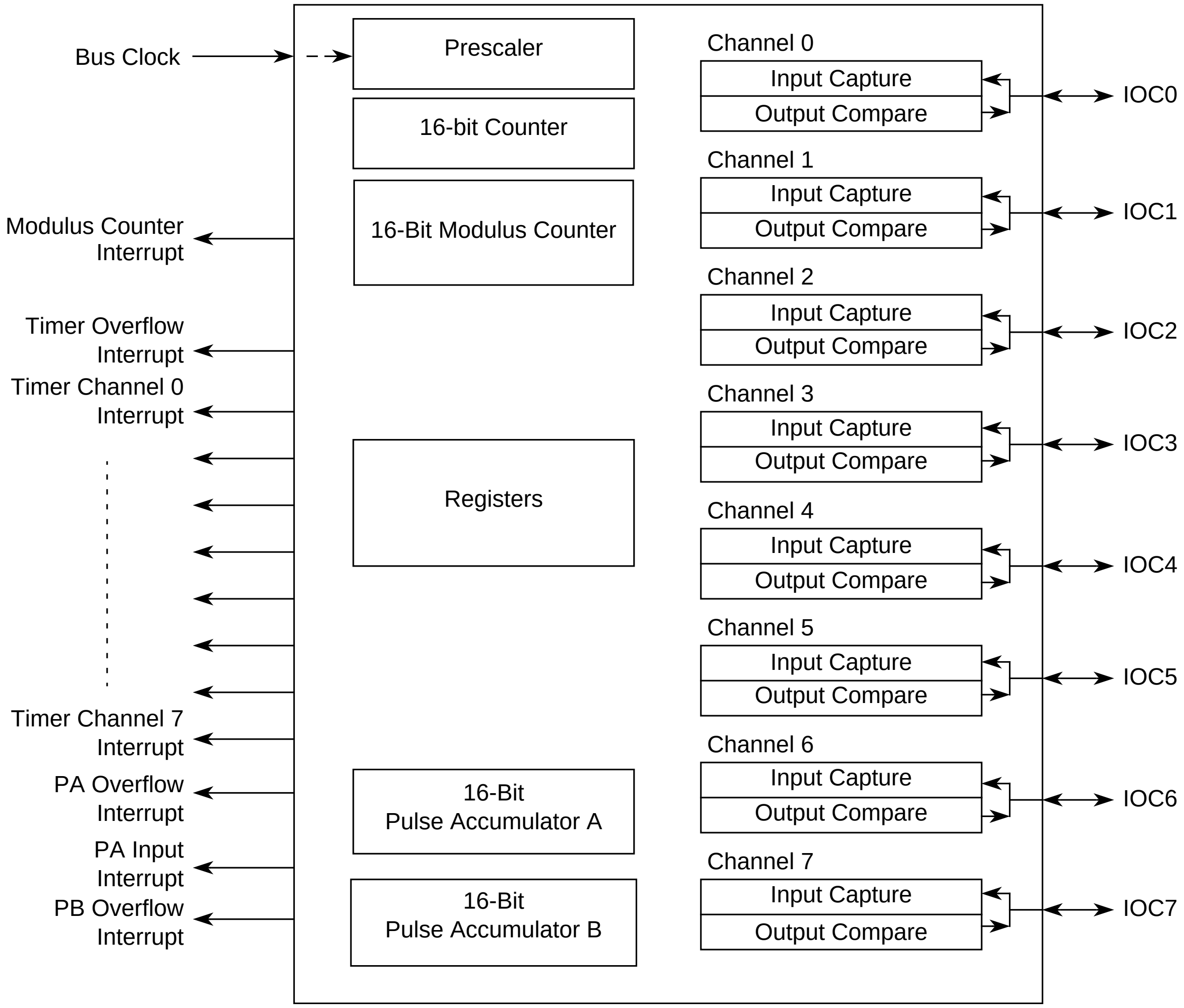


Figure 19-1. ECT Block Diagram

19.2 External Signal Description

The ECT module has a total of eight external pins.

19.2.1 IOC7 — Input Capture and Output Compare Channel 7

This pin serves as input capture or output compare for channel 7.

19.2.2 IOC6 — Input Capture and Output Compare Channel 6

This pin serves as input capture or output compare for channel 6.

19.2.3 IOC5 — Input Capture and Output Compare Channel 5

This pin serves as input capture or output compare for channel 5.

19.2.4 IOC4 — Input Capture and Output Compare Channel 4

This pin serves as input capture or output compare for channel 4.

19.2.5 IOC3 — Input Capture and Output Compare Channel 3

This pin serves as input capture or output compare for channel 3.

19.2.6 IOC2 — Input Capture and Output Compare Channel 2

This pin serves as input capture or output compare for channel 2.

19.2.7 IOC1 — Input Capture and Output Compare Channel 1

This pin serves as input capture or output compare for channel 1.

19.2.8 IOC0 — Input Capture and Output Compare Channel 0

This pin serves as input capture or output compare for channel 0.

NOTE

For the description of interrupts see Section 19.4.3, “Interrupts”.

19.3 Memory Map and Register Definition

This section provides a detailed description of all memory and registers.

19.3.1 Module Memory Map

The memory map for the ECT module is given below in the Table 19-2. The address listed for each register is the address offset. The total address for each register is the sum of the base address for the ECT module and the address offset for each register.

19.3.2 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 TIOS	R	IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0
	W								
0x0001 CFORC	R	0	0	0	0	0	0	0	0
	W	FOC7	FOC6	FOC5	FOC4	FOC3	FOC2	FOC1	FOC0
0x0002 OC7M	R	OC7M7	OC7M6	OC7M5	OC7M4	OC7M3	OC7M2	OC7M1	OC7M0
	W								
0x0003 OC7D	R	OC7D7	OC7D6	OC7D5	OC7D4	OC7D3	OC7D2	OC7D1	OC7D0
	W								
0x0004 TCNT (High)	R	TCNT15	TCNT14	TCNT13	TCNT12	TCNT11	TCNT10	TCNT9	TCNT8
	W								
0x0005 TCNT (Low)	R	TCNT7	TCNT6	TCNT5	TCNT4	TCNT3	TCNT2	TCNT1	TCNT0
	W								
0x0006 TSCR1	R	TEN	TSWAI	TSFRZ	TFFCA	PRNT	0	0	0
	W						▨	▨	▨
0x0007 TTOF	R	TOV7	TOV6	TOV5	TOV4	TOV3	TOV2	TOV1	TOV0
	W								
0x0008 TCTL1	R	OM7	OL7	OM6	OL6	OM5	OL5	OM4	OL4
	W								
0x0009 TCTL2	R	OM3	OL3	OM2	OL2	OM1	OL1	OM0	OL0
	W								
0x000A TCTL3	R	EDG7B	EDG7A	EDG6B	EDG6A	EDG5B	EDG5A	EDG4B	EDG4A
	W								
0x000B TCTL4	R	EDG3B	EDG3A	EDG2B	EDG2A	EDG1B	EDG1A	EDG0B	EDG0A
	W								
0x000C TIE	R	C7I	C6I	C5I	C4I	C3I	C2I	C1I	C0I
	W								

▨ = Unimplemented or Reserved

Figure 19-2. ECT Register Summary (Sheet 1 of 5)

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x000D TSCR2	R	TOI	0	0	0	TCRE	PR2	PR1	PR0
	W		▨	▨	▨				
0x000E TFLG1	R	C7F	C6F	C5F	C4F	C3F	C2F	C1F	C0F
	W								
0x000F TFLG2	R	TOF	0	0	0	0	0	0	0
	W		▨	▨	▨	▨	▨	▨	▨
0x0010 TC0 (High)	R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	W								
0x0011 TC0 (Low)	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	W								
0x0012 TC1 (High)	R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	W								
0x0013 TC1 (Low)	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	W								
0x0014 TC2 (High)	R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	W								
0x0015 TC2 (Low)	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	W								
0x0016 TC3 (High)	R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	W								
0x0017 TC3 (Low)	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	W								
0x0018 TC4 (High)	R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	W								
0x0019 TC4 (Low)	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	W								
0x001A TC5 (High)	R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	W								
0x001B TC5 (Low)	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	W								

▨ = Unimplemented or Reserved

Figure 19-2. ECT Register Summary (Sheet 2 of 5)

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x001C TC6 (High)	R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	W								
0x001D TC6 (Low)	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	W								
0x001E TC7 (High)	R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	W								
0x001F TC7 (Low)	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	W								
0x0020 PACTL	R	0	PAEN	PAMOD	PEDGE	CLK1	CLK0	PA0VI	PAI
	W	▨							
0x0021 PAFLG	R	0	0	0	0	0	0	PA0VF	PAIF
	W	▨	▨	▨	▨	▨	▨		
0x0022 PACN3	R	PACNT7(15)	PACNT6(14)	PACNT5(13)	PACNT4(12)	PACNT3(11)	PACNT2(10)	PACNT1(9)	PACNT0(8)
	W								
0x0023 PACN2	R	PACNT7	PACNT6	PACNT5	PACNT4	PACNT3	PACNT2	PACNT1	PACNT0
	W								
0x0024 PACN1	R	PACNT7(15)	PACNT6(14)	PACNT5(13)	PACNT4(12)	PACNT3(11)	PACNT2(10)	PACNT1(9)	PACNT0(8)
	W								
0x0025 PACN0	R	PACNT7	PACNT6	PACNT5	PACNT4	PACNT3	PACNT2	PACNT1	PACNT0
	W								
0x0026 MCCTL	R	MCZI	MODMC	RDMCL	0	0	MCEN	MCPR1	MCPR0
	W				ICLAT	FLMC			
0x0027 MCFLG	R	MCZF	0	0	0	POLF3	POLF2	POLF1	POLF0
	W		▨	▨	▨	▨	▨	▨	▨
0x0028 ICPAR	R	0	0	0	0	PA3EN	PA2EN	PA1EN	PA0EN
	W	▨	▨	▨	▨				
0x0029 DLYCT	R	DLY7	DLY6	DLY5	DLY4	DLY3	DLY2	DLY1	DLY0
	W								
0x002A ICOVW	R	NOVW7	NOVW6	NOVW5	NOVW4	NOVW3	NOVW2	NOVW1	NOVW0
	W								

▨ = Unimplemented or Reserved

Figure 19-2. ECT Register Summary (Sheet 3 of 5)

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x002B ICSYS	R	SH37	SH26	SH15	SH04	TFMOD	PACMX	BUFEN	LATQ
	W								
0x002C OCPD	R	OCPD7	OCPD6	OCPD5	OCPD4	OCPD3	OCPD2	OCPD1	OCPD0
	W								
0x002D TIMTST	R	Timer Test Register							
	W								
0x002E PTPSR	R	PTPS7	PTPS6	PTPS5	PTPS4	PTPS3	PTPS2	PTPS1	PTPS0
	W								
0x002F PTMCPSR	R	PTMPS7	PTMPS6	PTMPS5	PTMPS4	PTMPS3	PTMPS2	PTMPS1	PTMPS0
	W								
0x0030 PBCTL	R	0	PBEN	0	0	0	0	PBOVI	0
	W	▨		▨	▨	▨	▨		▨
0x0031 PBFLG	R	0	0	0	0	0	0	PBOVF	0
	W	▨	▨	▨	▨	▨	▨		▨
0x0032 PA3H	R	PA3H7	PA3H6	PA3H5	PA3H4	PA3H3	PA3H2	PA3H1	PA3H0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0033 PA2H	R	PA2H7	PA2H6	PA2H5	PA2H4	PA2H3	PA2H2	PA2H1	PA2H0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0034 PA1H	R	PA1H7	PA1H6	PA1H5	PA1H4	PA1H3	PA1H2	PA1H1	PA1H0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0035 PA0H	R	PA0H7	PA0H6	PA0H5	PA0H4	PA0H3	PA0H2	PA0H1	PA0H0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0036 MCCNT (High)	R	MCCNT15	MCCNT14	MCCNT13	MCCNT12	MCCNT11	MCCNT10	MCCNT9	MCCNT8
	W								
0x0037 MCCNT (Low)	R	MCCNT7	MCCNT6	MCCNT5	MCCNT4	MCCNT3	MCCNT2	MCCNT1	MCCNT0
	W								
0x0038 TC0H (High)	R	TC15	TC14	TC13	TC12	TC11	TC10	TC9	TC8
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x0039 TC0H (Low)	R	TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0
		▨	▨	▨	▨	▨	▨	▨	▨

▨ = Unimplemented or Reserved

Figure 19-2. ECT Register Summary (Sheet 4 of 5)

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x003A TC1H (High)	R	TC15	TC14	TC13	TC12	TC11	TC10	TC9	TC8
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x003B TC1H (Low)	R	TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x003C TC2H (High)	R	TC15	TC14	TC13	TC12	TC11	TC10	TC9	TC8
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x003D TC2H (Low)	R	TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x003E TC3H (High)	R	TC15	TC14	TC13	TC12	TC11	TC10	TC9	TC8
	W	▨	▨	▨	▨	▨	▨	▨	▨
0x003F TC3H (Low)	R	TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0
	W	▨	▨	▨	▨	▨	▨	▨	▨

▨ = Unimplemented or Reserved

Figure 19-2. ECT Register Summary (Sheet 5 of 5)

19.3.2.1 Timer Input Capture/Output Compare Select Register (TIOS)

Module Base + 0x0000

	7	6	5	4	3	2	1	0
R W	IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0
Reset	0	0	0	0	0	0	0	0

Figure 19-3. Timer Input Capture/Output Compare Register (TIOS)

Read or write: Anytime

All bits reset to zero.

Table 19-2. TIOS Field Descriptions

Field	Description
7:0 IOS[7:0]	**Input Capture or Output Compare Channel Configuration** 0 The corresponding channel acts as an input capture. 1 The corresponding channel acts as an output compare.

19.3.2.2 Timer Compare Force Register (CFORC)

Module Base + 0x0001

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	FOC7	FOC6	FOC5	FOC4	FOC3	FOC2	FOC1	FOC0
Reset	0	0	0	0	0	0	0	0

Figure 19-4. Timer Compare Force Register (CFORC)

Read or write: Anytime but reads will always return 0x0000 (1 state is transient).

All bits reset to zero.

Table 19-3. CFORC Field Descriptions

Field	Description
7:0 FOC[7:0]	**Force Output Compare Action for Channel 7:0** — A write to this register with the corresponding data bit(s) set causes the action which is programmed for output compare “x” to occur immediately. The action taken is the same as if a successful comparison had just taken place with the TCx register except the interrupt flag does not get set. **Note:** A channel 7 event, which can be a counter overflow when TTOV[7] is set or A successful channel 7 output compare overrides any channel 6:0 compares. If a forced output compare on any channel occurs at the same time as the successful output compare, then the forced output compare action will take precedence and the interrupt flag will not get set.

19.3.2.3 Output Compare 7 Mask Register (OC7M)

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R W	OC7M7	OC7M6	OC7M5	OC7M4	OC7M3	OC7M2	OC7M1	OC7M0
Reset	0	0	0	0	0	0	0	0

Figure 19-5. Output Compare 7 Mask Register (OC7M)

Read or write: Anytime

All bits reset to zero.

Table 19-4. OC7M Field Descriptions

Field	Description
7:0 OC7M[7:0]	**Output Compare Mask Action for Channel 7:0** **A channel 7 event, which can be a counter overflow when TTOV[7] is set or a successful output compare on channel 7, overrides any channel 6:0 compares. For each OC7M bit that is set,the output compare action reflects the corresponding OC7D bit.** 0 The corresponding OC7Dx bit in the output compare 7 data register will not be transferred to the timer port on a channel 7 event, even if the corresponding pin is setup for output compare. 1 The corresponding OC7Dx bit in the output compare 7 data register will be transferred to the timer port on a channel 7 event. **Note:** The corresponding channel must also be setup for output compare (IOSx = 1 andOCPDx = 0) for data to be transferred from the output compare 7 data register to the timer port.

19.3.2.4 Output Compare 7 Data Register (OC7D)

Module Base + 0x0003

	7	6	5	4	3	2	1	0
R W	OC7D7	OC7D6	OC7D5	OC7D4	OC7D3	OC7D2	OC7D1	OC7D0
Reset	0	0	0	0	0	0	0	0

Figure 19-6. Output Compare 7 Data Register (OC7D)

Read or write: Anytime

All bits reset to zero.

Table 19-5. OC7D Field Descriptions

Field	Description
7:0 OC7D[7:0]	**Output Compare 7 Data Bits** — A channel 7 event, which can be a counter overflow when TTOV[7] is set or A channel 7 output compare can cause bits in the output compare 7 data register to transfer to the timer port data register depending on the output compare 7 mask register.

19.3.2.5 Timer Count Register (TCNT)

Module Base + 0x0004

	15	14	13	12	11	10	9	8
R W	TCNT15	TCNT14	TCNT13	TCNT12	TCNT11	TCNT10	TCNT9	TCNT8
Reset	0	0	0	0	0	0	0	0

Figure 19-7. Timer Count Register High (TCNT)

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R W	TCNT7	TCNT6	TCNT5	TCNT4	TCNT3	TCNT2	TCNT1	TCNT0
Reset	0	0	0	0	0	0	0	0

Figure 19-8. Timer Count Register Low (TCNT)

Read: Anytime

Write: Writable in special modes.

All bits reset to zero.

Table 19-6. TCNT Field Descriptions

Field	Description
15:0 TCNT[15:0]	**Timer Counter Bits** — The 16-bit main timer is an up counter. A read to this register will return the current value of the counter. Access to the counter register will take place in one clock cycle. **Note:** A separate read/write for high byte and low byte in test mode will give a different result than accessing them as a word. The period of the first count after a write to the TCNT registers may be a different size because the write is not synchronized with the prescaler clock.

19.3.2.6 Timer System Control Register 1 (TSCR1)

Module Base + 0x0006

	7	6	5	4	3	2	1	0
R	TEN	TSWAI	TSFRZ	TFFCA	PRNT	0	0	0
W						▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-9. Timer System Control Register 1 (TSCR1)

Read or write: Anytime except PRNT bit is write once

All bits reset to zero.

Table 19-7. TSCR1 Field Descriptions

Field	Description
7 TEN	**Timer Enable** 0 Disables the main timer, including the counter. Can be used for reducing power consumption. 1 Allows the timer to function normally. **Note:** If for any reason the timer is not active, there is no ÷64 clock for the pulse accumulator since the ÷64 is generated by the timer prescaler.
6 TSWAI	**Timer Module Stops While in Wait** 0 Allows the timer module to continue running during wait. 1 Disables the timer counter, pulse accumulators and modulus down counter when the MCU is in wait mode. Timer interrupts cannot be used to get the MCU out of wait.

Table 19-7. TSCR1 Field Descriptions (continued)

Field	Description
5 TSFRZ	**Timer and Modulus Counter Stop While in Freeze Mode** 0 Allows the timer and modulus counter to continue running while in freeze mode. 1 Disables the timer and modulus counter whenever the MCU is in freeze mode. This is useful for emulation. The pulse accumulators do not stop in freeze mode.
4 TFFCA	**Timer Fast Flag Clear All** 0 Allows the timer flag clearing to function normally. 1 A read from an input capture or a write to the output compare channel registers causes the corresponding channel flag, CxF, to be cleared in the TFLG1 register. Any access to the TCNT register clears the TOF flag in the TFLG2 register. Any access to the PACN3 and PACN2 registers clears the PAOVF and PAIF flags in the PAFLG register. Any access to the PACN1 and PACN0 registers clears the PBOVF flag in the PBFLG register. Any access to the MCCNT register clears the MCZF flag in the MCFLG register. This has the advantage of eliminating software overhead in a separate clear sequence. Extra care is required to avoid accidental flag clearing due to unintended accesses. **Note:** The flags cannot be cleared via the normal flag clearing mechanism (writing a one to the flag) when TFFCA = 1.
3 PRNT	**Precision Timer** 0 Enables legacy timer. Only bits DLY0 and DLY1 of the DLYCT register are used for the delay selection of the delay counter. PR0, PR1, and PR2 bits of the TSCR2 register are used for timer counter prescaler selection. MCPR0 and MCPR1 bits of the MCCTL register are used for modulus down counter prescaler selection. 1 Enables precision timer. All bits in the DLYCT register are used for the delay selection, all bits of the PTPSR register are used for Precision Timer Prescaler Selection, and all bits of PTMCPSR register are used for the prescaler Precision Timer Modulus Counter Prescaler selection.

19.3.2.7 Timer Toggle On Overflow Register 1 (TTOV)

Module Base + 0x0007

	7	6	5	4	3	2	1	0
R W	TOV7	TOV6	TOV5	TOV4	TOV3	TOV2	TOV1	TOV0
Reset	0	0	0	0	0	0	0	0

Figure 19-10. Timer Toggle On Overflow Register 1 (TTOV)

Read or write: Anytime

All bits reset to zero.

Table 19-8. TTOV Field Descriptions

Field	Description
7:0 TOV[7:0]	**Toggle On Overflow Bits** — TOV97:0] toggles output compare pin on timer counter overflow. This feature only takes effect when in output compare mode. When set, it takes precedence over forced output compare but not channel 7 override events. 0 Toggle output compare pin on overflow feature disabled. 1 Toggle output compare pin on overflow feature enabled.

19.3.2.8 Timer Control Register 1/Timer Control Register 2 (TCTL1/TCTL2)

Module Base + 0x0008

	7	6	5	4	3	2	1	0
R W	OM7	OL7	OM6	OL6	OM5	OL5	OM4	OL4
Reset	0	0	0	0	0	0	0	0

Figure 19-11. Timer Control Register 1 (TCTL1)

Module Base + 0x0009

	7	6	5	4	3	2	1	0
R W	OM3	OL3	OM2	OL2	OM1	OL1	OM0	OL0
Reset	0	0	0	0	0	0	0	0

Figure 19-12. Timer Control Register 2 (TCTL2)

Read or write: Anytime

All bits reset to zero.

Table 19-9. TCTL1/TCTL2 Field Descriptions

Field	Description
OM[7:0] 7, 5, 3, 1	OMx — Output Mode OLx — Output Level These eight pairs of control bits are encoded to specify the output action to be taken as a result of a successful OCx compare. When either OMx or OLx is one, the pin associated with OCx becomes an output tied to OCx. See Table 19-10.
OL[7:0] 6, 4, 2, 0	

Table 19-10. Compare Result Output Action

OMx	OLx	Action
0	0	No output compare action on the timer output signal
0	1	Toggle OCx output line
1	0	Clear OCx output line to zero
1	1	Set OCx output line to one

NOTE

To enable output action by OMx and OLx bits on timer port, the corresponding bit in OC7M should be cleared. The settings for these bits can be seen in Table 19-11

Table 19-11. The OC7 and OCx event priority

OC7M7=0				OC7M7=1			
OC7Mx=1		OC7Mx=0		OC7Mx=1		OC7Mx=0	
TC7=TCx	TC7>TCx	TC7=TCx	TC7>TCx	TC7=TCx	TC7>TCx	TC7=TCx	TC7>TCx
IOCx=OC7Dx IOC7=OM7/OL7	IOCx=OC7Dx +OMx/OLx IOC7=OM7/OL7	IOCx=OMx/OLx IOC7=OM7/OL7		IOCx=OC7Dx IOC7=OC7D7	IOCx=OC7Dx +OMx/OLx IOC7=OC7D7	IOCx=OMx/OLx IOC7=OC7D7	

Note: in Table 19-11,the IOS7 and IOSx should be set to 1

IOSx is the register TIOS bit x,

OC7Mx is the register OC7M bit x,

TCx is timer Input Capture/Output Compare register,

IOCx is channel x,

OMx/OLx is the register TCTL1/TCTL2,

OC7Dx is the register OC7D bit x.

IOCx = OC7Dx+ OMx/OLx, means that both OC7 event and OCx event will change channel x value.

19.3.2.9 Timer Control Register 3/Timer Control Register 4 (TCTL3/TCTL4)

Module Base + 0x000A

	7	6	5	4	3	2	1	0
R W	EDG7B	EDG7A	EDG6B	EDG6A	EDG5B	EDG5A	EDG4B	EDG4A
Reset	0	0	0	0	0	0	0	0

Figure 19-13. Timer Control Register 3 (TCTL3)

Module Base + 0x000B

	7	6	5	4	3	2	1	0
R W	EDG3B	EDG3A	EDG2B	EDG2A	EDG1B	EDG1A	EDG0B	EDG0A
Reset	0	0	0	0	0	0	0	0

Figure 19-14. Timer Control Register 4 (TCTL4)

Read or write: Anytime

All bits reset to zero.

Table 19-12. TCTL3/TCTL4 Field Descriptions

Field	Description
EDG[7:0]B 7, 5, 3, 1	**Input Capture Edge Control** — These eight pairs of control bits configure the input capture edge detector circuits for each input capture channel. The four pairs of control bits in TCTL4 also configure the input capture edge control for the four 8-bit pulse accumulators PAC0–PAC3.EDG0B and EDG0A in TCTL4 also determine the active edge for the 16-bit pulse accumulator PACB. See Table 19-13.
EDG[7:0]A 6, 4, 2, 0	

Table 19-13. Edge Detector Circuit Configuration

EDGxB	EDGxA	Configuration
0	0	Capture disabled
0	1	Capture on rising edges only
1	0	Capture on falling edges only
1	1	Capture on any edge (rising or falling)

19.3.2.10 Timer Interrupt Enable Register (TIE)

Module Base + 0x000C

	7	6	5	4	3	2	1	0
R W	C7I	C6I	C5I	C4I	C3I	C2I	C1I	C0I
Reset	0	0	0	0	0	0	0	0

Figure 19-15. Timer Interrupt Enable Register (TIE)

Read or write: Anytime

All bits reset to zero.

The bits C7I–C0I correspond bit-for-bit with the flags in the TFLG1 status register.

Table 19-14. TIE Field Descriptions

Field	Description
7:0 C[7:0]I	**Input Capture/Output Compare "x" Interrupt Enable** 0 The corresponding flag is disabled from causing a hardware interrupt. 1 The corresponding flag is enabled to cause an interrupt.

19.3.2.11 Timer System Control Register 2 (TSCR2)

Module Base + 0x000D

	7	6	5	4	3	2	1	0
R	TOI	0	0	0	TCRE	PR2	PR1	PR0
W		▨	▨	▨				
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-16. Timer System Control Register 2 (TSCR2)

Read or write: Anytime

All bits reset to zero.

Table 19-15. TSCR2 Field Descriptions

Field	Description
7 TOI	**Timer Overflow Interrupt Enable** 0 Timer overflow interrupt disabled. 1 Hardware interrupt requested when TOF flag set.
3 TCRE	**Timer Counter Reset Enable** — This bit allows the timer counter to be reset by a successful channel 7 output compare. This mode of operation is similar to an up-counting modulus counter. 0 Counter reset disabled and counter free runs. 1 Counter reset by a successful output compare on channel 7. **Note:** If register TC7 = 0x0000 and TCRE = 1, then the TCNT register will stay at 0x0000 continuously. If register TC7 = 0xFFFF and TCRE = 1, the TOF flag will never be set when TCNT is reset from 0xFFFF to 0x0000. **Note:** TCRE=1 and TC7!=0, the TCNT cycle period will be TC7 x "prescaler counter width" + "1 Bus Clock". When TCRE is set and TC7 is not equal to 0, TCNT will cycle from 0 to TC7. When TCNT reaches TC7 value, it will last only one bus cycle then reset to 0. for a more detail explanation please refer to Figure 19-17. **Note:** in Figure 19-17,if PR[2:0] is equal to 0, one prescaler counter equal to one bus clock
2:0 PR[2:0]	**Timer Prescaler Select** — These three bits specify the division rate of the main Timer prescaler when the PRNT bit of register TSCR1 is set to 0. The newly selected prescale factor will not take effect until the next synchronized edge where all prescale counter stages equal zero. See Table 19-16.

Figure 19-17. The TCNT cycle diagram under TCRE=1 condition

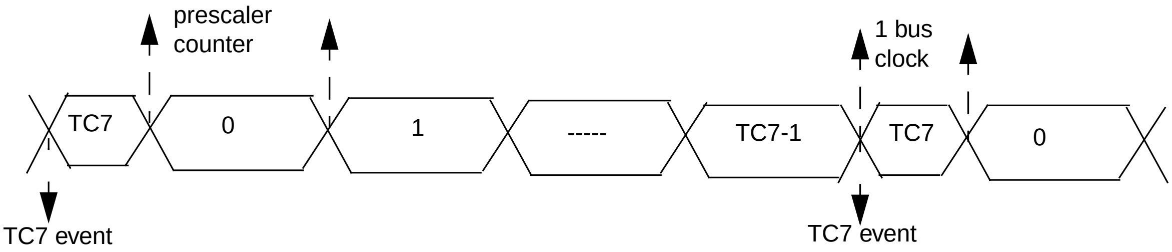


Table 19-16. Prescaler Selection

PR2	PR1	PR0	Prescale Factor
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

19.3.2.12 Main Timer Interrupt Flag 1 (TFLG1)

Module Base + 0x000E

	7	6	5	4	3	2	1	0
R W	C7F	C6F	C5F	C4F	C3F	C2F	C1F	C0F
Reset	0	0	0	0	0	0	0	0

Figure 19-18. Main Timer Interrupt Flag 1 (TFLG1)

Read: Anytime

Write used in the flag clearing mechanism. Writing a one to the flag clears the flag. Writing a zero will not affect the current status of the bit.

NOTE

> When TFFCA = 1, the flags cannot be cleared via the normal flag clearing mechanism (writing a one to the flag). Reference Section 19.3.2.6, “Timer System Control Register 1 (TSCR1)”.

All bits reset to zero.

TFLG1 indicates when interrupt conditions have occurred. The flags can be cleared via the normal flag clearing mechanism (writing a one to the flag) or via the fast flag clearing mechanism (reference TFFCA bit in Section 19.3.2.6, “Timer System Control Register 1 (TSCR1)”).

Use of the TFMOD bit in the ICSYS register in conjunction with the use of the ICOVW register allows a timer interrupt to be generated after capturing two values in the capture and holding registers, instead of generating an interrupt for every capture.

Table 19-17. TFLG1 Field Descriptions

Field	Description
7:0 C[7:0]F	**Input Capture/Output Compare Channel “x” Flag** — A CxF flag is set when a corresponding input capture or output compare is detected. C0F can also be set by 16-bit Pulse Accumulator B (PACB). C3F–C0F can also be set by 8-bit pulse accumulators PAC3–PAC0. If the delay counter is enabled, the CxF flag will not be set until after the delay.

19.3.2.13 Main Timer Interrupt Flag 2 (TFLG2)

Module Base + 0x000F

	7	6	5	4	3	2	1	0
R	TOF	0	0	0	0	0	0	0
W		▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-19. Main Timer Interrupt Flag 2 (TFLG2)

Read: Anytime

Write used in the flag clearing mechanism. Writing a one to the flag clears the flag. Writing a zero will not affect the current status of the bit.

NOTE

> When TFFCA = 1, the flag cannot be cleared via the normal flag clearing mechanism (writing a one to the flag). Reference Section 19.3.2.6, “Timer System Control Register 1 (TSCR1)”.

All bits reset to zero.

TFLG2 indicates when interrupt conditions have occurred. The flag can be cleared via the normal flag clearing mechanism (writing a one to the flag) or via the fast flag clearing mechanism (Reference TFFCA bit in Section 19.3.2.6, “Timer System Control Register 1 (TSCR1)”).

Table 19-18. TFLG2 Field Descriptions

Field	Description
7 TOF	**Timer Overflow Flag** — Set when 16-bit free-running timer overflows from 0xFFFF to 0x0000.

19.3.2.14 Timer Input Capture/Output Compare Registers 0–7

Module Base + 0x0010

	15	14	13	12	11	10	9	8
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset	0	0	0	0	0	0	0	0

Figure 19-20. Timer Input Capture/Output Compare Register 0 High (TC0)

Module Base + 0x0011

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 19-21. Timer Input Capture/Output Compare Register 0 Low (TC0)

Module Base + 0x0012

	15	14	13	12	11	10	9	8
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset	0	0	0	0	0	0	0	0

Figure 19-22. Timer Input Capture/Output Compare Register 1 High (TC1)

Module Base + 0x0013

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 19-23. Timer Input Capture/Output Compare Register 1 Low (TC1)

Module Base + 0x0014

	15	14	13	12	11	10	9	8
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset	0	0	0	0	0	0	0	0

Figure 19-24. Timer Input Capture/Output Compare Register 2 High (TC2)

Module Base + 0x0015

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 19-25. Timer Input Capture/Output Compare Register 2 Low (TC2)

Module Base + 0x0016

	15	14	13	12	11	10	9	8
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset	0	0	0	0	0	0	0	0

Figure 19-26. Timer Input Capture/Output Compare Register 3 High (TC3)

Module Base + 0x0017

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 19-27. Timer Input Capture/Output Compare Register 3 Low (TC3)

Module Base + 0x0018

	15	14	13	12	11	10	9	8
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset	0	0	0	0	0	0	0	0

Figure 19-28. Timer Input Capture/Output Compare Register 4 High (TC4)

Module Base + 0x0019

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 19-29. Timer Input Capture/Output Compare Register 4 Low (TC4)

Module Base + 0x001A

	15	14	13	12	11	10	9	8
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset	0	0	0	0	0	0	0	0

Figure 19-30. Timer Input Capture/Output Compare Register 5 High (TC5)

Module Base + 0x001B

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 19-31. Timer Input Capture/Output Compare Register 5 Low (TC5)

Module Base + 0x001C

	15	14	13	12	11	10	9	8
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset	0	0	0	0	0	0	0	0

Figure 19-32. Timer Input Capture/Output Compare Register 6 High (TC6)

Module Base + 0x001D

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 19-33. Timer Input Capture/Output Compare Register 6 Low (TC6)

Module Base + 0x001E

	15	14	13	12	11	10	9	8
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset	0	0	0	0	0	0	0	0

Figure 19-34. Timer Input Capture/Output Compare Register 7 High (TC7)

Module Base + 0x001F

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 19-35. Timer Input Capture/Output Compare Register 7 Low (TC7)

Read: Anytime

Write anytime for output compare function. Writes to these registers have no meaning or effect during input capture.

All bits reset to zero.

Depending on the TIOS bit for the corresponding channel, these registers are used to latch the value of the free-running counter when a defined transition is sensed by the corresponding input capture edge detector or to trigger an output action for output compare.

19.3.2.15 16-Bit Pulse Accumulator A Control Register (PACTL)

Module Base + 0x0020

	7	6	5	4	3	2	1	0
R	0	PAEN	PAMOD	PEDGE	CLK1	CLK0	PAOVI	PAI
W	▨							
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-36. 16-Bit Pulse Accumulator Control Register (PACTL)

Read: Anytime

Write: Anytime

All bits reset to zero.

Table 19-19. PACTL Field Descriptions

Field	Description
6 PAEN	**Pulse Accumulator A System Enable** — PAEN is independent from TEN. With timer disabled, the pulse accumulator can still function unless pulse accumulator is disabled. 0 16-Bit Pulse Accumulator A system disabled. 8-bit PAC3 and PAC2 can be enabled when their related enable bits in ICPAR are set. Pulse Accumulator Input Edge Flag (PAIF) function is disabled. 1 16-Bit Pulse Accumulator A system enabled. The two 8-bit pulse accumulators PAC3 and PAC2 are cascaded to form the PACA 16-bit pulse accumulator. When PACA in enabled, the PACN3 and PACN2 registers contents are respectively the high and low byte of the PACA. PA3EN and PA2EN control bits in ICPAR have no effect. Pulse Accumulator Input Edge Flag (PAIF) function is enabled. The PACA shares the input pin with IC7.
5 PAMOD	**Pulse Accumulator Mode** — This bit is active only when the Pulse Accumulator A is enabled (PAEN = 1). 0 Event counter mode 1 Gated time accumulation mode

Table 19-19. PACTL Field Descriptions (continued)

Field	Description
4 PEDGE	**Pulse Accumulator Edge Control** — This bit is active only when the Pulse Accumulator A is enabled (PAEN = 1). Refer to Table 19-20. For PAMOD bit = 0 (event counter mode). 0 Falling edges on IC7 pin cause the count to be incremented 1 Rising edges on IC7 pin cause the count to be incremented For PAMOD bit = 1 (gated time accumulation mode). 0 IC7 input pin high enables bus clock divided by 64 to Pulse Accumulator and the trailing falling edge on IC7 sets the PAIF flag. 1 IC7 input pin low enables bus clock divided by 64 to Pulse Accumulator and the trailing rising edge on IC7 sets the PAIF flag. If the timer is not active (TEN = 0 in TSCR1), there is no divide-by-64 since the ÷64 clock is generated by the timer prescaler.
3:2 CLK[1:0]	**Clock Select Bits** — For the description of PACLK please refer to Figure 19-72. If the pulse accumulator is disabled (PAEN = 0), the prescaler clock from the timer is always used as an input clock to the timer counter. The change from one selected clock to the other happens immediately after these bits are written. Refer to Table 19-21.
2 PAOVI	**Pulse Accumulator A Overflow Interrupt Enable** 0 Interrupt inhibited 1 Interrupt requested if PAOVF is set
0 PAI	**Pulse Accumulator Input Interrupt Enable** 0 Interrupt inhibited 1 Interrupt requested if PAIF is set

Table 19-20. Pin Action

PAMOD	PEDGE	Pin Action
0	0	Falling edge
0	1	Rising edge
1	0	Divide by 64 clock enabled with pin high level
1	1	Divide by 64 clock enabled with pin low level

Table 19-21. Clock Selection

CLK1	CLK0	Clock Source
0	0	Use timer prescaler clock as timer counter clock
0	1	Use PACLK as input to timer counter clock
1	0	Use PACLK/256 as timer counter clock frequency
1	1	Use PACLK/65536 as timer counter clock frequency

19.3.2.16 Pulse Accumulator A Flag Register (PAFLG)

Module Base + 0x0021

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	PAOVF	PAIF
W	▨	▨	▨	▨	▨	▨		
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-37. Pulse Accumulator A Flag Register (PAFLG)

Read: Anytime

Write used in the flag clearing mechanism. Writing a one to the flag clears the flag. Writing a zero will not affect the current status of the bit.

NOTE

> When TFFCA = 1, the flags cannot be cleared via the normal flag clearing mechanism (writing a one to the flag). Reference Section 19.3.2.6, “Timer System Control Register 1 (TSCR1)”.

All bits reset to zero.

PAFLG indicates when interrupt conditions have occurred. The flags can be cleared via the normal flag clearing mechanism (writing a one to the flag) or via the fast flag clearing mechanism (Reference TFFCA bit in Section 19.3.2.6, “Timer System Control Register 1 (TSCR1)”).

Table 19-22. PAFLG Field Descriptions

Field	Description
1 PAOVF	**Pulse Accumulator A Overflow Flag** — Set when the 16-bit pulse accumulator A overflows from 0xFFFF to 0x0000, or when 8-bit pulse accumulator 3 (PAC3) overflows from 0x00FF to 0x0000. When PACMX = 1, PAOVF bit can also be set if 8-bit pulse accumulator 3 (PAC3) reaches 0x00FF followed by an active edge on IC3.
0 PAIF	**Pulse Accumulator Input edge Flag** — Set when the selected edge is detected at the IC7 input pin. In event mode the event edge triggers PAIF and in gated time accumulation mode the trailing edge of the gate signal at the IC7 input pin triggers PAIF.

19.3.2.17 Pulse Accumulators Count Registers (PACN3 and PACN2)

Module Base + 0x0022

	7	6	5	4	3	2	1	0
R W	PACNT7(15)	PACNT6(14)	PACNT5(13)	PACNT4(12)	PACNT3(11)	PACNT2(10)	PACNT1(9)	PACNT0(8)
Reset	0	0	0	0	0	0	0	0

Figure 19-38. Pulse Accumulators Count Register 3 (PACN3)

Module Base + 0x0023

	7	6	5	4	3	2	1	0
R W	PACNT7	PACNT6	PACNT5	PACNT4	PACNT3	PACNT2	PACNT1	PACNT0
Reset	0	0	0	0	0	0	0	0

Figure 19-39. Pulse Accumulators Count Register 2 (PACN2)

Read: Anytime

Write: Anytime

All bits reset to zero.

The two 8-bit pulse accumulators PAC3 and PAC2 are cascaded to form the PACA 16-bit pulse accumulator. When PACA in enabled (PAEN = 1 in PACTL), the PACN3 and PACN2 registers contents are respectively the high and low byte of the PACA.

When PACN3 overflows from 0x00FF to 0x0000, the interrupt flag PAOVF in PAFLG is set.

Full count register access will take place in one clock cycle.

NOTE

A separate read/write for high byte and low byte will give a different result than accessing them as a word.

When clocking pulse and write to the registers occurs simultaneously, write takes priority and the register is not incremented.

19.3.2.18 Pulse Accumulators Count Registers (PACN1 and PACN0)

Module Base + 0x0024

	7	6	5	4	3	2	1	0
R W	PACNT7(15)	PACNT6(14)	PACNT5(13)	PACNT4(12)	PACNT3(11)	PACNT2(10)	PACNT1(9)	PACNT0(8)
Reset	0	0	0	0	0	0	0	0

Figure 19-40. Pulse Accumulators Count Register 1 (PACN1)

Module Base + 0x0025

	7	6	5	4	3	2	1	0
R W	PACNT7	PACNT6	PACNT5	PACNT4	PACNT3	PACNT2	PACNT1	PACNT0
Reset	0	0	0	0	0	0	0	0

Figure 19-41. Pulse Accumulators Count Register 0 (PACN0)

Read: Anytime

Write: Anytime

All bits reset to zero.

The two 8-bit pulse accumulators PAC1 and PAC0 are cascaded to form the PACB 16-bit pulse accumulator. When PACB in enabled, (PBEN = 1 in PBCTL) the PACN1 and PACN0 registers contents are respectively the high and low byte of the PACB.

When PACN1 overflows from 0x00FF to 0x0000, the interrupt flag PBOVF in PBFLG is set.

Full count register access will take place in one clock cycle.

NOTE

> A separate read/write for high byte and low byte will give a different result than accessing them as a word.
>
> When clocking pulse and write to the registers occurs simultaneously, write takes priority and the register is not incremented.

19.3.2.19 16-Bit Modulus Down-Counter Control Register (MCCTL)

Module Base + 0x0026

	7	6	5	4	3	2	1	0
R	MCZI	MODMC	RDMCL	0	0	MCEN	MCPR1	MCPR0
W				ICLAT	FLMC			
Reset	0	0	0	0	0	0	0	0

Figure 19-42. 16-Bit Modulus Down-Counter Control Register (MCCTL)

Read: Anytime

Write: Anytime

All bits reset to zero.

Table 19-23. MCCTL Field Descriptions

Field	Description
7 MCZI	**Modulus Counter Underflow Interrupt Enable** 0 Modulus counter interrupt is disabled. 1 Modulus counter interrupt is enabled.
6 MODMC	**Modulus Mode Enable** 0 The modulus counter counts down from the value written to it and will stop at 0x0000. 1 Modulus mode is enabled. When the modulus counter reaches 0x0000, the counter is loaded with the latest value written to the modulus count register. **Note:** For proper operation, the MCEN bit should be cleared before modifying the MODMC bit in order to reset the modulus counter to 0xFFFF.
5 RDMCL	**Read Modulus Down-Counter Load** 0 Reads of the modulus count register (MCCNT) will return the present value of the count register. 1 Reads of the modulus count register (MCCNT) will return the contents of the load register.

Table 19-23. MCCTL Field Descriptions (continued)

Field	Description
4 ICLAT	**Input Capture Force Latch Action** — When input capture latch mode is enabled (LATQ and BUFEN bit in ICSYS are set), a write one to this bit immediately forces the contents of the input capture registers TC0 to TC3 and their corresponding 8-bit pulse accumulators to be latched into the associated holding registers. The pulse accumulators will be automatically cleared when the latch action occurs. Writing zero to this bit has no effect. Read of this bit will always return zero.
3 FLMC	**Force Load Register into the Modulus Counter Count Register** — This bit is active only when the modulus down-counter is enabled (MCEN = 1). A write one into this bit loads the load register into the modulus counter count register (MCCNT). This also resets the modulus counter prescaler. Write zero to this bit has no effect. Read of this bit will return always zero.
2 MCEN	**Modulus Down-Counter Enable** 0 Modulus counter disabled. The modulus counter (MCCNT) is preset to 0xFFFF. This will prevent an early interrupt flag when the modulus down-counter is enabled. 1 Modulus counter is enabled.
1:0 MCPR[1:0]	**Modulus Counter Prescaler Select** — These two bits specify the division rate of the modulus counter prescaler when PRNT of TSCR1 is set to 0. The newly selected prescaler division rate will not be effective until a load of the load register into the modulus counter count register occurs.

Table 19-24. Modulus Counter Prescaler Select

MCPR1	MCPR0	Prescaler Division
0	0	1
0	1	4
1	0	8
1	1	16

19.3.2.20 16-Bit Modulus Down-Counter FLAG Register (MCFLG)

Module Base + 0x0027

	7	6	5	4	3	2	1	0
R	MCZF	0	0	0	POLF3	POLF2	POLF1	POLF0
W		▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-43. 16-Bit Modulus Down-Counter FLAG Register (MCFLG)

Read: Anytime

Write only used in the flag clearing mechanism for bit 7. Writing a one to bit 7 clears the flag. Writing a zero will not affect the current status of the bit.

NOTE

> When TFFCA = 1, the flag cannot be cleared via the normal flag clearing mechanism (writing a one to the flag). Reference Section 19.3.2.6, “Timer System Control Register 1 (TSCR1)”.

All bits reset to zero.

Table 19-25. MCFLG Field Descriptions

Field	Description
7 MCZF	**Modulus Counter Underflow Flag** — The flag is set when the modulus down-counter reaches 0x0000. The flag indicates when interrupt conditions have occurred. The flag can be cleared via the normal flag clearing mechanism (writing a one to the flag) or via the fast flag clearing mechanism (Reference TFFCA bit in Section 19.3.2.6, “Timer System Control Register 1 (TSCR1)”).
3:0 POLF[3:0]	**First Input Capture Polarity Status** — These are read only bits. Writes to these bits have no effect. Each status bit gives the polarity of the first edge which has caused an input capture to occur after capture latch has been read. Each POLFx corresponds to a timer PORTx input. 0 The first input capture has been caused by a falling edge. 1 The first input capture has been caused by a rising edge.

19.3.2.21 ICPAR — Input Control Pulse Accumulators Register (ICPAR)

Module Base + 0x0028

	7	6	5	4	3	2	1	0
R	0	0	0	0	PA3EN	PA2EN	PA1EN	PA0EN
W	▨	▨	▨	▨				
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-44. Input Control Pulse Accumulators Register (ICPAR)

Read: Anytime

Write: Anytime.

All bits reset to zero.

The 8-bit pulse accumulators PAC3 and PAC2 can be enabled only if PAEN in PACTL is cleared. If PAEN is set, PA3EN and PA2EN have no effect.

The 8-bit pulse accumulators PAC1 and PAC0 can be enabled only if PBEN in PBCTL is cleared. If PBEN is set, PA1EN and PA0EN have no effect.

Table 19-26. ICPAR Field Descriptions

Field	Description
3:0 PA[3:0]EN	**8-Bit Pulse Accumulator ‘x’ Enable** 0 8-Bit Pulse Accumulator is disabled. 1 8-Bit Pulse Accumulator is enabled.

19.3.2.22 Delay Counter Control Register (DLYCT)

Module Base + 0x0029

	7	6	5	4	3	2	1	0
R W	DLY7	DLY6	DLY5	DLY4	DLY3	DLY2	DLY1	DLY0
Reset	0	0	0	0	0	0	0	0

Figure 19-45. Delay Counter Control Register (DLYCT)

Read: Anytime

Write: Anytime

All bits reset to zero.

Table 19-27. DLYCT Field Descriptions

Field	Description
7:0 DLY[7:0]	**Delay Counter Select** — When the PRNT bit of TSCR1 register is set to 0, only bits DLY0, DLY1 are used to calculate the delay.Table 19-28 shows the delay settings in this case. When the PRNT bit of TSCR1 register is set to 1, all bits are used to set a more precise delay. Table 19-29 shows the delay settings in this case. After detection of a valid edge on an input capture pin, the delay counter counts the pre-selected number of [(dly_cnt + 1)*4]bus clock cycles, then it will generate a pulse on its output if the level of input signal, after the preset delay, is the opposite of the level before the transition.This will avoid reaction to narrow input pulses. Delay between two active edges of the input signal period should be longer than the selected counter delay. **Note:** It is recommended to not write to this register while the timer is enabled, that is when TEN is set in register TSCR1.

Table 19-28. Delay Counter Select when PRNT = 0

DLY1	DLY0	Delay
0	0	Disabled
0	1	256 bus clock cycles
1	0	512 bus clock cycles
1	1	1024 bus clock cycles

Table 19-29. Delay Counter Select Examples when PRNT = 1

DLY7	DLY6	DLY5	DLY4	DLY3	DLY2	DLY1	DLY0	Delay
0	0	0	0	0	0	0	0	Disabled (bypassed)
0	0	0	0	0	0	0	1	8 bus clock cycles
0	0	0	0	0	0	1	0	12 bus clock cycles
0	0	0	0	0	0	1	1	16 bus clock cycles
0	0	0	0	0	1	0	0	20 bus clock cycles
0	0	0	0	0	1	0	1	24 bus clock cycles
0	0	0	0	0	1	1	0	28 bus clock cycles

Table 19-29. Delay Counter Select Examples when PRNT = 1

DLY7	DLY6	DLY5	DLY4	DLY3	DLY2	DLY1	DLY0	Delay
0	0	0	0	0	1	1	1	32 bus clock cycles
0	0	0	0	1	1	1	1	64 bus clock cycles
0	0	0	1	1	1	1	1	128 bus clock cycles
0	0	1	1	1	1	1	1	256 bus clock cycles
0	1	1	1	1	1	1	1	512 bus clock cycles
1	1	1	1	1	1	1	1	1024 bus clock cycles

19.3.2.23 Input Control Overwrite Register (ICOVW)

Module Base + 0x002A

	7	6	5	4	3	2	1	0
R W	NOVW7	NOVW6	NOVW5	NOVW4	NOVW3	NOVW2	NOVW1	NOVW0
Reset	0	0	0	0	0	0	0	0

Figure 19-46. Input Control Overwrite Register (ICOVW)

Read: Anytime

Write: Anytime

All bits reset to zero.

Table 19-30. ICOVW Field Descriptions

Field	Description
7:0 NOVW[7:0]	**No Input Capture Overwrite** 0 The contents of the related capture register or holding register can be overwritten when a new input capture or latch occurs. 1 The related capture register or holding register cannot be written by an event unless they are empty (see Section 19.4.1.1, “IC Channels”). This will prevent the captured value being overwritten until it is read or latched in the holding register.

19.3.2.24 Input Control System Control Register (ICSYS)

Module Base + 0x002B

	7	6	5	4	3	2	1	0
R W	SH37	SH26	SH15	SH04	TFMOD	PACMX	BUFEN	LATQ
Reset	0	0	0	0	0	0	0	0

Figure 19-47. Input Control System Register (ICSYS)

Read: Anytime

Write: Once in normal modes

All bits reset to zero.

Table 19-31. ICSYS Field Descriptions

Field	Description
7:4 SHxy	**Share Input action of Input Capture Channels x and y** 0 Normal operation 1 The channel input ‘x’ causes the same action on the channel ‘y’. The port pin ‘x’ and the corresponding edge detector is used to be active on the channel ‘y’.
3 TFMOD	**Timer Flag Setting Mode** — Use of the TFMOD bit in conjunction with the use of the ICOVW register allows a timer interrupt to be generated after capturing two values in the capture and holding registers instead of generating an interrupt for every capture. By setting TFMOD in queue mode, when NOVWx bit is set and the corresponding capture and holding registers are emptied, an input capture event will first update the related input capture register with the main timer contents. At the next event, the TCx data is transferred to the TCxH register, the TCx is updated and the CxF interrupt flag is set. In all other input capture cases the interrupt flag is set by a valid external event on ICx. 0 The timer flags C3F–C0F in TFLG1 are set when a valid input capture transition on the corresponding port pin occurs. 1 If in queue mode (BUFEN = 1 and LATQ = 0), the timer flags C3F–C0F in TFLG1 are set only when a latch on the corresponding holding register occurs. If the queue mode is not engaged, the timer flags C3F–C0F are set the same way as for TFMOD = 0.
2 PACMX	**8-Bit Pulse Accumulators Maximum Count** 0 Normal operation. When the 8-bit pulse accumulator has reached the value 0x00FF, with the next active edge, it will be incremented to 0x0000. 1 When the 8-bit pulse accumulator has reached the value 0x00FF, it will not be incremented further. The value 0x00FF indicates a count of 255 or more.
1 BUFFEN	**IC Buffer Enable** 0 Input capture and pulse accumulator holding registers are disabled. 1 Input capture and pulse accumulator holding registers are enabled. The latching mode is defined by LATQ control bit.
0 LATQ	**Input Control Latch or Queue Mode Enable** — The BUFEN control bit should be set in order to enable the IC and pulse accumulators holding registers. Otherwise LATQ latching modes are disabled. Write one into ICLAT bit in MCCTL, when LATQ and BUFEN are set will produce latching of input capture and pulse accumulators registers into their holding registers. 0 Queue mode of Input Capture is enabled. The main timer value is memorized in the IC register by a valid input pin transition. With a new occurrence of a capture, the value of the IC register will be transferred to its holding register and the IC register memorizes the new timer value. 1 Latch mode is enabled. Latching function occurs when modulus down-counter reaches zero or a zero is written into the count register MCCNT (see Section 19.4.1.1.2, “Buffered IC Channels”). With a latching event the contents of IC registers and 8-bit pulse accumulators are transferred to their holding registers. 8-bit pulse accumulators are cleared.

19.3.2.25 Output Compare Pin Disconnect Register (OCPD)

Module Base + 0x002C

	7	6	5	4	3	2	1	0
R W	OCPD7	OCPD6	OCPD5	OCPD4	OCPD3	OCPD2	OCPD1	OCPD0
Reset	0	0	0	0	0	0	0	0

Figure 19-48. Output Compare Pin Disconnect Register (OCPD)

Read: Anytime

Write: Anytime

All bits reset to zero.

Table 19-32. OCPD Field Descriptions

Field	Description
7:0 OCPD[7:0]	**Output Compare Pin Disconnect Bits** 0 Enables the timer channel IO port. Output Compare actions will occur on the channel pin. These bits do not affect the input capture or pulse accumulator functions. 1 Disables the timer channel IO port. Output Compare actions will not affect on the channel pin; the output compare flag will still be set on an Output Compare event.

19.3.2.26 Precision Timer Prescaler Select Register (PTPSR)

Module Base + 0x002E

	7	6	5	4	3	2	1	0
R W	PTPS7	PTPS6	PTPS5	PTPS4	PTPS3	PTPS2	PTPS1	PTPS0
Reset	0	0	0	0	0	0	0	0

Figure 19-49. Precision Timer Prescaler Select Register (PTPSR)

Read: Anytime

Write: Anytime

All bits reset to zero.

Table 19-33. PTPSR Field Descriptions

Field	Description
7:0 PTPS[7:0]	**Precision Timer Prescaler Select Bits** — These eight bits specify the division rate of the main Timer prescaler. These are effective only when the PRNT bit of TSCR1 is set to 1. Table 19-34 shows some selection examples in this case. The newly selected prescale factor will not take effect until the next synchronized edge where all prescale counter stages equal zero.

Table 19-34. Precision Timer Prescaler Selection Examples when PRNT = 1

PTPS7	PTPS6	PTPS5	PTPS4	PTPS3	PTPS2	PTPS1	PTPS0	Prescale Factor
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	2
0	0	0	0	0	0	1	0	3
0	0	0	0	0	0	1	1	4
0	0	0	0	0	1	0	0	5
0	0	0	0	0	1	0	1	6
0	0	0	0	0	1	1	0	7
0	0	0	0	0	1	1	1	8
0	0	0	0	1	1	1	1	16
0	0	0	1	1	1	1	1	32
0	0	1	1	1	1	1	1	64
0	1	1	1	1	1	1	1	128
1	1	1	1	1	1	1	1	256

19.3.2.27 Precision Timer Modulus Counter Prescaler Select Register (PTMCPSR)

Module Base + 0x002F

	7	6	5	4	3	2	1	0
R W	PTMPS7	PTMPS6	PTMPS5	PTMPS4	PTMPS3	PTMPS2	PTMPS1	PTMPS0
Reset	0	0	0	0	0	0	0	0

Figure 19-50. Precision Timer Modulus Counter Prescaler Select Register (PTMCPSR)

Read: Anytime

Write: Anytime

All bits reset to zero.

Table 19-35. PTMCPSR Field Descriptions

Field	Description
7:0 PTMPS[7:0]	**Precision Timer Modulus Counter Prescaler Select Bits** — These eight bits specify the division rate of the modulus counter prescaler. These are effective only when the PRNT bit of TSCR1 is set to 1. Table 19-36 shows some possible division rates. The newly selected prescaler division rate will not be effective until a load of the load register into the modulus counter count register occurs.

Table 19-36. Precision Timer Modulus Counter Prescaler Select Examples when PRNT = 1

PTMPS7	PTMPS6	PTMPS5	PTMPS4	PTMPS3	PTMPS2	PTMPS1	PTMPS0	Prescaler Division Rate
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	2
0	0	0	0	0	0	1	0	3
0	0	0	0	0	0	1	1	4
0	0	0	0	0	1	0	0	5
0	0	0	0	0	1	0	1	6
0	0	0	0	0	1	1	0	7
0	0	0	0	0	1	1	1	8
0	0	0	0	1	1	1	1	16
0	0	0	1	1	1	1	1	32
0	0	1	1	1	1	1	1	64
0	1	1	1	1	1	1	1	128
1	1	1	1	1	1	1	1	256

19.3.2.28 16-Bit Pulse Accumulator B Control Register (PBCTL)

Module Base + 0x0030

	7	6	5	4	3	2	1	0
R	0	PBEN	0	0	0	0	PBOVI	0
W	▨		▨	▨	▨	▨		▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-51. 16-Bit Pulse Accumulator B Control Register (PBCTL)

Read: Anytime

Write: Anytime

All bits reset to zero.

Table 19-37. PBCTL Field Descriptions

Field	Description
6 PBEN	**Pulse Accumulator B System Enable** — PBEN is independent from TEN. With timer disabled, the pulse accumulator can still function unless pulse accumulator is disabled. 0 16-bit pulse accumulator system disabled. 8-bit PAC1 and PAC0 can be enabled when their related enable bits in ICPAR are set. 1 Pulse accumulator B system enabled. The two 8-bit pulse accumulators PAC1 and PAC0 are cascaded to form the PACB 16-bit pulse accumulator B. When PACB is enabled, the PACN1 and PACN0 registers contents are respectively the high and low byte of the PACB. PA1EN and PA0EN control bits in ICPAR have no effect. The PACB shares the input pin with IC0.
1 PBOVI	**Pulse Accumulator B Overflow Interrupt Enable** 0 Interrupt inhibited 1 Interrupt requested if PBOVF is set

19.3.2.29 Pulse Accumulator B Flag Register (PBFLG)

Module Base + 0x0031

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	PBOVF	0
W	▨	▨	▨	▨	▨	▨		▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-52. Pulse Accumulator B Flag Register (PBFLG)

Read: Anytime

Write used in the flag clearing mechanism. Writing a one to the flag clears the flag. Writing a zero will not affect the current status of the bit.

NOTE

When TFFCA = 1, the flag cannot be cleared via the normal flag clearing mechanism (writing a one to the flag). Reference Section 19.3.2.6, "Timer System Control Register 1 (TSCR1)".

All bits reset to zero.

PBFLG indicates when interrupt conditions have occurred. The flag can be cleared via the normal flag clearing mechanism (writing a one to the flag) or via the fast flag clearing mechanism (Reference TFFCA bit in Section 19.3.2.6, "Timer System Control Register 1 (TSCR1)").

Table 19-38. PBFLG Field Descriptions

Field	Description
1 PBOVF	**Pulse Accumulator B Overflow Flag** — This bit is set when the 16-bit pulse accumulator B overflows from 0xFFFF to 0x0000, or when 8-bit pulse accumulator 1 (PAC1) overflows from 0x00FF to 0x0000. When PACMX = 1, PBOVF bit can also be set if 8-bit pulse accumulator 1 (PAC1) reaches 0x00FF and an active edge follows on IC1.

19.3.2.30 8-Bit Pulse Accumulators Holding Registers (PA3H–PA0H)

Module Base + 0x0032

	7	6	5	4	3	2	1	0
R	PA3H7	PA3H6	PA3H5	PA3H4	PA3H3	PA3H2	PA3H1	PA3H0
W								
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-53. 8-Bit Pulse Accumulators Holding Register 3 (PA3H)

Module Base + 0x0033

	7	6	5	4	3	2	1	0
R	PA2H7	PA2H6	PA2H5	PA2H4	PA2H3	PA2H2	PA2H1	PA2H0
W								
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-54. 8-Bit Pulse Accumulators Holding Register 2 (PA2H)

Module Base + 0x0034

	7	6	5	4	3	2	1	0
R	PA1H7	PA1H6	PA1H5	PA1H4	PA1H3	PA1H2	PA1H1	PA1H0
W								
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-55. 8-Bit Pulse Accumulators Holding Register 1 (PA1H)

Module Base + 0x0035

	7	6	5	4	3	2	1	0
R	PA0H7	PA0H6	PA0H5	PA0H4	PA0H3	PA0H2	PA0H1	PA0H0
W								
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-56. 8-Bit Pulse Accumulators Holding Register 0 (PA0H)

Read: Anytime.

Write: Has no effect.

All bits reset to zero.

These registers are used to latch the value of the corresponding pulse accumulator when the related bits in register ICPAR are enabled (see Section 19.4.1.3, “Pulse Accumulators”).

19.3.2.31 Modulus Down-Counter Count Register (MCCNT)

Module Base + 0x0036

	15	14	13	12	11	10	9	8
R W	MCCNT15	MCCNT14	MCCNT13	MCCNT12	MCCNT11	MCCNT10	MCCNT9	MCCNT8
Reset	1	1	1	1	1	1	1	1

Figure 19-57. Modulus Down-Counter Count Register High (MCCNT)

Module Base + 0x0037

	7	6	5	4	3	2	1	0
R W	MCCNT7	MCCNT6	MCCNT5	MCCNT4	MCCNT3	MCCNT2	MCCNT1	MCCNT0
Reset	1	1	1	1	1	1	1	1

Figure 19-58. Modulus Down-Counter Count Register Low (MCCNT)

Read: Anytime

Write: Anytime.

All bits reset to one.

A full access for the counter register will take place in one clock cycle.

NOTE

A separate read/write for high byte and low byte will give different results than accessing them as a word.

If the RDMCL bit in MCCTL register is cleared, reads of the MCCNT register will return the present value of the count register. If the RDMCL bit is set, reads of the MCCNT will return the contents of the load register.

If a 0x0000 is written into MCCNT when LATQ and BUFEN in ICSYS register are set, the input capture and pulse accumulator registers will be latched.

With a 0x0000 write to the MCCNT, the modulus counter will stay at zero and does not set the MCZF flag in MCFLG register.

If the modulus down counter is enabled (MCEN = 1) and modulus mode is enabled (MODMC = 1), a write to MCCNT will update the load register with the value written to it. The count register will not be updated with the new value until the next counter underflow.

If modulus mode is not enabled (MODMC = 0), a write to MCCNT will clear the modulus prescaler and will immediately update the counter register with the value written to it and down-counts to 0x0000 and stops.

The FLMC bit in MCCTL can be used to immediately update the count register with the new value if an immediate load is desired.

19.3.2.32 Timer Input Capture Holding Registers 0–3 (TCxH)

Module Base + 0x0038

	15	14	13	12	11	10	9	8
R	TC15	TC14	TC13	TC12	TC11	TC10	TC9	TC8
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-59. Timer Input Capture Holding Register 0 High (TC0H)

Module Base + 0x0039

	7	6	5	4	3	2	1	0
R	TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-60. Timer Input Capture Holding Register 0 Low (TC0H)

Module Base + 0x003A

	15	14	13	12	11	10	9	8
R	TC15	TC14	TC13	TC12	TC11	TC10	TC9	TC8
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-61. Timer Input Capture Holding Register 1 High (TC1H)

Module Base + 0x003B

	7	6	5	4	3	2	1	0
R	TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-62. Timer Input Capture Holding Register 1 Low (TC1H)

Module Base + 0x003C

	15	14	13	12	11	10	9	8
R	TC15	TC14	TC13	TC12	TC11	TC10	TC9	TC8
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-63. Timer Input Capture Holding Register 2 High (TC2H)

Module Base + 0x003D

	7	6	5	4	3	2	1	0
R	TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-64. Timer Input Capture Holding Register 2 Low (TC2H)

Module Base + 0x003E

	15	14	13	12	11	10	9	8
R	TC15	TC14	TC13	TC12	TC11	TC10	TC9	TC8
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-65. Timer Input Capture Holding Register 3 High (TC3H)

Module Base + 0x003F

	7	6	5	4	3	2	1	0
R	TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 19-66. Timer Input Capture Holding Register 3 Low (TC3H)

Read: Anytime

Write: Has no effect.

All bits reset to zero.

These registers are used to latch the value of the input capture registers TC0–TC3. The corresponding IOSx bits in TIOS should be cleared (see Section 19.4.1.1, “IC Channels”).

19.4 Functional Description

This section provides a complete functional description of the ECT block, detailing the operation of the design from the end user perspective in a number of subsections.

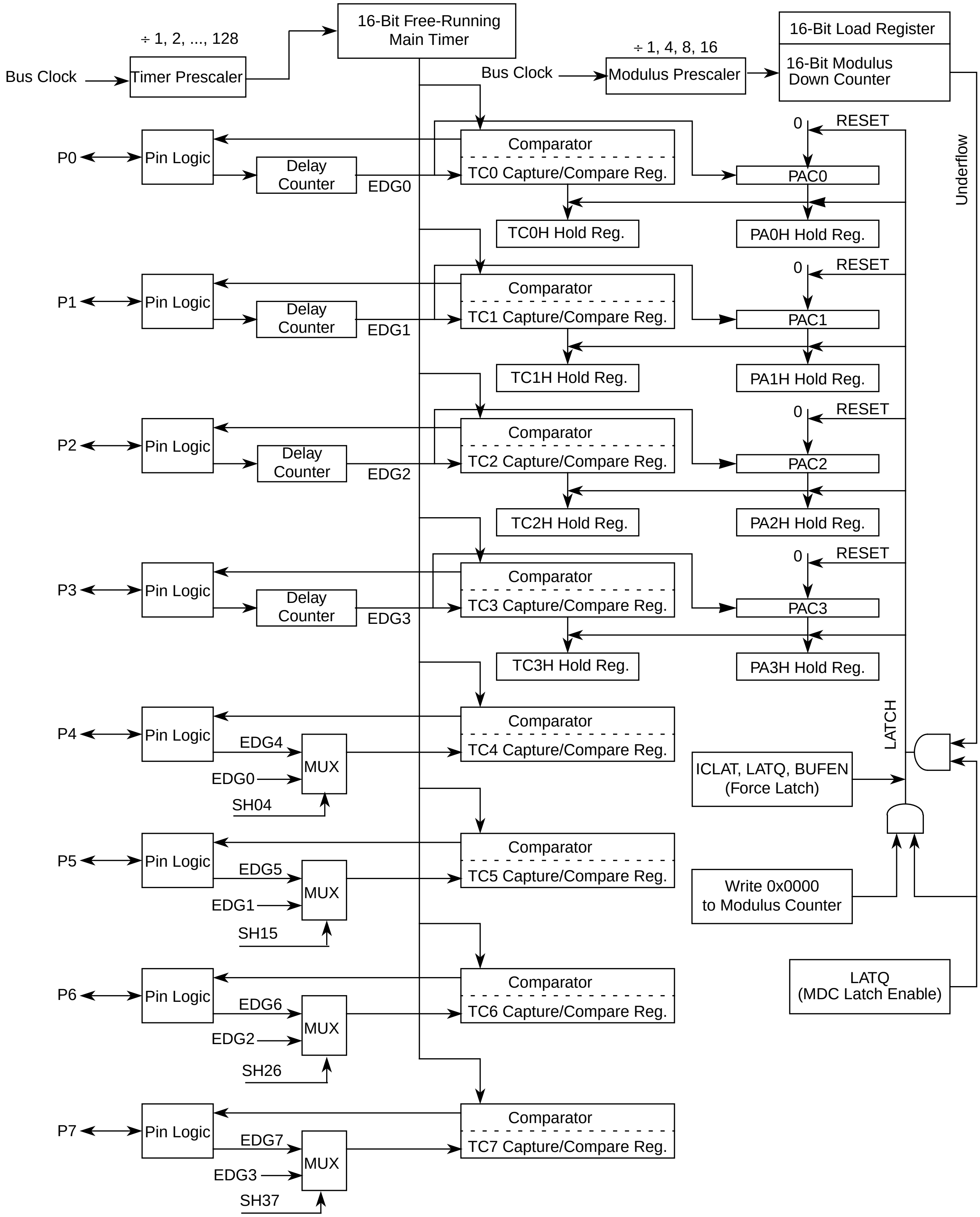


Figure 19-67. Detailed Timer Block Diagram in Latch Mode when PRNT = 0

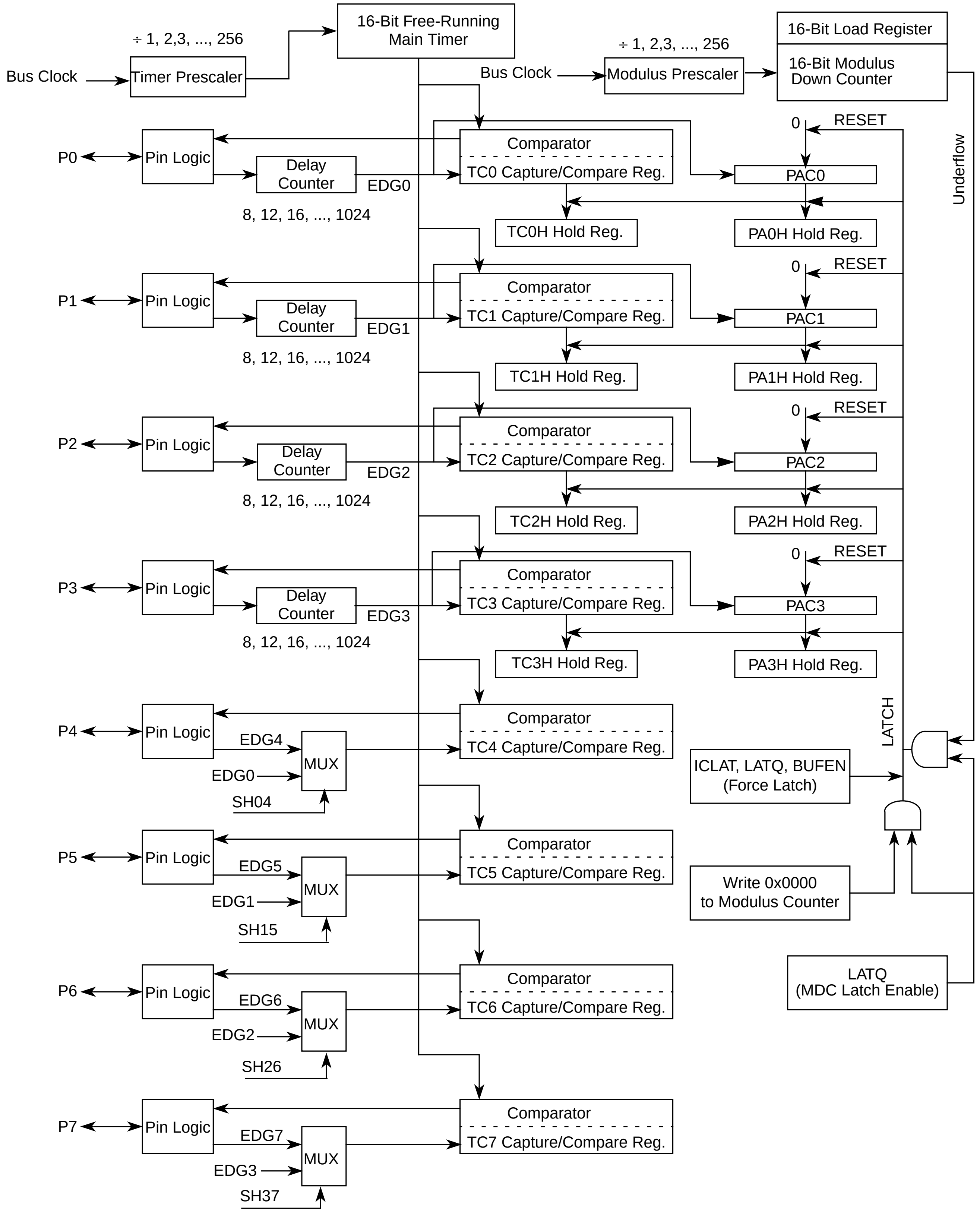


Figure 19-68. Detailed Timer Block Diagram in Latch Mode when PRNT = 1

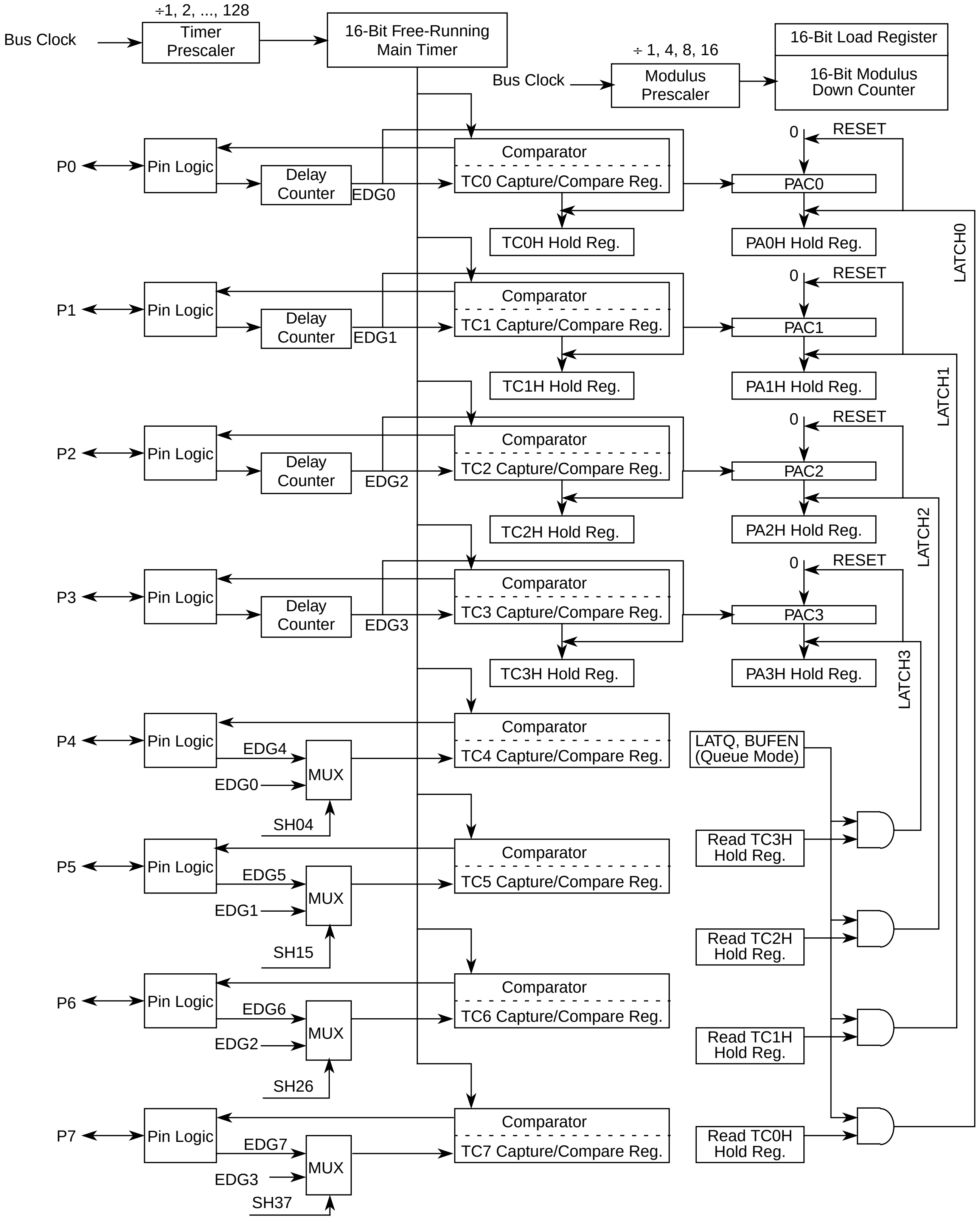


Figure 19-69. Detailed Timer Block Diagram in Queue Mode when PRNT = 0

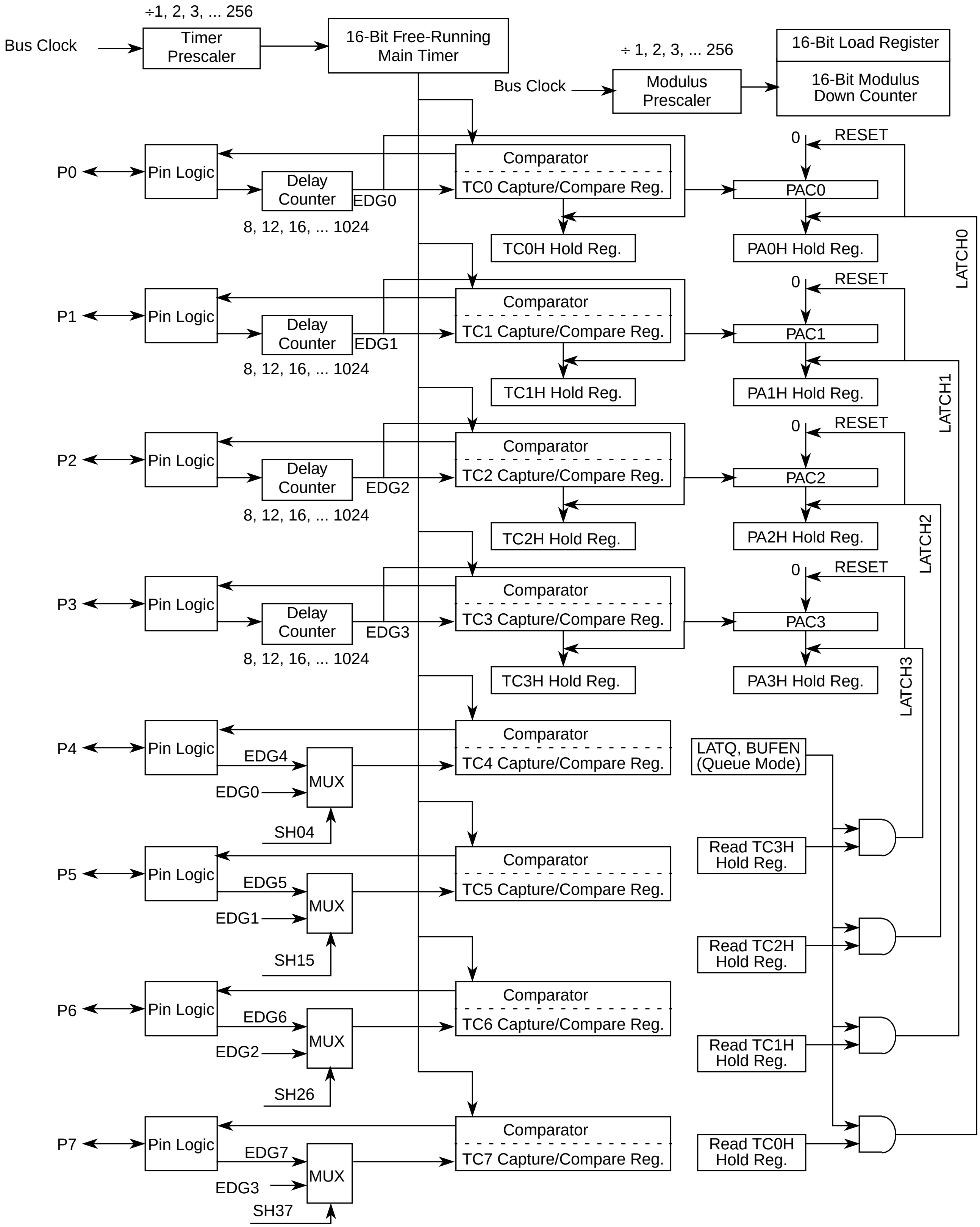


Figure 19-70. Detailed Timer Block Diagram in Queue Mode when PRNT = 1

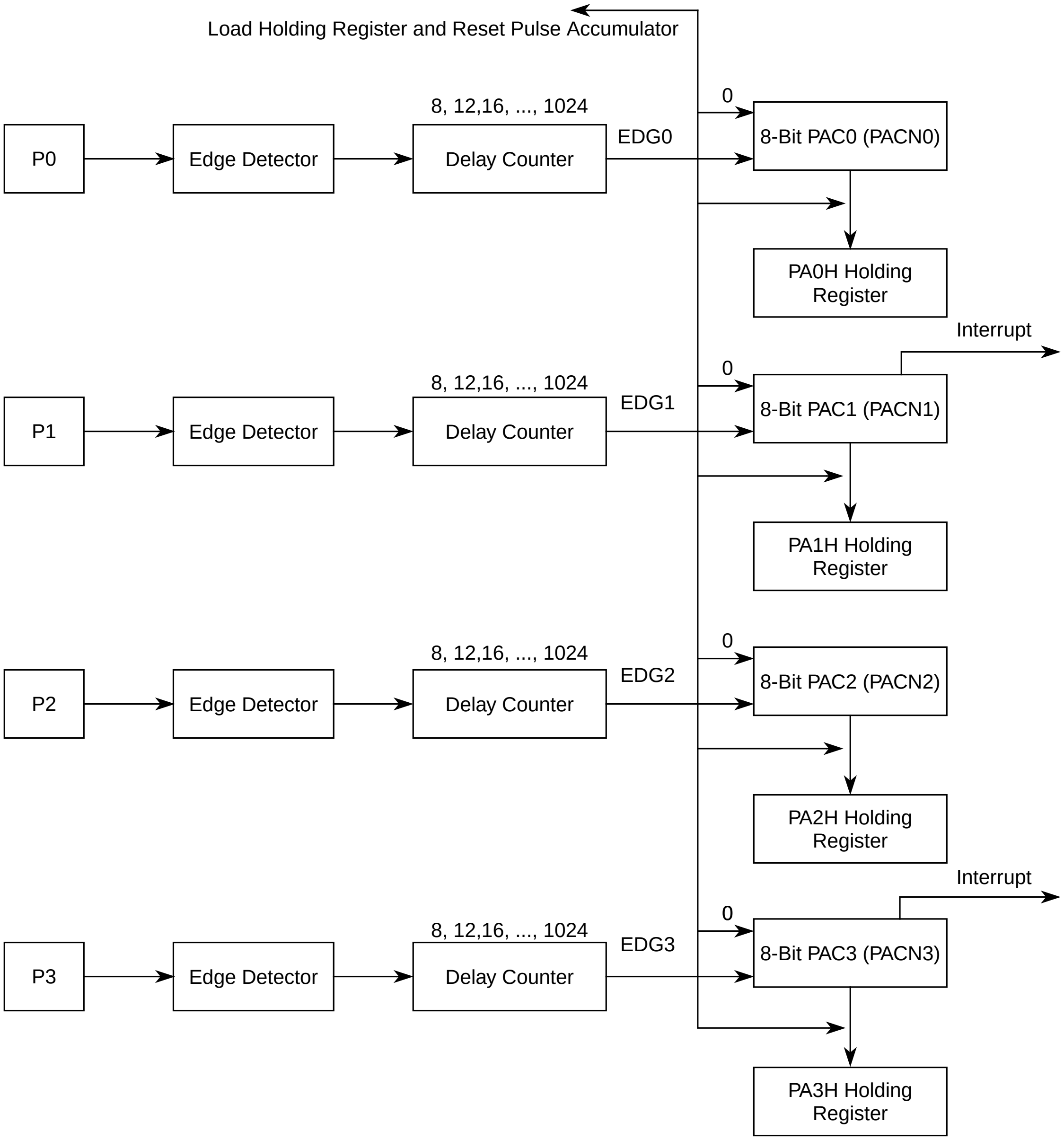


Figure 19-71. 8-Bit Pulse Accumulators Block Diagram

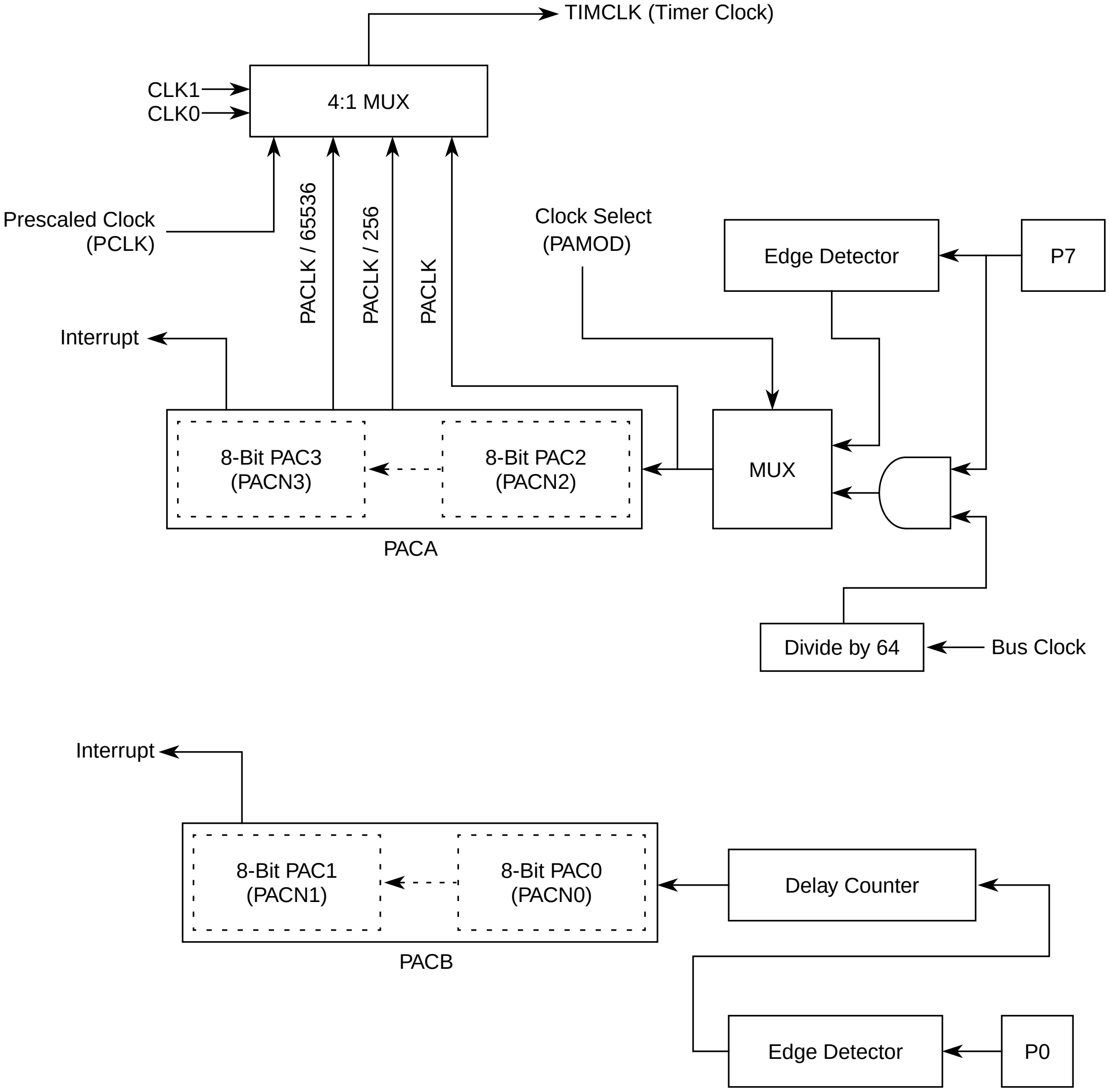


Figure 19-72. 16-Bit Pulse Accumulators Block Diagram

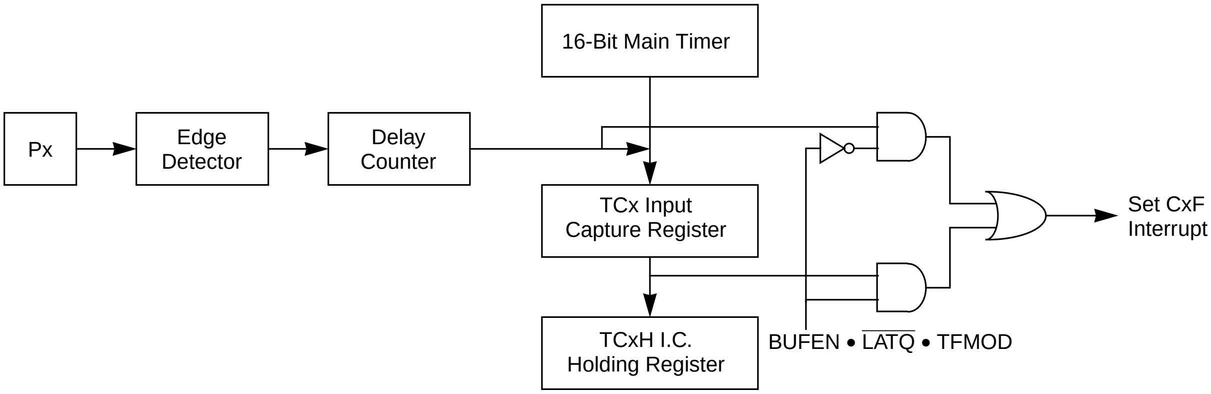


Figure 19-73. Block Diagram for Port 7 with Output Compare/Pulse Accumulator A

19.4.1 Enhanced Capture Timer Modes of Operation

The enhanced capture timer has 8 input capture, output compare (IC/OC) channels, same as on the HC12 standard timer (timer channels TC0 to TC7). When channels are selected as input capture by selecting the IOSx bit in TIOS register, they are called input capture (IC) channels.

Four IC channels (channels 7–4) are the same as on the standard timer with one capture register each that memorizes the timer value captured by an action on the associated input pin.

Four other IC channels (channels 3–0), in addition to the capture register, also have one buffer each called a holding register. This allows two different timer values to be saved without generating any interrupts.

Four 8-bit pulse accumulators are associated with the four buffered IC channels (channels 3–0). Each pulse accumulator has a holding register to memorize their value by an action on its external input. Each pair of pulse accumulators can be used as a 16-bit pulse accumulator.

The 16-bit modulus down-counter can control the transfer of the IC registers and the pulse accumulators contents to the respective holding registers for a given period, every time the count reaches zero.

The modulus down-counter can also be used as a stand-alone time base with periodic interrupt capability.

19.4.1.1 IC Channels

The IC channels are composed of four standard IC registers and four buffered IC channels.

- An IC register is empty when it has been read or latched into the holding register.
- A holding register is empty when it has been read.

19.4.1.1.1 Non-Buffered IC Channels

The main timer value is memorized in the IC register by a valid input pin transition. If the corresponding NOVWx bit of the ICOVW register is cleared, with a new occurrence of a capture, the contents of IC register are overwritten by the new value. If the corresponding NOVWx bit of the ICOVW register is set, the capture register cannot be written unless it is empty. This will prevent the captured value from being overwritten until it is read.

19.4.1.1.2 Buffered IC Channels

There are two modes of operations for the buffered IC channels:

1. IC latch mode (LATQ = 1)

   The main timer value is memorized in the IC register by a valid input pin transition (see Figure 19-67 and Figure 19-68).

   The value of the buffered IC register is latched to its holding register by the modulus counter for a given period when the count reaches zero, by a write 0x0000 to the modulus counter or by a write to ICLAT in the MCCTL register.

   If the corresponding NOVWx bit of the ICOVW register is cleared, with a new occurrence of a capture, the contents of IC register are overwritten by the new value. In case of latching, the contents of its holding register are overwritten.

If the corresponding NOVWx bit of the ICOVW register is set, the capture register or its holding register cannot be written by an event unless they are empty (see Section 19.4.1.1, “IC Channels”). This will prevent the captured value from being overwritten until it is read or latched in the holding register.

2. IC Queue Mode (LATQ = 0)

   The main timer value is memorized in the IC register by a valid input pin transition (see Figure 19-69 and Figure 19-70).

   If the corresponding NOVWx bit of the ICOVW register is cleared, with a new occurrence of a capture, the value of the IC register will be transferred to its holding register and the IC register memorizes the new timer value.

   If the corresponding NOVWx bit of the ICOVW register is set, the capture register or its holding register cannot be written by an event unless they are empty (see Section 19.4.1.1, “IC Channels”).

   if the TFMOD bit of the ICSYS register is set,the timer flags C3F--C0F in TFLG register are set only when a latch on the corresponding holding register occurs,after C3F--C0F are set,user should clear flag C3F--C0F,then read TCx and TCxH to make TCx and TCxH be empty.

   In queue mode, reads of the holding register will latch the corresponding pulse accumulator value to its holding register.

19.4.1.1.3 Delayed IC Channels

There are four delay counters in this module associated with IC channels 0–3. The use of this feature is explained in the diagram and notes below.

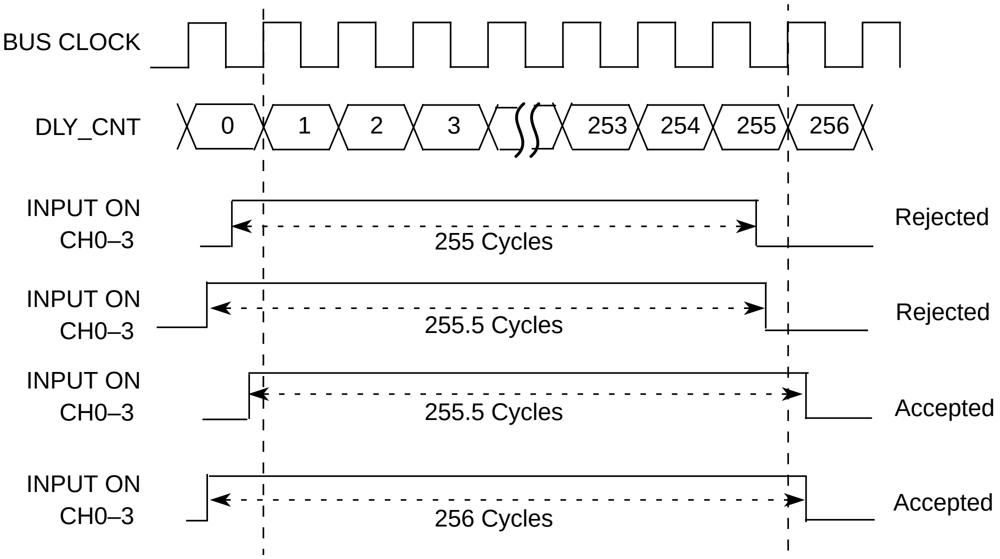


Figure 19-74. Channel Input Validity with Delay Counter Feature

In Figure 19-74 a delay counter value of 256 bus cycles is considered.

1. Input pulses with a duration of (DLY_CNT – 1) cycles or shorter are rejected.
2. Input pulses with a duration between (DLY_CNT – 1) and DLY_CNT cycles may be rejected or accepted, depending on their relative alignment with the sample points.

3. Input pulses with a duration between (DLY_CNT – 1) and DLY_CNT cycles may be rejected or accepted, depending on their relative alignment with the sample points.
4. Input pulses with a duration of DLY_CNT or longer are accepted.

19.4.1.2 OC Channel Initialization

An internal compare channel whose output drives OCx may be programmed before the timer drives the output compare state (OCx). The required output of the compare logic can be disconnected from the pin, leaving it driven by the GP IO port, by setting the appropriate OCPDx bit before enabling the output compare channel (by default the OCPD bits are cleared which would enable the output compare logic to drive the pin as soon as the timer output compare channel is enabled). The desired initial state can then be configured in the internal output compare logic by forcing a compare action with the logic disconnected from the IO (by writing a one to CFORCx bit with TIOSx, OCPDx and TEN bits set to one). Clearing the output compare disconnect bit (OCPDx) will then allow the internal compare logic to drive the programmed state to OCx. This allows a glitch free switching between general purpose I/O and timer output functionality.

19.4.1.3 Pulse Accumulators

There are four 8-bit pulse accumulators with four 8-bit holding registers associated with the four IC buffered channels 3–0. A pulse accumulator counts the number of active edges at the input of its channel.

The minimum pulse width for the PAI input is greater than two bus clocks.The maximum input frequency on the pulse accumulator channel is one half the bus frequency or Eclk.

The user can prevent the 8-bit pulse accumulators from counting further than 0x00FF by utilizing the PACMX control bit in the ICSYS register. In this case, a value of 0x00FF means that 255 counts or more have occurred.

Each pair of pulse accumulators can be used as a 16-bit pulse accumulator (see Figure 19-72).

Pulse accumulator B operates only as an event counter, it does not feature gated time accumulation mode. The edge control for pulse accumulator B as a 16-bit pulse accumulator is defined by TCTL4[1:0].

To operate the 16-bit pulse accumulators A and B (PACA and PACB) independently of input capture or output compare 7 and 0 respectively, the user must set the corresponding bits: IOSx = 1, OMx = 0, and OLx = 0. OC7M7 or OC7M0 in the OC7M register must also be cleared.

There are two modes of operation for the pulse accumulators:

- Pulse accumulator latch mode

  The value of the pulse accumulator is transferred to its holding register when the modulus down-counter reaches zero, a write 0x0000 to the modulus counter or when the force latch control bit ICLAT is written.

  At the same time the pulse accumulator is cleared.
- Pulse accumulator queue mode

  When queue mode is enabled, reads of an input capture holding register will transfer the contents of the associated pulse accumulator to its holding register.

At the same time the pulse accumulator is cleared.

19.4.1.4 Modulus Down-Counter

The modulus down-counter can be used as a time base to generate a periodic interrupt. It can also be used to latch the values of the IC registers and the pulse accumulators to their holding registers.

The action of latching can be programmed to be periodic or only once.

19.4.1.5 Precision Timer

By enabling the PRNT bit of the TSCR1 register, the performance of the timer can be enhanced. In this case, it is possible to set additional prescaler settings for the main timer counter and modulus down counter and enhance delay counter settings compared to the settings in the present ECT timer.

19.4.1.6 Flag Clearing Mechanisms

The flags in the ECT can be cleared one of two ways:

1. Normal flag clearing mechanism (TFFCA = 0)

   Any of the ECT flags can be cleared by writing a one to the flag.

2. Fast flag clearing mechanism (TFFCA = 1)

   With the timer fast flag clear all (TFFCA) enabled, the ECT flags can only be cleared by accessing the various registers associated with the ECT modes of operation as described below. The flags cannot be cleared via the normal flag clearing mechanism. This fast flag clearing mechanism has the advantage of eliminating the software overhead required by a separate clear sequence. Extra care must be taken to avoid accidental flag clearing due to unintended accesses.

   - Input capture

     A read from an input capture channel register causes the corresponding channel flag, CxF, to be cleared in the TFLG1 register.

   - Output compare

     A write to the output compare channel register causes the corresponding channel flag, CxF, to be cleared in the TFLG1 register.

   - Timer counter

     Any access to the TCNT register clears the TOF flag in the TFLG2 register.

   - Pulse accumulator A

     Any access to the PACN3 and PACN2 registers clears the PAOVF and PAIF flags in the PAFLG register.

   - Pulse accumulator B

     Any access to the PACN1 and PACN0 registers clears the PBOVF flag in the PBFLG register.

   - Modulus down counter

     Any access to the MCCNT register clears the MCZF flag in the MCFLG register.

19.4.2 Reset

The reset state of each individual bit is listed within the register description section (Section 19.3, “Memory Map and Register Definition”) which details the registers and their bit-fields.

19.4.3 Interrupts

This section describes interrupts originated by the ECT block. The MCU must service the interrupt requests. Table 19-39 lists the interrupts generated by the ECT to communicate with the MCU.

Table 19-39. ECT Interrupts

Interrupt Source	Description
Timer channel 7–0	Active high timer channel interrupts 7–0
Modulus counter underflow	Active high modulus counter interrupt
Pulse accumulator B overflow	Active high pulse accumulator B interrupt
Pulse accumulator A input	Active high pulse accumulator A input interrupt
Pulse accumulator A overflow	Pulse accumulator overflow interrupt
Timer overflow	Timer 0verflow interrupt

The ECT only originates interrupt requests. The following is a description of how the module makes a request and how the MCU should acknowledge that request. The interrupt vector offset and interrupt number are chip dependent.

19.4.3.1 Channel [7:0] Interrupt

This active high output will be asserted by the module to request a timer channel 7–0 interrupt to be serviced by the system controller.

19.4.3.2 Modulus Counter Interrupt

This active high output will be asserted by the module to request a modulus counter underflow interrupt to be serviced by the system controller.

19.4.3.3 Pulse Accumulator B Overflow Interrupt

This active high output will be asserted by the module to request a timer pulse accumulator B overflow interrupt to be serviced by the system controller.

19.4.3.4 Pulse Accumulator A Input Interrupt

This active high output will be asserted by the module to request a timer pulse accumulator A input interrupt to be serviced by the system controller.

19.4.3.5 Pulse Accumulator A Overflow Interrupt

This active high output will be asserted by the module to request a timer pulse accumulator A overflow interrupt to be serviced by the system controller.

19.4.3.6 Timer Overflow Interrupt

This active high output will be asserted by the module to request a timer overflow interrupt to be serviced by the system controller.

Chapter 20
Voltage Regulator (S12VREG3V3V5)

Revision History

Version Number	Revision Date	Effective Date	Author	Description of Changes
V 5.03	09 Sep 2004	09 Sep 2004		Corrected Typos.
V 5.04	24 Nov 2004	24 Nov 2004		Updated Section 20.4.7 & Table 20-6 to be easier understood. .
V 5.05	1 Apr 2005	1 Apr 2005		Corrected Section regarding the non usage of VSSR. For info on VSSR see Section 20.2.1.

20.1 Introduction

Module VREG_3V3 is a dual output voltage regulator that provides two separate 2.5V (typical) supplies differing in the amount of current that can be sourced. The regulator input voltage range is from 3.3V up to 5V (typical).

20.1.1 Features

Module VREG_3V3 includes these distinctive features:

- Two parallel, linear voltage regulators
  - Bandgap reference
- Low-voltage detect (LVD) with low-voltage interrupt (LVI)
- Power-on reset (POR)
- Low-voltage reset (LVR)
- Autonomous periodical interrupt (API)

20.1.2 Modes of Operation

There are three modes VREG_3V3 can operate in:

1. Full performance mode (FPM) (MCU is not in stop mode)

The regulator is active, providing the nominal supply voltage of 2.5 V with full current sourcing capability at both outputs. Features LVD (low-voltage detect), LVR (low-voltage reset), and POR (power-on reset) are available. The API is available.

2. Reduced power mode (RPM) (MCU is in stop mode)

   The purpose is to reduce power consumption of the device. The output voltage may degrade to a lower value than in full performance mode, additionally the current sourcing capability is substantially reduced. Only the POR is available in this mode, LVD and LVR are disabled. The API is available.

3. Shutdown mode

   Controlled by VREGEN (see device level specification for connectivity of VREGEN).

   This mode is characterized by minimum power consumption. The regulator outputs are in a high-impedance state, only the POR feature is available, LVD and LVR are disabled. The API internal RC oscillator clock is not available.

   This mode must be used to disable the chip internal regulator VREG_3V3, i.e., to bypass the VREG_3V3 to use external supplies.

20.1.3 Block Diagram

Figure 20-1 shows the function principle of VREG_3V3 by means of a block diagram. The regulator core REG consists of two parallel subblocks, REG1 and REG2, providing two independent output voltages.

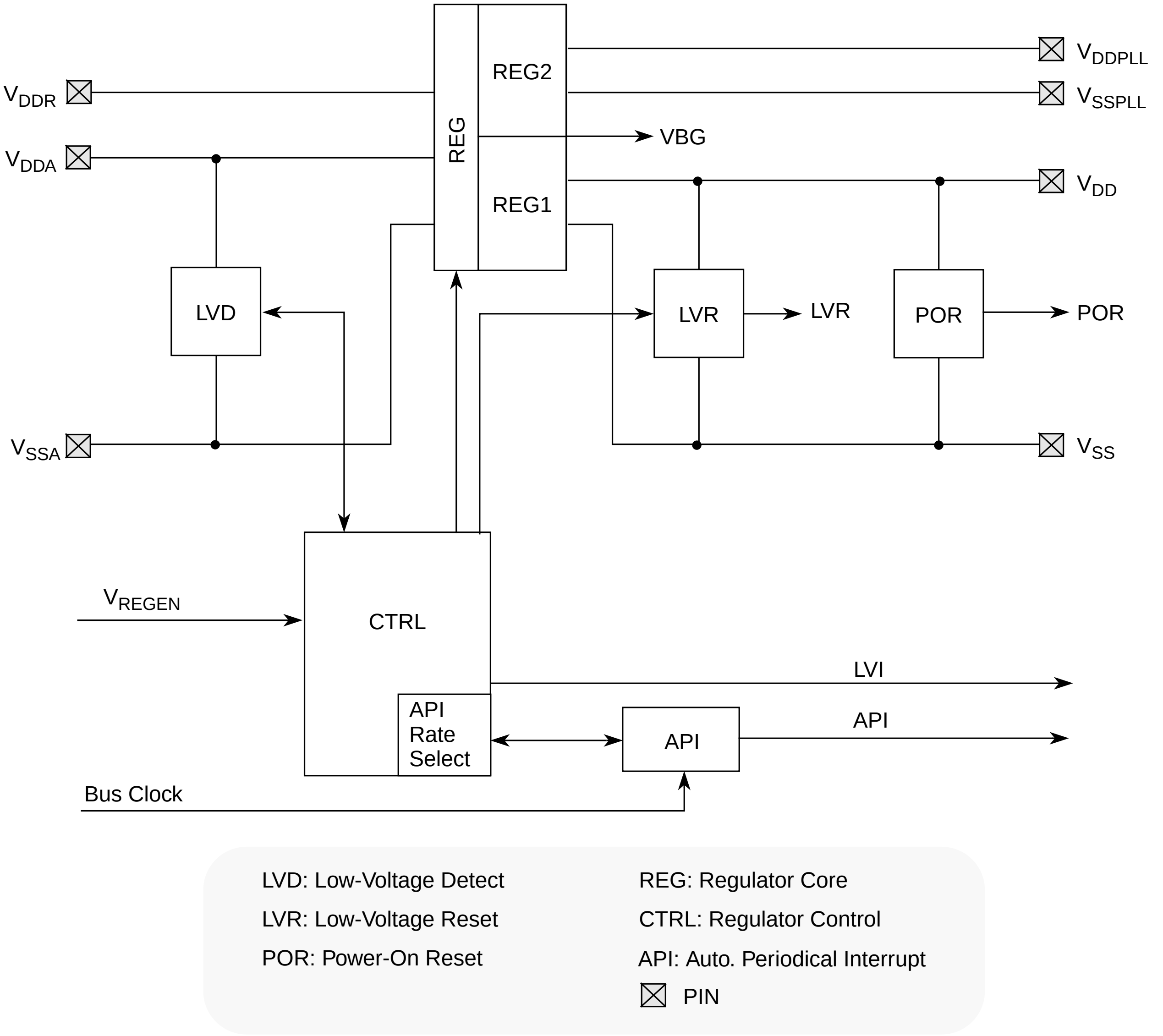


Figure 20-1. VREG_3V3 Block Diagram

20.2 External Signal Description

Due to the nature of VREG_3V3 being a voltage regulator providing the chip internal power supply voltages, most signals are power supply signals connected to pads.

Table 20-1 shows all signals of VREG_3V3 associated with pins.

Table 20-1. Signal Properties

Name	Function	Reset State	Pull Up
V_{DDR}	Power input (positive supply)	—	—
V_{DDA}	Quiet input (positive supply)	—	—
V_{SSA}	Quiet input (ground)	—	—
V_{DD}	Primary output (positive supply)	—	—
V_{SS}	Primary output (ground)	—	—
V_{DDPLL}	Secondary output (positive supply)	—	—
V_{SSPLL}	Secondary output (ground)	—	—
V_{REGEN} (optional)	Optional Regulator Enable	—	—

NOTE

Check device level specification for connectivity of the signals.

20.2.1 VDDR — Regulator Power Input Pins

Signal V_{DDR} is the power input of VREG_3V3. All currents sourced into the regulator loads flow through this pin. A chip external decoupling capacitor (100 nF...220 nF, X7R ceramic) between V_{DDR} and V_{SSR} (if V_{SSR} is not available V_{SS}) can smooth ripple on V_{DDR}.

For entering shutdown mode, pin V_{DDR} should also be tied to ground on devices without VREGEN pin.

20.2.2 VDDA, VSSA — Regulator Reference Supply Pins

Signals V_{DDA}/$V_{SSA,}$ which are supposed to be relatively quiet, are used to supply the analog parts of the regulator. Internal precision reference circuits are supplied from these signals. A chip external decoupling capacitor (100 nF...220 nF, X7R ceramic) between V_{DDA} and V_{SSA} can further improve the quality of this supply.

20.2.3 VDD, VSS — Regulator Output1 (Core Logic) Pins

Signals V_{DD}/V_{SS} are the primary outputs of VREG_3V3 that provide the power supply for the core logic. These signals are connected to device pins to allow external decoupling capacitors (100 nF...220 nF, X7R ceramic).

In shutdown mode an external supply driving V_{DD}/V_{SS} can replace the voltage regulator.

20.2.4 VDDPLL, VSSPLL — Regulator Output2 (PLL) Pins

Signals V_{DDPLL}/V_{SSPLL} are the secondary outputs of VREG_3V3 that provide the power supply for the PLL and oscillator. These signals are connected to device pins to allow external decoupling capacitors (100 nF...220 nF, X7R ceramic).

In shutdown mode, an external supply driving V_{DDPLL}/V_{SSPLL} can replace the voltage regulator.

20.2.5 V_{REGEN} — Optional Regulator Enable Pin

This optional signal is used to shutdown VREG_3V3. In that case, V_{DD}/V_{SS} and V_{DDPLL}/V_{SSPLL} must be provided externally. Shutdown mode is entered with VREGEN being low. If VREGEN is high, the VREG_3V3 is either in full peformance mode or in reduced power mode.

For the connectivity of VREGEN, see device specification.

NOTE

Switching from FPM or RPM to shutdown of VREG_3V3 and vice versa is not supported while MCU is powered.

20.3 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in VREG_3V3.

If enabled in the system, the VREG_3V3 will abort all read and write accesses to reserved registers within it’s memory slice.

20.3.1 Module Memory Map

Table 20-2 provides an overview of all used registers.

Table 20-2. Memory Map

Address Offset	Use	Access
0x0000	HT Control Register (VREGHTCL)	—
0x0001	Control Register (VREGCTRL)	R/W
0x0002	Autonomous Periodical Interrupt Control Register (VREGAPICL)	R/W
0x0003	Autonomous Periodical Interrupt Trimming Register (VREGAPITR)	R/W
0x0004	Autonomous Periodical Interrupt Period High (VREGAPIRH)	R/W
0x0005	Autonomous Periodical Interrupt Period Low (VREGAPIRL)	R/W
0x0006	Reserved 06	—
0x0007	Reserved 07	—

20.3.2 Register Descriptions

This section describes all the VREG_3V3 registers and their individual bits.

20.3.2.1 HT Control Register (VREGHTCL)

The VREGHTCL is reserved for test purposes. This register should not be written.

Module Base + 0x0000

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 20-2. HT Control Register (VREGHTCL)

20.3.2.2 Control Register (VREGCTRL)

The VREGCTRL register allows the configuration of the VREG_3V3 low-voltage detect features.

Module Base + 0x0001

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	LVDS	LVIE	LVIF
W	▨	▨	▨	▨	▨	▨		
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 20-3. Control Register (VREGCTRL)

Table 20-3. VREGCTRL Field Descriptions

Field	Description
2 LVDS	**Low-Voltage Detect Status Bit** — This read-only status bit reflects the input voltage. Writes have no effect. 0 Input voltage V_{DDA} is above level V_{LVID} or RPM or shutdown mode. 1 Input voltage V_{DDA} is below level V_{LVIA} and FPM.
1 LVIE	**Low-Voltage Interrupt Enable Bit** 0 Interrupt request is disabled. 1 Interrupt will be requested whenever LVIF is set.
0 LVIF	**Low-Voltage Interrupt Flag** — LVIF is set to 1 when LVDS status bit changes. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (LVIE = 1), LVIF causes an interrupt request. 0 No change in LVDS bit. 1 LVDS bit has changed. **Note:** On entering the reduced power mode the LVIF is not cleared by the VREG_3V3.

20.3.2.3 Autonomous Periodical Interrupt Control Register (VREGAPICL)

The VREGAPICL register allows the configuration of the VREG_3V3 autonomous periodical interrupt features.

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R	APICLK	0	0	0	0	APIFE	APIE	APIF
W		▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 20-4. Autonomous Periodical Interrupt Control Register (VREGAPICL)

Table 20-4. VREGAPICL Field Descriptions

Field	Description
7 APICLK	**Autonomous Periodical Interrupt Clock Select Bit** — Selects the clock source for the API. Writable only if APIFE = 0; APICLK cannot be changed if APIFE is set by the same write operation. 0 Autonomous periodical interrupt clock used as source. 1 Bus clock used as source.
2 APIFE	**Autonomous Periodical Interrupt Feature Enable Bit** — Enables the API feature and starts the API timer when set. 0 Autonomous periodical interrupt is disabled. 1 Autonomous periodical interrupt is enabled and timer starts running.
1 APIE	**Autonomous Periodical Interrupt Enable Bit** 0 API interrupt request is disabled. 1 API interrupt will be requested whenever APIF is set.
0 APIF	**Autonomous Periodical Interrupt Flag** — APIF is set to 1 when the in the API configured time has elapsed. This flag can only be cleared by writing a 1 to it. Clearing of the flag has precedence over setting. Writing a 0 has no effect. If enabled (APIE = 1), APIF causes an interrupt request. 0 API timeout has not yet occurred. 1 API timeout has occurred.

20.3.2.4 Autonomous Periodical Interrupt Trimming Register (VREGAPITR)

The VREGAPITR register allows to trim the API timeout period.

Module Base + 0x0003

	7	6	5	4	3	2	1	0
R	APITR5	APITR4	APITR3	APITR2	APITR1	APITR0	0	0
W							▨	▨
Reset	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0	0

[1] Reset value is either 0 or preset by factory. See Section 1 (Device Overview) for details.

▨ = Unimplemented or Reserved

Figure 20-5. Autonomous Periodical Interrupt Trimming Register (VREGAPITR)

Table 20-5. VREGAPITR Field Descriptions

Field	Description
7–2 APITR[5:0]	**Autonomous Periodical Interrupt Period Trimming Bits** — See Table 20-6 for trimming effects.

Table 20-6. Trimming Effect of APIT

Bit	Trimming Effect
APITR[5]	Increases period
APITR[4]	Decreases period less than APITR[5] increased it
APITR[3]	Decreases period less than APITR[4]
APITR[2]	Decreases period less than APITR[3]
APITR[1]	Decreases period less than APITR[2]
APITR[0]	Decreases period less than APITR[1]

20.3.2.5 Autonomous Periodical Interrupt Rate High and Low Register (VREGAPIRH / VREGAPIRL)

The VREGAPIRH and VREGAPIRL register allows the configuration of the VREG_3V3 autonomous periodical interrupt rate.

Module Base + 0x0004

	7	6	5	4	3	2	1	0
R	0	0	0	0	APIR11	APIR10	APIR9	APIR8
W	▨	▨	▨	▨				
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 20-6. Autonomous Periodical Interrupt Rate High Register (VREGAPIRH)

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R W	APIR7	APIR6	APIR5	APIR4	APIR3	APIR2	APIR1	APIR0
Reset	0	0	0	0	0	0	0	0

Figure 20-7. Autonomous Periodical Interrupt Rate Low Register (VREGAPIRL)

Table 20-7. VREGAPIRH / VREGAPIRL Field Descriptions

Field	Description
11-0 APIR[11:0]	**Autonomous Periodical Interrupt Rate Bits** — These bits define the timeout period of the API. See Table 20-8 for details of the effect of the autonomous periodical interrupt rate bits. Writable only if APIFE = 0 of VREGAPICL register.

Table 20-8. Selectable Autonomous Periodical Interrupt Periods

APICLK	APIR[11:0]	Selected Period
0	000	0.2 ms[1]
0	001	0.4 ms[1]
0	002	0.6 ms[1]
0	003	0.8 ms[1]
0	004	1.0 ms[1]
0	005	1.2 ms[1]
0	.....	.....
0	FFD	818.8 ms[1]
0	FFE	819 ms[1]
0	FFF	819.2 ms[1]
1	000	2 * bus clock period
1	001	4 * bus clock period
1	002	6 * bus clock period
1	003	8 * bus clock period
1	004	10 * bus clock period
1	005	12 * bus clock period
1	.....	.....
1	FFD	8188 * bus clock period
1	FFE	8190 * bus clock period
1	FFF	8192 * bus clock period

[1] When trimmed within specified accuracy. See electrical specifications for details.

You can calculate the selected period depending of APICLK as:

$$\text{Period} = 2*(\text{APIR}[11:0] + 1) * 0.1\text{ ms} \quad \text{or} \quad \text{period} = 2*(\text{APIR}[11:0] + 1) * \text{bus clock period}$$

20.3.2.6 Reserved 06

The Reserved 06 is reserved for test purposes.

Module Base + 0x0006

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 20-8. Reserved 06

20.3.2.7 Reserved 07

The Reserved 07 is reserved for test purposes.

Module Base + 0x0007

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 20-9. Reserved 07

20.4 Functional Description

20.4.1 General

Module VREG_3V3 is a voltage regulator, as depicted in Figure 20-1. The regulator functional elements are the regulator core (REG), a low-voltage detect module (LVD), a control block (CTRL), a power-on reset module (POR), and a low-voltage reset module (LVR).

20.4.2 Regulator Core (REG)

Respectively its regulator core has two parallel, independent regulation loops (REG1 and REG2) that differ only in the amount of current that can be delivered.

The regulator is a linear regulator with a bandgap reference when operated in full peformance mode. It acts as a voltage clamp in reduced power mode. All load currents flow from input V_{DDR} to V_{SS} or V_{SSPLL}. The reference circuits are supplied by V_{DDA} and V_{SSA}.

20.4.2.1 Full Performance Mode

In full peformance mode, the output voltage is compared with a reference voltage by an operational amplifier. The amplified input voltage difference drives the gate of an output transistor.

20.4.2.2 Reduced Power Mode

In reduced power mode, the gate of the output transistor is connected directly to a reference voltage to reduce power consumption.

20.4.3 Low-Voltage Detect (LVD)

Subblock LVD is responsible for generating the low-voltage interrupt (LVI). LVD monitors the input voltage (V_{DDA}–V_{SSA}) and continuously updates the status flag LVDS. Interrupt flag LVIF is set whenever status flag LVDS changes its value. The LVD is available in FPM and is inactive in reduced power mode or shutdown mode.

20.4.4 Power-On Reset (POR)

This functional block monitors V_{DD}. If V_{DD} is below V_{PORD}, POR is asserted; if V_{DD} exceeds V_{PORD}, the POR is deasserted. POR asserted forces the MCU into Reset. POR Deasserted will trigger the power-on sequence.

20.4.5 Low-Voltage Reset (LVR)

Block LVR monitors the primary output voltage V_{DD}. If it drops below the assertion level (V_{LVRA}) signal, LVR asserts; if V_{DD} rises above the deassertion level (V_{LVRD}) signal, LVR deasserts. The LVR function is available only in full peformance mode.

20.4.6 Regulator Control (CTRL)

This part contains the register block of VREG_3V3 and further digital functionality needed to control the operating modes. CTRL also represents the interface to the digital core logic.

20.4.7 Autonomous Periodical Interrupt (API)

Subblock API can generate periodical interrupts independent of the clock source of the MCU. To enable the timer, the bit APIFE needs to be set.

The API timer is either clocked by a trimmable internal RC oscillator or the bus clock. Timer operation will freeze when MCU clock source is selected and bus clock is turned off. See CRG specification for details. The clock source can be selected with bit APICLK. APICLK can only be written when APIFE is not set.

The APIR[11:0] bits determine the interrupt period. APIR[11:0] can only be written when APIFE is cleared. As soon as APIFE is set, the timer starts running for the period selected by APIR[11:0] bits. When the configured time has elapsed, the flag APIF is set. An interrupt, indicated by flag APIF = 1, is triggered if interrupt enable bit APIE = 1. The timer is started automatically again after it has set APIF.

The procedure to change APICLK or APIR[11:0] is first to clear APIFE, then write to APICLK or APIR[11:0], and afterwards set APIFE.

The API Trimming bits APITR[5:0] must be set so the minimum period equals 0.2 ms if stable frequency is desired.

See Table 20-6 for the trimming effect of APITR.

NOTE

> The first period after enabling the counter by APIFE might be reduced.
>
> The API internal RC oscillator clock is not available if VREG_3V3 is in Shutdown Mode.

20.4.8 Resets

This section describes how VREG_3V3 controls the reset of the MCU.The reset values of registers and signals are provided in Section 20.3, “Memory Map and Register Definition”. Possible reset sources are listed in Table 20-9.

Table 20-9. Reset Sources

Reset Source	Local Enable
Power-on reset	Always active
Low-voltage reset	Available only in full peformance mode

20.4.9 Description of Reset Operation

20.4.9.1 Power-On Reset (POR)

During chip power-up the digital core may not work if its supply voltage V_{DD} is below the POR deassertion level (V_{PORD}). Therefore, signal POR, which forces the other blocks of the device into reset, is kept high until V_{DD} exceeds V_{PORD}. The MCU will run the start-up sequence after POR deassertion. The power-on reset is active in all operation modes of VREG_3V3.

20.4.9.2 Low-Voltage Reset (LVR)

For details on low-voltage reset, see Section 20.4.5, “Low-Voltage Reset (LVR)”.

20.4.10 Interrupts

This section describes all interrupts originated by VREG_3V3.

The interrupt vectors requested by VREG_3V3 are listed in Table 20-10. Vector addresses and interrupt priorities are defined at MCU level.

Table 20-10. Interrupt Vectors

Interrupt Source	Local Enable
Low-voltage interrupt (LVI)	LVIE = 1; available only in full peformance mode

Table 20-10. Interrupt Vectors

Interrupt Source	Local Enable
Autonomous periodical interrupt (API)	APIE = 1

20.4.10.1 Low-Voltage Interrupt (LVI)

In FPM, VREG_3V3 monitors the input voltage V_{DDA}. Whenever V_{DDA} drops below level $V_{LVIA,}$ the status bit LVDS is set to 1. On the other hand, LVDS is reset to 0 when V_{DDA} rises above level V_{LVID}. An interrupt, indicated by flag LVIF = 1, is triggered by any change of the status bit LVDS if interrupt enable bit LVIE = 1.

NOTE

On entering the reduced power mode, the LVIF is not cleared by the VREG_3V3.

20.4.10.2 Autonomous Periodical Interrupt (API)

As soon as the configured timeout period of the API has elapsed, the APIF bit is set. An interrupt, indicated by flag APIF = 1, is triggered if interrupt enable bit APIE = 1.

Chapter 21
Background Debug Module (S12XBDMV2)

Table 21-1. Revision History

Revision Number	Revision Date	Sections Affected	Description of Changes
V02.00	07 Mar 2006		- First version of S12XBDMV2
V02.01	14 May 2008		- Introduced standardized Revision History Table

21.1 Introduction

This section describes the functionality of the background debug module (BDM) sub-block of the HCS12X core platform.

The background debug module (BDM) sub-block is a single-wire, background debug system implemented in on-chip hardware for minimal CPU intervention. All interfacing with the BDM is done via the BKGD pin.

The BDM has enhanced capability for maintaining synchronization between the target and host while allowing more flexibility in clock rates. This includes a sync signal to determine the communication rate and a handshake signal to indicate when an operation is complete. The system is backwards compatible to the BDM of the S12 family with the following exceptions:

- TAGGO command no longer supported by BDM
- External instruction tagging feature now part of DBG module
- BDM register map and register content extended/modified
- Global page access functionality
- Enabled but not active out of reset in emulation modes (if modes available)
- CLKSW bit set out of reset in emulation modes (if modes available).
- Family ID readable from firmware ROM at global address 0x7FFF0F (value for HCS12X devices is 0xC1)

21.1.1 Features

The BDM includes these distinctive features:

- Single-wire communication with host development system
- Enhanced capability for allowing more flexibility in clock rates
- SYNC command to determine communication rate
- GO_UNTIL command
- Hardware handshake protocol to increase the performance of the serial communication

- Active out of reset in special single chip mode
- Nine hardware commands using free cycles, if available, for minimal CPU intervention
- Hardware commands not requiring active BDM
- 14 firmware commands execute from the standard BDM firmware lookup table
- Software control of BDM operation during wait mode
- Software selectable clocks
- Global page access functionality
- Enabled but not active out of reset in emulation modes (if modes available)
- CLKSW bit set out of reset in emulation modes (if modes available).
- When secured, hardware commands are allowed to access the register space in special single chip mode, if the non-volatile memory erase test fail.
- Family ID readable from firmware ROM at global address 0x7FFF0F (value for HCS12X devices is 0xC1)
- BDM hardware commands are operational until system stop mode is entered (all bus masters are in stop mode)

21.1.2 Modes of Operation

BDM is available in all operating modes but must be enabled before firmware commands are executed. Some systems may have a control bit that allows suspending thefunction during background debug mode.

21.1.2.1 Regular Run Modes

All of these operations refer to the part in run mode and not being secured. The BDM does not provide controls to conserve power during run mode.

- Normal modes

  General operation of the BDM is available and operates the same in all normal modes.
- Special single chip mode

  In special single chip mode, background operation is enabled and active out of reset. This allows programming a system with blank memory.
- Emulation modes (if modes available)

  In emulation mode, background operation is enabled but not active out of reset. This allows debugging and programming a system in this mode more easily.

21.1.2.2 Secure Mode Operation

If the device is in secure mode, the operation of the BDM is reduced to a small subset of its regular run mode operation. Secure operation prevents BDM and CPU accesses to non-volatile memory (Flash and/or EEPROM) other than allowing erasure. For more information please see Section 21.4.1, “Security”.

21.1.2.3 Low-Power Modes

The BDM can be used until all bus masters (e.g., CPU or XGATE or others depending on which masters are available on the SOC) are in stop mode. When CPU is in a low power mode (wait or stop mode) all BDM firmware commands as well as the hardware BACKGROUND command can not be used respectively are ignored. In this case the CPU can not enter BDM active mode, and only hardware read and write commands are available. Also the CPU can not enter a low power mode during BDM active mode.

If all bus masters are in stop mode, the BDM clocks are stopped as well. When BDM clocks are disabled and one of the bus masters exits from stop mode the BDM clocks will restart and BDM will have a soft reset (clearing the instruction register, any command in progress and disable the ACK function). The BDM is now ready to receive a new command.

21.1.3 Block Diagram

A block diagram of the BDM is shown in Figure 21-1.

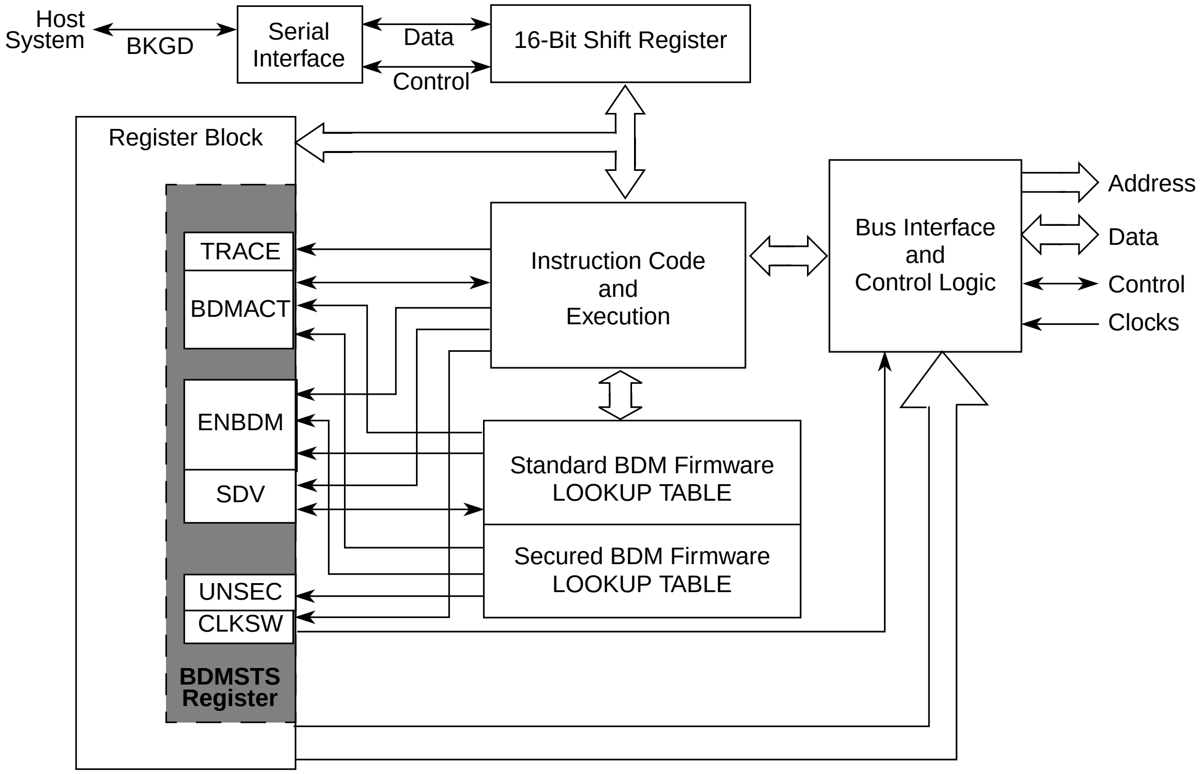


Figure 21-1. BDM Block Diagram

21.2 External Signal Description

A single-wire interface pin called the background debug interface (BKGD) pin is used to communicate with the BDM system. During reset, this pin is a mode select input which selects between normal and special modes of operation. After reset, this pin becomes the dedicated serial interface pin for the background debug mode.

21.3 Memory Map and Register Definition

21.3.1 Module Memory Map

Table 21-2 shows the BDM memory map when BDM is active.

Table 21-2. BDM Memory Map

Global Address	Module	Size (Bytes)
0x7FFF00–0x7FFF0B	BDM registers	12
0x7FFF0C–0x7FFF0E	BDM firmware ROM	3
0x7FFF0F	Family ID (part of BDM firmware ROM)	1
0x7FFF10–0x7FFFFF	BDM firmware ROM	240

21.3.2 Register Descriptions

A summary of the registers associated with the BDM is shown in Figure 21-2. Registers are accessed by host-driven communications to the BDM hardware using READ_BD and WRITE_BD commands.

Global Address	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x7FFF00	Reserved	R	X	X	X	X	X	X	0	0
		W	░	░	░	░	░	░	░	░
0x7FFF01	BDMSTS	R	ENBDM	BDMACT	0	SDV	TRACE	CLKSW	UNSEC	0
		W		░	░	■	■		░	░
0x7FFF02	Reserved	R	X	X	X	X	X	X	X	X
		W	░	░	░	░	░	░	░	░
0x7FFF03	Reserved	R	X	X	X	X	X	X	X	X
		W	░	░	░	░	░	░	░	░
0x7FFF04	Reserved	R	X	X	X	X	X	X	X	X
		W	░	░	░	░	░	░	░	░
0x7FFF05	Reserved	R	X	X	X	X	X	X	X	X
		W	░	░	░	░	░	░	░	░
0x7FFF06	BDMCCRL	R	CCR7	CCR6	CCR5	CCR4	CCR3	CCR2	CCR1	CCR0
		W								

░ = Unimplemented, Reserved

■ = Implemented (do not alter)

X = Indeterminate

0 = Always read zero

Figure 21-2. BDM Register Summary

Global Address	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x7FFF07	BDMCCRH	R	0	0	0	0	0	CCR10	CCR9	CCR8
		W	▨	▨	▨	▨	▨			
0x7FFF08	BDMGPR	R	BGAE	BGP6	BGP5	BGP4	BGP3	BGP2	BGP1	BGP0
		W								
0x7FFF09	Reserved	R	0	0	0	0	0	0	0	0
		W	▨	▨	▨	▨	▨	▨	▨	▨
0x7FFF0A	Reserved	R	0	0	0	0	0	0	0	0
		W	▨	▨	▨	▨	▨	▨	▨	▨
0x7FFF0B	Reserved	R	0	0	0	0	0	0	0	0
		W	▨	▨	▨	▨	▨	▨	▨	▨

▨ = Unimplemented, Reserved

■ = Implemented (do not alter)

X = Indeterminate

0 = Always read zero

Figure 21-2. BDM Register Summary (continued)

21.3.2.1 BDM Status Register (BDMSTS)

Register Global Address 0x7FFF01

	7	6	5	4	3	2	1	0
R	ENBDM	BDMACT	0	SDV	TRACE	CLKSW	UNSEC	0
W		▨	▨		■		▨	▨
Reset								
Special Single-Chip Mode	0[1]	1	0	0	0	0	0[3]	0
Emulation Modes (if modes available)	1	0	0	0	0	1[2]	0	0
All Other Modes	0	0	0	0	0	0	0	0

▨ = Unimplemented, Reserved

■ = Implemented (do not alter)

0 = Always read zero

[1] ENBDM is read as 1 by a debugging environment in special single chip mode when the device is not secured or secured but fully erased (non-volatile memory). This is because the ENBDM bit is set by the standard firmware before a BDM command can be fully transmitted and executed.

[2] CLKSW is read as 1 by a debugging environment in emulation modes when the device is not secured and read as 0 when secured if emulation modes available.

[3] UNSEC is read as 1 by a debugging environment in special single chip mode when the device is secured and fully erased, else it is 0 and can only be read if not secure (see also bit description).

Figure 21-3. BDM Status Register (BDMSTS)

Read: All modes through BDM operation when not secured

Write: All modes through BDM operation when not secured, but subject to the following:

- — ENBDM should only be set via a BDM hardware command if the BDM firmware commands are needed. (This does not apply in special single chip and emulation modes).
- — BDMACT can only be set by BDM hardware upon entry into BDM. It can only be cleared by the standard BDM firmware lookup table upon exit from BDM active mode.
- — CLKSW can only be written via BDM hardware WRITE_BD commands.
- — All other bits, while writable via BDM hardware or standard BDM firmware write commands, should only be altered by the BDM hardware or standard firmware lookup table as part of BDM command execution.

Table 21-3. BDMSTS Field Descriptions

Field	Description
7 ENBDM	**Enable BDM** — This bit controls whether the BDM is enabled or disabled. When enabled, BDM can be made active to allow firmware commands to be executed. When disabled, BDM cannot be made active but BDM hardware commands are still allowed. 0 BDM disabled 1 BDM enabled **Note:** ENBDM is set by the firmware out of reset in special single chip mode. In emulation modes (if modes available) the ENBDM bit is set by BDM hardware out of reset. In special single chip mode with the device secured, this bit will not be set by the firmware until after the non-volatile memory erase verify tests are complete. In emulation modes (if modes available) with the device secured, the BDM operations are blocked.
6 BDMACT	**BDM Active Status** — This bit becomes set upon entering BDM. The standard BDM firmware lookup table is then enabled and put into the memory map. BDMACT is cleared by a carefully timed store instruction in the standard BDM firmware as part of the exit sequence to return to user code and remove the BDM memory from the map. 0 BDM not active 1 BDM active
4 SDV	**Shift Data Valid** — This bit is set and cleared by the BDM hardware. It is set after data has been transmitted as part of a firmware or hardware read command or after data has been received as part of a firmware or hardware write command. It is cleared when the next BDM command has been received or BDM is exited. SDV is used by the standard BDM firmware to control program flow execution. 0 Data phase of command not complete 1 Data phase of command is complete
3 TRACE	**TRACE1 BDM Firmware Command is Being Executed** — This bit gets set when a BDM TRACE1 firmware command is first recognized. It will stay set until BDM firmware is exited by one of the following BDM commands: GO or GO_UNTIL. 0 TRACE1 command is not being executed 1 TRACE1 command is being executed

Table 21-3. BDMSTS Field Descriptions (continued)

Field	Description
2 CLKSW	**Clock Switch** — The CLKSW bit controls which clock the BDM operates with. It is only writable from a hardware BDM command. A minimum delay of 150 cycles at the clock speed that is active during the data portion of the command send to change the clock source should occur before the next command can be send. The delay should be obtained no matter which bit is modified to effectively change the clock source (either PLLSEL bit or CLKSW bit). This guarantees that the start of the next BDM command uses the new clock for timing subsequent BDM communications. Table 21-4 shows the resulting BDM clock source based on the CLKSW and the PLLSEL (PLL select in the CRG module, the bit is part of the CLKSEL register) bits. **Note:** The BDM alternate clock source can only be selected when CLKSW = 0 and PLLSEL = 1. The BDM serial interface is now fully synchronized to the alternate clock source, when enabled. This eliminates frequency restriction on the alternate clock which was required on previous versions. Refer to the device specification to determine which clock connects to the alternate clock source input. **Note:** If the acknowledge function is turned on, changing the CLKSW bit will cause the ACK to be at the new rate for the write command which changes it. **Note:** In emulation modes (if modes available), the CLKSW bit will be set out of RESET.
1 UNSEC	**Unsecure** — If the device is secured this bit is only writable in special single chip mode from the BDM secure firmware. It is in a zero state as secure mode is entered so that the secure BDM firmware lookup table is enabled and put into the memory map overlapping the standard BDM firmware lookup table. The secure BDM firmware lookup table verifies that the non-volatile memories (e.g. on-chip EEPROM and/or Flash EEPROM) are erased. This being the case, the UNSEC bit is set and the BDM program jumps to the start of the standard BDM firmware lookup table and the secure BDM firmware lookup table is turned off. If the erase test fails, the UNSEC bit will not be asserted. 0 System is in a secured mode. 1 System is in a unsecured mode. **Note:** When UNSEC is set, security is off and the user can change the state of the secure bits in the on-chip Flash EEPROM. Note that if the user does not change the state of the bits to “unsecured” mode, the system will be secured again when it is next taken out of reset.After reset this bit has no meaning or effect when the security byte in the Flash EEPROM is configured for unsecure mode.

Table 21-4. BDM Clock Sources

PLLSEL	CLKSW	BDMCLK
0	0	Bus clock dependent on oscillator
0	1	Bus clock dependent on oscillator
1	0	Alternate clock (refer to the device specification to determine the alternate clock source)
1	1	Bus clock dependent on the PLL

21.3.2.2 BDM CCR LOW Holding Register (BDMCCRL)

Register Global Address 0x7FFF06

	7	6	5	4	3	2	1	0
R W	CCR7	CCR6	CCR5	CCR4	CCR3	CCR2	CCR1	CCR0
Reset								
Special Single-Chip Mode	1	1	0	0	1	0	0	0
All Other Modes	0	0	0	0	0	0	0	0

Figure 21-4. BDM CCR LOW Holding Register (BDMCCRL)

Read: All modes through BDM operation when not secured

Write: All modes through BDM operation when not secured

NOTE

> When BDM is made active, the CPU stores the content of its CCR_L register in the BDMCCRL register. However, out of special single-chip reset, the BDMCCRL is set to 0xD8 and not 0xD0 which is the reset value of the CCR_L register in this CPU mode. Out of reset in all other modes the BDMCCRL register is read zero.

When entering background debug mode, the BDM CCR LOW holding register is used to save the low byte of the condition code register of the user's program. It is also used for temporary storage in the standard BDM firmware mode. The BDM CCR LOW holding register can be written to modify the CCR value.

21.3.2.3 BDM CCR HIGH Holding Register (BDMCCRH)

Register Global Address 0x7FFF07

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	CCR10	CCR9	CCR8
W	▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 21-5. BDM CCR HIGH Holding Register (BDMCCRH)

Read: All modes through BDM operation when not secured

Write: All modes through BDM operation when not secured

When entering background debug mode, the BDM CCR HIGH holding register is used to save the high byte of the condition code register of the user's program. The BDM CCR HIGH holding register can be written to modify the CCR value.

21.3.2.4 BDM Global Page Index Register (BDMGPR)

Register Global Address 0x7FFF08

	7	6	5	4	3	2	1	0
R W	BGAE	BGP6	BGP5	BGP4	BGP3	BGP2	BGP1	BGP0
Reset	0	0	0	0	0	0	0	0

Figure 21-6. BDM Global Page Register (BDMGPR)

Read: All modes through BDM operation when not secured

Write: All modes through BDM operation when not secured

Table 21-5. BDMGPR Field Descriptions

Field	Description
7 BGAE	**BDM Global Page Access Enable Bit** — BGAE enables global page access for BDM hardware and firmware read/write instructions The BDM hardware commands used to access the BDM registers (READ_BD_ and WRITE_BD_) can not be used for global accesses even if the BGAE bit is set. 0 BDM Global Access disabled 1 BDM Global Access enabled
6–0 BGP[6:0]	**BDM Global Page Index Bits 6–0** — These bits define the extended address bits from 22 to 16. For more detailed information regarding the global page window scheme, please refer to the S12X_MMC Block Guide.

21.3.3 Family ID Assignment

The family ID is a 8-bit value located in the firmware ROM (at global address: 0x7FFF0F). The read-only value is a unique family ID which is 0xC1 for S12X devices.

21.4 Functional Description

The BDM receives and executes commands from a host via a single wire serial interface. There are two types of BDM commands: hardware and firmware commands.

Hardware commands are used to read and write target system memory locations and to enter active background debug mode, see Section 21.4.3, “BDM Hardware Commands”. Target system memory includes all memory that is accessible by the CPU.

Firmware commands are used to read and write CPU resources and to exit from active background debug mode, see Section 21.4.4, “Standard BDM Firmware Commands”. The CPU resources referred to are the accumulator (D), X index register (X), Y index register (Y), stack pointer (SP), and program counter (PC).

Hardware commands can be executed at any time and in any mode excluding a few exceptions as highlighted (see Section 21.4.3, “BDM Hardware Commands”) and in secure mode (see Section 21.4.1, “Security”). Firmware commands can only be executed when the system is not secure and is in active background debug mode (BDM).

21.4.1 Security

If the user resets into special single chip mode with the system secured, a secured mode BDM firmware lookup table is brought into the map overlapping a portion of the standard BDM firmware lookup table. The secure BDM firmware verifies that the on-chip non-volatile memory (e.g. EEPROM and Flash EEPROM) is erased. This being the case, the UNSEC and ENBDM bit will get set. The BDM program jumps to the start of the standard BDM firmware and the secured mode BDM firmware is turned off and all BDM commands are allowed. If the non-volatile memory does not verify as erased, the BDM firmware sets the ENBDM bit, without asserting UNSEC, and the firmware enters a loop. This causes the BDM hardware commands to become enabled, but does not enable the firmware commands. This allows the BDM hardware to be used to erase the non-volatile memory.

BDM operation is not possible in any other mode than special single chip mode when the device is secured. The device can be unsecured via BDM serial interface in special single chip mode only. For more information regarding security, please see the S12X_9SEC Block Guide.

21.4.2 Enabling and Activating BDM

The system must be in active BDM to execute standard BDM firmware commands. BDM can be activated only after being enabled. BDM is enabled by setting the ENBDM bit in the BDM status (BDMSTS) register. The ENBDM bit is set by writing to the BDM status (BDMSTS) register, via the single-wire interface, using a hardware command such as WRITE_BD_BYTE.

After being enabled, BDM is activated by one of the following[1]:

- Hardware BACKGROUND command
- CPU BGND instruction
- External instruction tagging mechanism[2]
- Breakpoint force or tag mechanism[2]

When BDM is activated, the CPU finishes executing the current instruction and then begins executing the firmware in the standard BDM firmware lookup table. When BDM is activated by a breakpoint, the type of breakpoint used determines if BDM becomes active before or after execution of the next instruction.

NOTE

> If an attempt is made to activate BDM before being enabled, the CPU resumes normal instruction execution after a brief delay. If BDM is not enabled, any hardware BACKGROUND commands issued are ignored by the BDM and the CPU is not delayed.

In active BDM, the BDM registers and standard BDM firmware lookup table are mapped to addresses 0x7FFF00 to 0x7FFFFF. BDM registers are mapped to addresses 0x7FFF00 to 0x7FFF0B. The BDM uses these registers which are readable anytime by the BDM. However, these registers are not readable by user programs.

---

[1] BDM is enabled and active immediately out of special single-chip reset.

[2] This method is provided by the S12X_DBG module.

21.4.3 BDM Hardware Commands

Hardware commands are used to read and write target system memory locations and to enter active background debug mode. Target system memory includes all memory that is accessible by the CPU on the SOC which can be on-chip RAM, non-volatile memory (e.g. EEPROM, Flash EEPROM), I/O and control registers, and all external memory.

Hardware commands are executed with minimal or no CPU intervention and do not require the system to be in active BDM for execution, although, they can still be executed in this mode. When executing a hardware command, the BDM sub-block waits for a free bus cycle so that the background access does not disturb the running application program. If a free cycle is not found within 128 clock cycles, the CPU is momentarily frozen so that the BDM can steal a cycle. When the BDM finds a free cycle, the operation does not intrude on normal CPU operation provided that it can be completed in a single cycle. However, if an operation requires multiple cycles the CPU is frozen until the operation is complete, even though the BDM found a free cycle.

The BDM hardware commands are listed in Table 21-6.

The READ_BD and WRITE_BD commands allow access to the BDM register locations. These locations are not normally in the system memory map but share addresses with the application in memory. To distinguish between physical memory locations that share the same address, BDM memory resources are enabled just for the READ_BD and WRITE_BD access cycle. This allows the BDM to access BDM locations unobtrusively, even if the addresses conflict with the application memory map.

Table 21-6. Hardware Commands

Command	Opcode (hex)	Data	Description
BACKGROUND	90	None	Enter background mode if firmware is enabled. If enabled, an ACK will be issued when the part enters active background mode.
ACK_ENABLE	D5	None	Enable Handshake. Issues an ACK pulse after the command is executed.
ACK_DISABLE	D6	None	Disable Handshake. This command does not issue an ACK pulse.
READ_BD_BYTE	E4	16-bit address 16-bit data out	Read from memory with standard BDM firmware lookup table in map. Odd address data on low byte; even address data on high byte.
READ_BD_WORD	EC	16-bit address 16-bit data out	Read from memory with standard BDM firmware lookup table in map. Must be aligned access.
READ_BYTE	E0	16-bit address 16-bit data out	Read from memory with standard BDM firmware lookup table out of map. Odd address data on low byte; even address data on high byte.
READ_WORD	E8	16-bit address 16-bit data out	Read from memory with standard BDM firmware lookup table out of map. Must be aligned access.
WRITE_BD_BYTE	C4	16-bit address 16-bit data in	Write to memory with standard BDM firmware lookup table in map. Odd address data on low byte; even address data on high byte.
WRITE_BD_WORD	CC	16-bit address 16-bit data in	Write to memory with standard BDM firmware lookup table in map. Must be aligned access.
WRITE_BYTE	C0	16-bit address 16-bit data in	Write to memory with standard BDM firmware lookup table out of map. Odd address data on low byte; even address data on high byte.

Table 21-6. Hardware Commands (continued)

Command	Opcode (hex)	Data	Description
WRITE_WORD	C8	16-bit address 16-bit data in	Write to memory with standard BDM firmware lookup table out of map. Must be aligned access.

NOTE:
If enabled, ACK will occur when data is ready for transmission for all BDM READ commands and will occur after the write is complete for all BDM WRITE commands.

21.4.4 Standard BDM Firmware Commands

Firmware commands are used to access and manipulate CPU resources. The system must be in active BDM to execute standard BDM firmware commands, see Section 21.4.2, “Enabling and Activating BDM”. Normal instruction execution is suspended while the CPU executes the firmware located in the standard BDM firmware lookup table. The hardware command BACKGROUND is the usual way to activate BDM.

As the system enters active BDM, the standard BDM firmware lookup table and BDM registers become visible in the on-chip memory map at 0x7FFF00–0x7FFFFF, and the CPU begins executing the standard BDM firmware. The standard BDM firmware watches for serial commands and executes them as they are received.

The firmware commands are shown in Table 21-7.

Table 21-7. Firmware Commands

Command[1]	Opcode (hex)	Data	Description
READ_NEXT[2]	62	16-bit data out	Increment X index register by 2 (X = X + 2), then read word X points to.
READ_PC	63	16-bit data out	Read program counter.
READ_D	64	16-bit data out	Read D accumulator.
READ_X	65	16-bit data out	Read X index register.
READ_Y	66	16-bit data out	Read Y index register.
READ_SP	67	16-bit data out	Read stack pointer.
WRITE_NEXT<f-helvetica><st-superscript>	42	16-bit data in	Increment X index register by 2 (X = X + 2), then write word to location pointed to by X.
WRITE_PC	43	16-bit data in	Write program counter.
WRITE_D	44	16-bit data in	Write D accumulator.
WRITE_X	45	16-bit data in	Write X index register.
WRITE_Y	46	16-bit data in	Write Y index register.
WRITE_SP	47	16-bit data in	Write stack pointer.
GO	08	none	Go to user program. If enabled, ACK will occur when leaving active background mode.
GO_UNTIL[3]	0C	none	Go to user program. If enabled, ACK will occur upon returning to active background mode.
TRACE1	10	none	Execute one user instruction then return to active BDM. If enabled, ACK will occur upon returning to active background mode.
TAGGO -> GO	18	none	(Previous enable tagging and go to user program.) This command will be deprecated and should not be used anymore. Opcode will be executed as a GO command.

[1] If enabled, ACK will occur when data is ready for transmission for all BDM READ commands and will occur after the write is complete for all BDM WRITE commands.

[2] When the firmware command READ_NEXT or WRITE_NEXT is used to access the BDM address space the BDM resources are accessed rather than user code. Writing BDM firmware is not possible.

[3] System stop disables the ACK function and ignored commands will not have an ACK-pulse (e.g., CPU in stop or wait mode). The GO_UNTIL command will not get an Acknowledge if CPU executes the wait or stop instruction before the “UNTIL” condition (BDM active again) is reached (see Section 21.4.7, “Serial Interface Hardware Handshake Protocol” last Note).

21.4.5 BDM Command Structure

Hardware and firmware BDM commands start with an 8-bit opcode followed by a 16-bit address and/or a 16-bit data word depending on the command. All the read commands return 16 bits of data despite the byte or word implication in the command name.

> 8-bit reads return 16-bits of data, of which, only one byte will contain valid data. If reading an even address, the valid data will appear in the MSB. If reading an odd address, the valid data will appear in the LSB.

> 16-bit misaligned reads and writes are generally not allowed. If attempted by BDM hardware command, the BDM will ignore the least significant bit of the address and will assume an even address from the remaining bits.
>
> For devices with external bus:
>
> The following cycle count information is only valid when the external wait function is not used (see wait bit of EBI sub-block). During an external wait the BDM can not steal a cycle. Hence be careful with the external wait function if the BDM serial interface is much faster than the bus, because of the BDM soft-reset after time-out (see Section 21.4.11, "Serial Communication Time Out").

For hardware data read commands, the external host must wait at least 150 bus clock cycles after sending the address before attempting to obtain the read data. This is to be certain that valid data is available in the BDM shift register, ready to be shifted out. For hardware write commands, the external host must wait 150 bus clock cycles after sending the data to be written before attempting to send a new command. This is to avoid disturbing the BDM shift register before the write has been completed. The 150 bus clock cycle delay in both cases includes the maximum 128 cycle delay that can be incurred as the BDM waits for a free cycle before stealing a cycle.

For firmware read commands, the external host should wait at least 48 bus clock cycles after sending the command opcode and before attempting to obtain the read data. This includes the potential of extra cycles when the access is external and stretched (+1 to maximum +7 cycles) or to registers of the PRU (port replacement unit) in emulation modes (if modes available). The 48 cycle wait allows enough time for the requested data to be made available in the BDM shift register, ready to be shifted out.

NOTE

> This timing has increased from previous BDM modules due to the new capability in which the BDM serial interface can potentially run faster than the bus. On previous BDM modules this extra time could be hidden within the serial time.

For firmware write commands, the external host must wait 36 bus clock cycles after sending the data to be written before attempting to send a new command. This is to avoid disturbing the BDM shift register before the write has been completed.

The external host should wait at least for 76 bus clock cycles after a TRACE1 or GO command before starting any new serial command. This is to allow the CPU to exit gracefully from the standard BDM firmware lookup table and resume execution of the user code. Disturbing the BDM shift register prematurely may adversely affect the exit from the standard BDM firmware lookup table.

NOTE

> If the bus rate of the target processor is unknown or could be changing or the external wait function is used, it is recommended that the ACK (acknowledge function) is used to indicate when an operation is complete. When using ACK, the delay times are automated.

Figure 21-7 represents the BDM command structure. The command blocks illustrate a series of eight bit times starting with a falling edge. The bar across the top of the blocks indicates that the BKGD line idles in the high state. The time for an 8-bit command is 8 × 16 target clock cycles.[1]

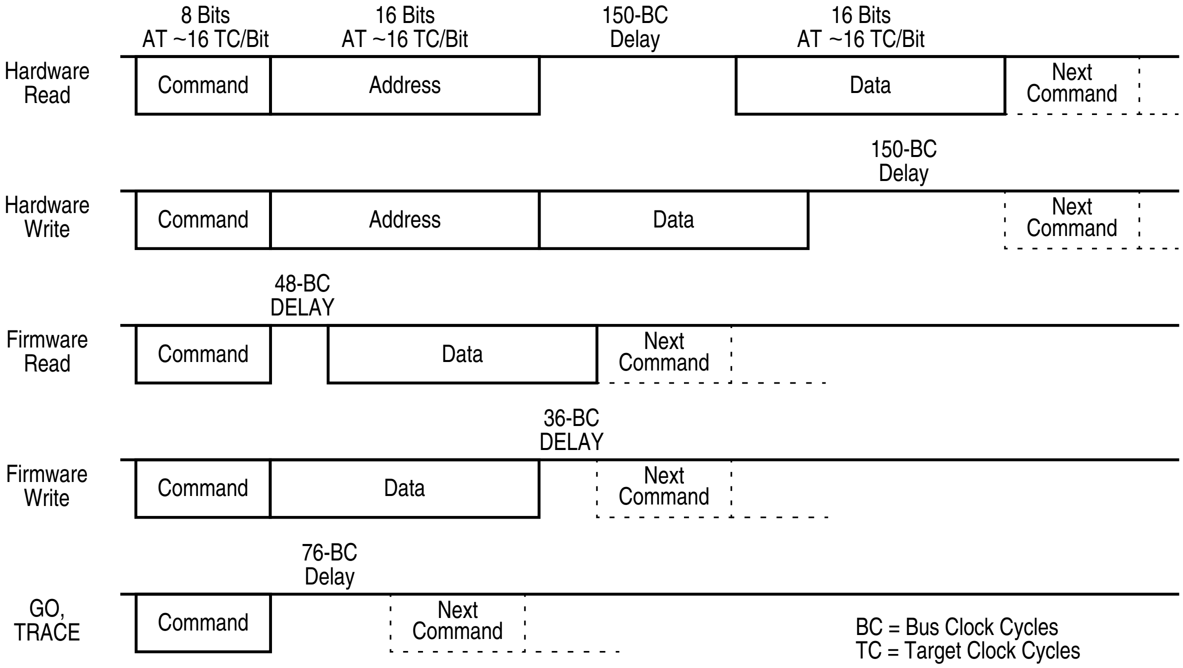


Figure 21-7. BDM Command Structure

21.4.6 BDM Serial Interface

The BDM communicates with external devices serially via the BKGD pin. During reset, this pin is a mode select input which selects between normal and special modes of operation. After reset, this pin becomes the dedicated serial interface pin for the BDM.

The BDM serial interface is timed using the clock selected by the CLKSW bit in the status register see Section 21.3.2.1, “BDM Status Register (BDMSTS)”. This clock will be referred to as the target clock in the following explanation.

The BDM serial interface uses a clocking scheme in which the external host generates a falling edge on the BKGD pin to indicate the start of each bit time. This falling edge is sent for every bit whether data is transmitted or received. Data is transferred most significant bit (MSB) first at 16 target clock cycles per bit. The interface times out if 512 clock cycles occur between falling edges from the host.

The BKGD pin is a pseudo open-drain pin and has an weak on-chip active pull-up that is enabled at all times. It is assumed that there is an external pull-up and that drivers connected to BKGD do not typically drive the high level. Since R-C rise time could be unacceptably long, the target system and host provide brief driven-high (speedup) pulses to drive BKGD to a logic 1. The source of this speedup pulse is the host for transmit cases and the target for receive cases.

---

[1] Target clock cycles are cycles measured using the target MCU’s serial clock rate. See Section 21.4.6, “BDM Serial Interface” and Section 21.3.2.1, “BDM Status Register (BDMSTS)” for information on how serial clock rate is selected.

The timing for host-to-target is shown in Figure 21-8 and that of target-to-host in Figure 21-9 and Figure 21-10. All four cases begin when the host drives the BKGD pin low to generate a falling edge. Since the host and target are operating from separate clocks, it can take the target system up to one full clock cycle to recognize this edge. The target measures delays from this perceived start of the bit time while the host measures delays from the point it actually drove BKGD low to start the bit up to one target clock cycle earlier. Synchronization between the host and target is established in this manner at the start of every bit time.

Figure 21-8 shows an external host transmitting a logic 1 and transmitting a logic 0 to the BKGD pin of a target system. The host is asynchronous to the target, so there is up to a one clock-cycle delay from the host-generated falling edge to where the target recognizes this edge as the beginning of the bit time. Ten target clock cycles later, the target senses the bit level on the BKGD pin. Internal glitch detect logic requires the pin be driven high no later that eight target clock cycles after the falling edge for a logic 1 transmission.

Since the host drives the high speedup pulses in these two cases, the rising edges look like digitally driven signals.

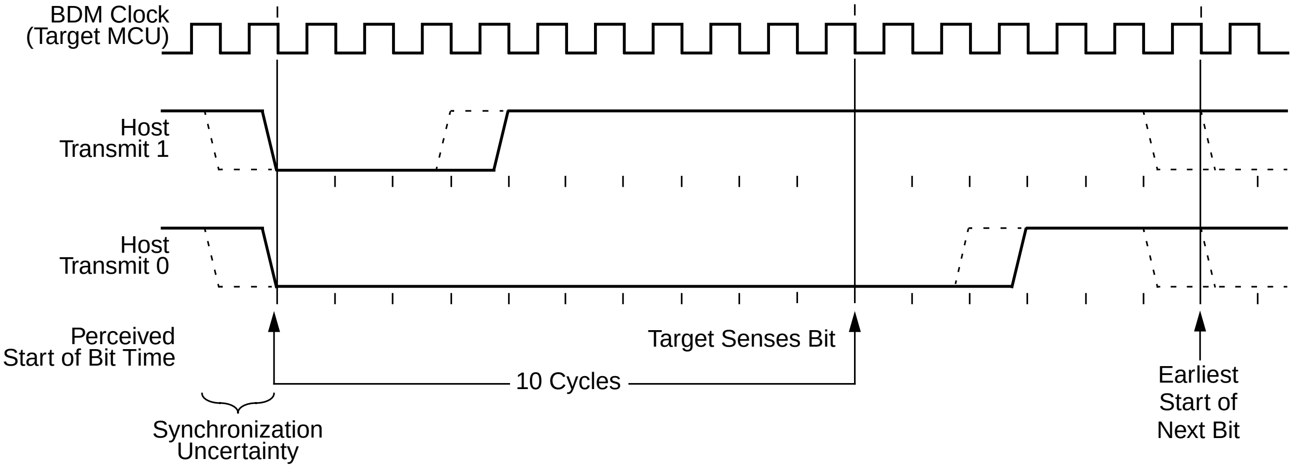


Figure 21-8. BDM Host-to-Target Serial Bit Timing

The receive cases are more complicated. Figure 21-9 shows the host receiving a logic 1 from the target system. Since the host is asynchronous to the target, there is up to one clock-cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target. The host holds the BKGD pin low long enough for the target to recognize it (at least two target clock cycles). The host must release the low drive before the target drives a brief high speedup pulse seven target clock cycles after the perceived start of the bit time. The host should sample the bit level about 10 target clock cycles after it started the bit time.

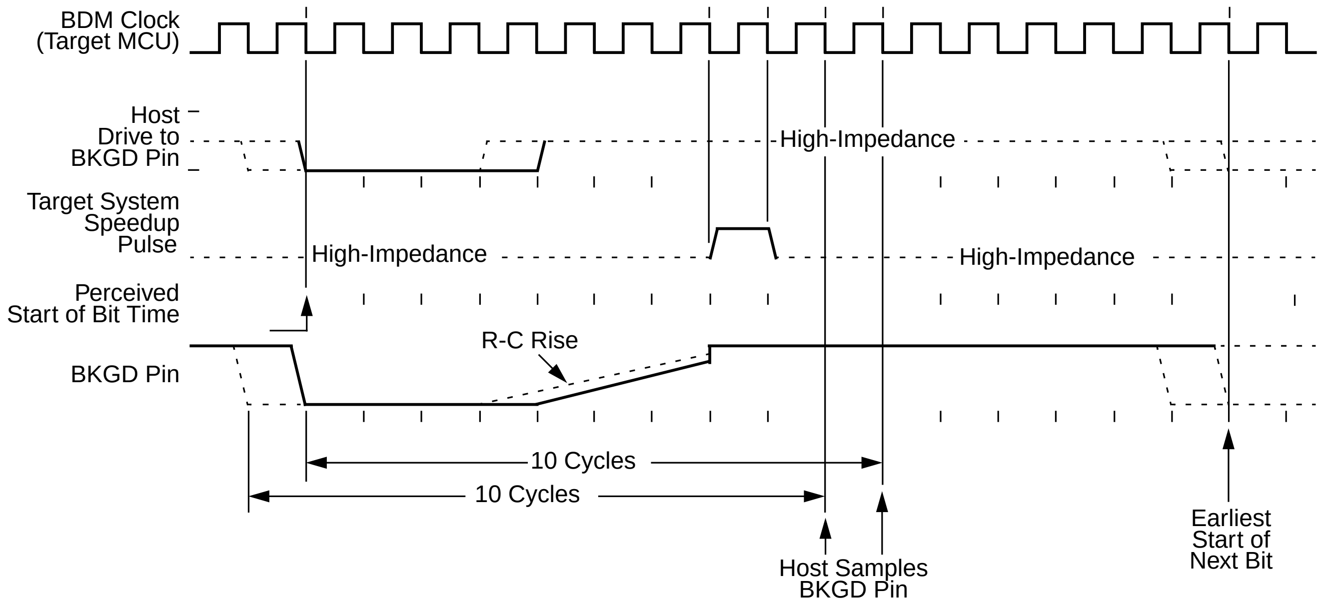


Figure 21-9. BDM Target-to-Host Serial Bit Timing (Logic 1)

Figure 21-10 shows the host receiving a logic 0 from the target. Since the host is asynchronous to the target, there is up to a one clock-cycle delay from the host-generated falling edge on BKGD to the start of the bit time as perceived by the target. The host initiates the bit time but the target finishes it. Since the target wants the host to receive a logic 0, it drives the BKGD pin low for 13 target clock cycles then briefly drives it high to speed up the rising edge. The host samples the bit level about 10 target clock cycles after starting the bit time.

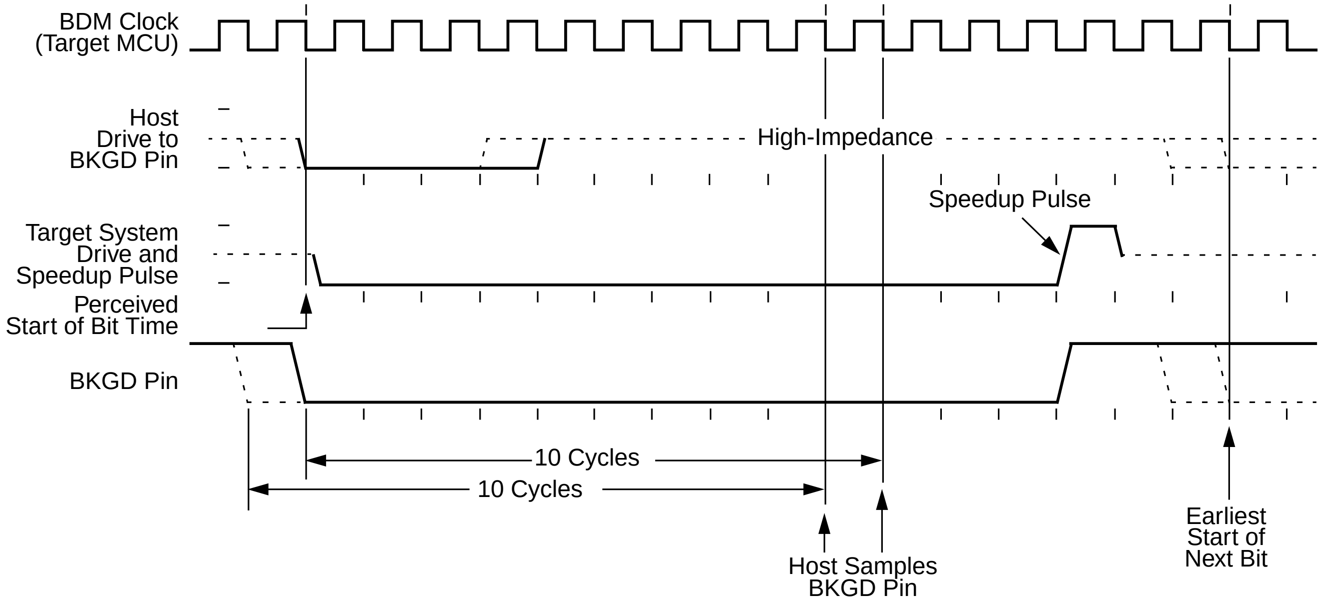


Figure 21-10. BDM Target-to-Host Serial Bit Timing (Logic 0)

21.4.7 Serial Interface Hardware Handshake Protocol

BDM commands that require CPU execution are ultimately treated at the MCU bus rate. Since the BDM clock source can be asynchronously related to the bus frequency, when CLKSW = 0, it is very helpful to provide a handshake protocol in which the host could determine when an issued command is executed by the CPU. The alternative is to always wait the amount of time equal to the appropriate number of cycles at the slowest possible rate the clock could be running. This sub-section will describe the hardware handshake protocol.

The hardware handshake protocol signals to the host controller when an issued command was successfully executed by the target. This protocol is implemented by a 16 serial clock cycle low pulse followed by a brief speedup pulse in the BKGD pin. This pulse is generated by the target MCU when a command, issued by the host, has been successfully executed (see Figure 21-11). This pulse is referred to as the ACK pulse. After the ACK pulse has finished: the host can start the bit retrieval if the last issued command was a read command, or start a new command if the last command was a write command or a control command (BACKGROUND, GO, GO_UNTIL or TRACE1). The ACK pulse is not issued earlier than 32 serial clock cycles after the BDM command was issued. The end of the BDM command is assumed to be the 16th tick of the last bit. This minimum delay assures enough time for the host to perceive the ACK pulse. Note also that, there is no upper limit for the delay between the command and the related ACK pulse, since the command execution depends upon the CPU bus frequency, which in some cases could be very slow compared to the serial communication rate. This protocol allows a great flexibility for the POD designers, since it does not rely on any accurate time measurement or short response time to any event in the serial communication.

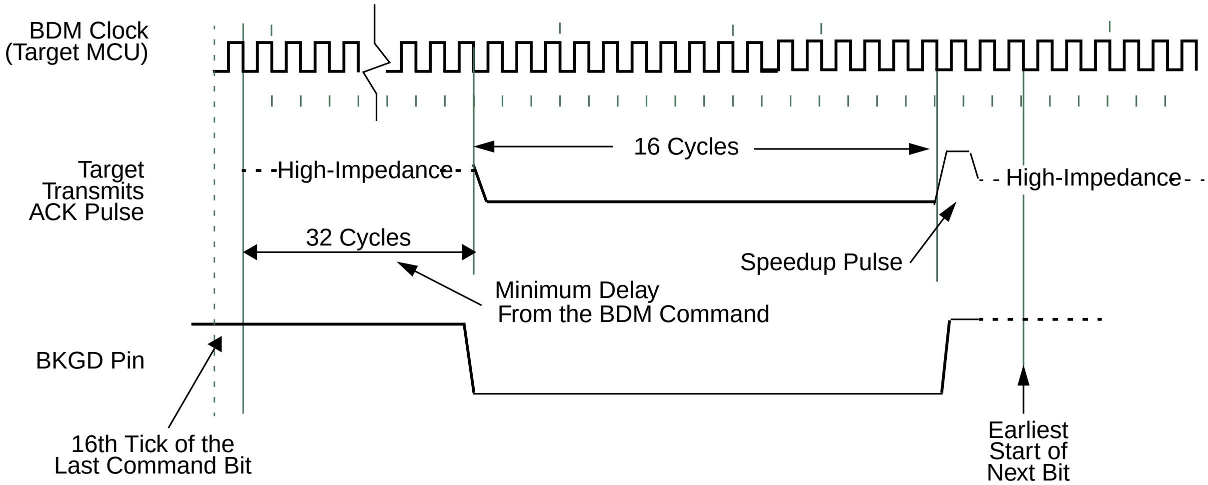


Figure 21-11. Target Acknowledge Pulse (ACK)

NOTE

If the ACK pulse was issued by the target, the host assumes the previous command was executed. If the CPU enters wait or stop prior to executing a hardware command, the ACK pulse will not be issued meaning that the BDM command was not executed. After entering wait or stop mode, the BDM command is no longer pending.

Figure 21-12 shows the ACK handshake protocol in a command level timing diagram. The READ_BYTE instruction is used as an example. First, the 8-bit instruction opcode is sent by the host, followed by the address of the memory location to be read. The target BDM decodes the instruction. A bus cycle is grabbed (free or stolen) by the BDM and it executes the READ_BYTE operation. Having retrieved the data, the BDM issues an ACK pulse to the host controller, indicating that the addressed byte is ready to be retrieved. After detecting the ACK pulse, the host initiates the byte retrieval process. Note that data is sent in the form of a word and the host needs to determine which is the appropriate byte based on whether the address was odd or even.

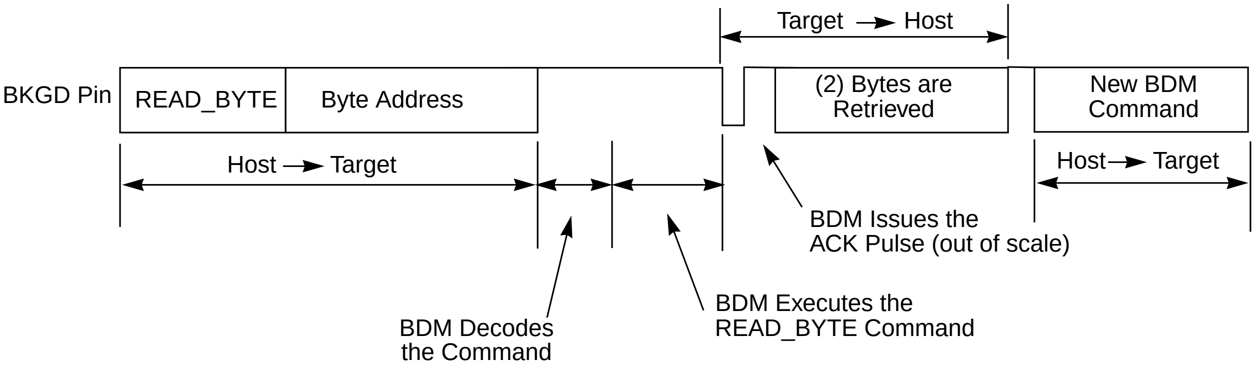


Figure 21-12. Handshake Protocol at Command Level

Differently from the normal bit transfer (where the host initiates the transmission), the serial interface ACK handshake pulse is initiated by the target MCU by issuing a negative edge in the BKGD pin. The hardware handshake protocol in Figure 21-11 specifies the timing when the BKGD pin is being driven, so the host should follow this timing constraint in order to avoid the risk of an electrical conflict in the BKGD pin.

NOTE

> The only place the BKGD pin can have an electrical conflict is when one side is driving low and the other side is issuing a speedup pulse (high). Other "highs" are pulled rather than driven. However, at low rates the time of the speedup pulse can become lengthy and so the potential conflict time becomes longer as well.

The ACK handshake protocol does not support nested ACK pulses. If a BDM command is not acknowledge by an ACK pulse, the host needs to abort the pending command first in order to be able to issue a new BDM command. When the CPU enters wait or stop while the host issues a hardware command (e.g., WRITE_BYTE), the target discards the incoming command due to the wait or stop being detected. Therefore, the command is not acknowledged by the target, which means that the ACK pulse will not be issued in this case. After a certain time the host (not aware of stop or wait) should decide to abort any possible pending ACK pulse in order to be sure a new command can be issued. Therefore, the protocol provides a mechanism in which a command, and its corresponding ACK, can be aborted.

NOTE

> The ACK pulse does not provide a time out. This means for the GO_UNTIL command that it can not be distinguished if a stop or wait has been executed (command discarded and ACK not issued) or if the "UNTIL" condition (BDM active) is just not reached yet. Hence in any case where the ACK pulse of a command is not issued the possible pending command should be aborted before issuing a new command. See the handshake abort procedure described in Section 21.4.8, "Hardware Handshake Abort Procedure".

21.4.8 Hardware Handshake Abort Procedure

The abort procedure is based on the SYNC command. In order to abort a command, which had not issued the corresponding ACK pulse, the host controller should generate a low pulse in the BKGD pin by driving it low for at least 128 serial clock cycles and then driving it high for one serial clock cycle, providing a speedup pulse. By detecting this long low pulse in the BKGD pin, the target executes the SYNC protocol, see Section 21.4.9, "SYNC — Request Timed Reference Pulse", and assumes that the pending command and therefore the related ACK pulse, are being aborted. Therefore, after the SYNC protocol has been completed the host is free to issue new BDM commands. For Firmware READ or WRITE commands it can not be guaranteed that the pending command is aborted when issuing a SYNC before the corresponding ACK pulse. There is a short latency time from the time the READ or WRITE access begins until it is finished and the corresponding ACK pulse is issued. The latency time depends on the firmware READ or WRITE command that is issued and if the serial interface is running on a different clock rate than the bus. When the SYNC command starts during this latency time the READ or WRITE command will not be aborted, but the corresponding ACK pulse will be aborted. A pending GO, TRACE1 or

GO_UNTIL command can not be aborted. Only the corresponding ACK pulse can be aborted by the SYNC command.

Although it is not recommended, the host could abort a pending BDM command by issuing a low pulse in the BKGD pin shorter than 128 serial clock cycles, which will not be interpreted as the SYNC command. The ACK is actually aborted when a negative edge is perceived by the target in the BKGD pin. The short abort pulse should have at least 4 clock cycles keeping the BKGD pin low, in order to allow the negative edge to be detected by the target. In this case, the target will not execute the SYNC protocol but the pending command will be aborted along with the ACK pulse. The potential problem with this abort procedure is when there is a conflict between the ACK pulse and the short abort pulse. In this case, the target may not perceive the abort pulse. The worst case is when the pending command is a read command (i.e., READ_BYTE). If the abort pulse is not perceived by the target the host will attempt to send a new command after the abort pulse was issued, while the target expects the host to retrieve the accessed memory byte. In this case, host and target will run out of synchronism. However, if the command to be aborted is not a read command the short abort pulse could be used. After a command is aborted the target assumes the next negative edge, after the abort pulse, is the first bit of a new BDM command.

NOTE

The details about the short abort pulse are being provided only as a reference for the reader to better understand the BDM internal behavior. It is not recommended that this procedure be used in a real application.

Since the host knows the target serial clock frequency, the SYNC command (used to abort a command) does not need to consider the lower possible target frequency. In this case, the host could issue a SYNC very close to the 128 serial clock cycles length. Providing a small overhead on the pulse length in order to assure the SYNC pulse will not be misinterpreted by the target. See Section 21.4.9, “SYNC — Request Timed Reference Pulse”.

Figure 21-13 shows a SYNC command being issued after a READ_BYTE, which aborts the READ_BYTE command. Note that, after the command is aborted a new command could be issued by the host computer.

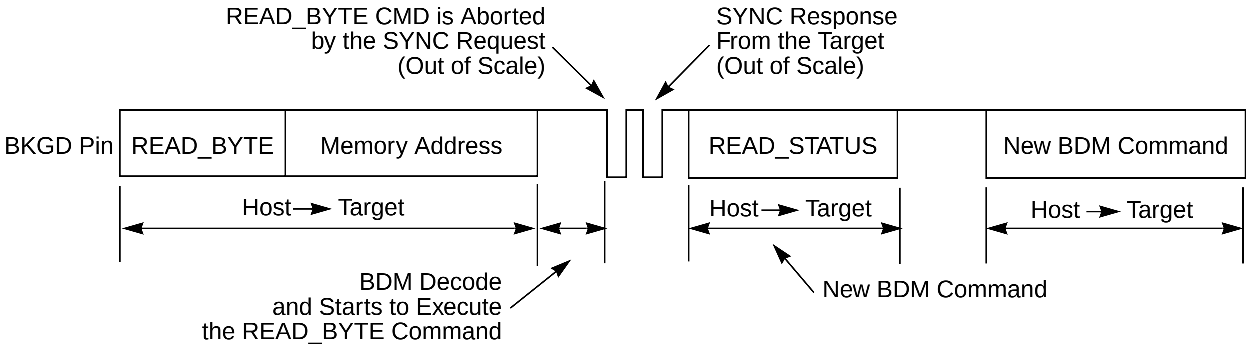


Figure 21-13. ACK Abort Procedure at the Command Level

NOTE

Figure 21-13 does not represent the signals in a true timing scale

Figure 21-14 shows a conflict between the ACK pulse and the SYNC request pulse. This conflict could occur if a POD device is connected to the target BKGD pin and the target is already in debug active mode. Consider that the target CPU is executing a pending BDM command at the exact moment the POD is being connected to the BKGD pin. In this case, an ACK pulse is issued along with the SYNC command. In this case, there is an electrical conflict between the ACK speedup pulse and the SYNC pulse. Since this is not a probable situation, the protocol does not prevent this conflict from happening.

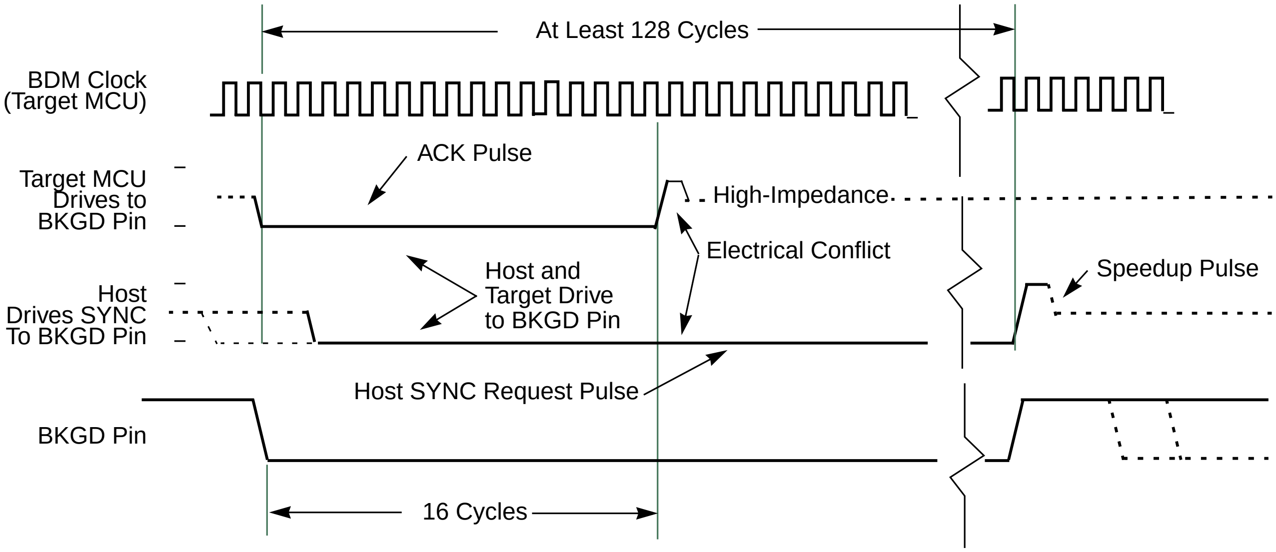


Figure 21-14. ACK Pulse and SYNC Request Conflict

NOTE

> This information is being provided so that the MCU integrator will be aware that such a conflict could eventually occur.

The hardware handshake protocol is enabled by the ACK_ENABLE and disabled by the ACK_DISABLE BDM commands. This provides backwards compatibility with the existing POD devices which are not able to execute the hardware handshake protocol. It also allows for new POD devices, that support the hardware handshake protocol, to freely communicate with the target device. If desired, without the need for waiting for the ACK pulse.

The commands are described as follows:

- ACK_ENABLE — enables the hardware handshake protocol. The target will issue the ACK pulse when a CPU command is executed by the CPU. The ACK_ENABLE command itself also has the ACK pulse as a response.
- ACK_DISABLE — disables the ACK pulse protocol. In this case, the host needs to use the worst case delay time at the appropriate places in the protocol.

The default state of the BDM after reset is hardware handshake protocol disabled.

All the read commands will ACK (if enabled) when the data bus cycle has completed and the data is then ready for reading out by the BKGD serial pin. All the write commands will ACK (if enabled) after the data has been received by the BDM through the BKGD serial pin and when the data bus cycle is complete. See Section 21.4.3, “BDM Hardware Commands” and Section 21.4.4, “Standard BDM Firmware Commands” for more information on the BDM commands.

The ACK_ENABLE sends an ACK pulse when the command has been completed. This feature could be used by the host to evaluate if the target supports the hardware handshake protocol. If an ACK pulse is issued in response to this command, the host knows that the target supports the hardware handshake protocol. If the target does not support the hardware handshake protocol the ACK pulse is not issued. In this case, the ACK_ENABLE command is ignored by the target since it is not recognized as a valid command.

The BACKGROUND command will issue an ACK pulse when the CPU changes from normal to background mode. The ACK pulse related to this command could be aborted using the SYNC command.

The GO command will issue an ACK pulse when the CPU exits from background mode. The ACK pulse related to this command could be aborted using the SYNC command.

The GO_UNTIL command is equivalent to a GO command with exception that the ACK pulse, in this case, is issued when the CPU enters into background mode. This command is an alternative to the GO command and should be used when the host wants to trace if a breakpoint match occurs and causes the CPU to enter active background mode. Note that the ACK is issued whenever the CPU enters BDM, which could be caused by a breakpoint match or by a BGND instruction being executed. The ACK pulse related to this command could be aborted using the SYNC command.

The TRACE1 command has the related ACK pulse issued when the CPU enters background active mode after one instruction of the application program is executed. The ACK pulse related to this command could be aborted using the SYNC command.

21.4.9 SYNC — Request Timed Reference Pulse

The SYNC command is unlike other BDM commands because the host does not necessarily know the correct communication speed to use for BDM communications until after it has analyzed the response to the SYNC command. To issue a SYNC command, the host should perform the following steps:

1. Drive the BKGD pin low for at least 128 cycles at the lowest possible BDM serial communication frequency (the lowest serial communication frequency is determined by the crystal oscillator or the clock chosen by CLKSW.)
2. Drive BKGD high for a brief speedup pulse to get a fast rise time (this speedup pulse is typically one cycle of the host clock.)
3. Remove all drive to the BKGD pin so it reverts to high impedance.
4. Listen to the BKGD pin for the sync response pulse.

Upon detecting the SYNC request from the host, the target performs the following steps:

1. Discards any incomplete command received or bit retrieved.
2. Waits for BKGD to return to a logic one.
3. Delays 16 cycles to allow the host to stop driving the high speedup pulse.
4. Drives BKGD low for 128 cycles at the current BDM serial communication frequency.
5. Drives a one-cycle high speedup pulse to force a fast rise time on BKGD.
6. Removes all drive to the BKGD pin so it reverts to high impedance.

The host measures the low time of this 128 cycle SYNC response pulse and determines the correct speed for subsequent BDM communications. Typically, the host can determine the correct communication speed within a few percent of the actual target speed and the communication protocol can easily tolerate speed errors of several percent.

As soon as the SYNC request is detected by the target, any partially received command or bit retrieved is discarded. This is referred to as a soft-reset, equivalent to a time-out in the serial communication. After the SYNC response, the target will consider the next negative edge (issued by the host) as the start of a new BDM command or the start of new SYNC request.

Another use of the SYNC command pulse is to abort a pending ACK pulse. The behavior is exactly the same as in a regular SYNC command. Note that one of the possible causes for a command to not be acknowledged by the target is a host-target synchronization problem. In this case, the command may not have been understood by the target and so an ACK response pulse will not be issued.

21.4.10 Instruction Tracing

When a TRACE1 command is issued to the BDM in active BDM, the CPU exits the standard BDM firmware and executes a single instruction in the user code. Once this has occurred, the CPU is forced to return to the standard BDM firmware and the BDM is active and ready to receive a new command. If the TRACE1 command is issued again, the next user instruction will be executed. This facilitates stepping or tracing through the user code one instruction at a time.

If an interrupt is pending when a TRACE1 command is issued, the interrupt stacking operation occurs but no user instruction is executed. Once back in standard BDM firmware execution, the program counter points to the first instruction in the interrupt service routine.

Be aware when tracing through the user code that the execution of the user code is done step by step but all peripherals are free running. Hence possible timing relations between CPU code execution and occurrence of events of other peripherals no longer exist.

Do not trace the CPU instruction BGND used for soft breakpoints. Tracing the BGND instruction will result in a return address pointing to BDM firmware address space.

When tracing through user code which contains stop or wait instructions the following will happen when the stop or wait instruction is traced:

> The CPU enters stop or wait mode and the TRACE1 command can not be finished before leaving the low power mode. This is the case because BDM active mode can not be entered after CPU executed the stop instruction. However all BDM hardware commands except the BACKGROUND command are operational after tracing a stop or wait instruction and still being in stop or wait mode. If system stop mode is entered (all bus masters are in stop mode) no BDM command is operational.
>
> As soon as stop or wait mode is exited the CPU enters BDM active mode and the saved PC value points to the entry of the corresponding interrupt service routine.
>
> In case the handshake feature is enabled the corresponding ACK pulse of the TRACE1 command will be discarded when tracing a stop or wait instruction. Hence there is no ACK pulse when BDM active mode is entered as part of the TRACE1 command after CPU exited from stop or wait mode. All valid commands sent during CPU being in stop or wait mode or after CPU exited from stop or wait mode will have an ACK pulse. The handshake feature becomes disabled only when system

stop mode has been reached. Hence after a system stop mode the handshake feature must be enabled again by sending the ACK_ENABLE command.

21.4.11 Serial Communication Time Out

The host initiates a host-to-target serial transmission by generating a falling edge on the BKGD pin. If BKGD is kept low for more than 128 target clock cycles, the target understands that a SYNC command was issued. In this case, the target will keep waiting for a rising edge on BKGD in order to answer the SYNC request pulse. If the rising edge is not detected, the target will keep waiting forever without any time-out limit.

Consider now the case where the host returns BKGD to logic one before 128 cycles. This is interpreted as a valid bit transmission, and not as a SYNC request. The target will keep waiting for another falling edge marking the start of a new bit. If, however, a new falling edge is not detected by the target within 512 clock cycles since the last falling edge, a time-out occurs and the current command is discarded without affecting memory or the operating mode of the MCU. This is referred to as a soft-reset.

If a read command is issued but the data is not retrieved within 512 serial clock cycles, a soft-reset will occur causing the command to be disregarded. The data is not available for retrieval after the time-out has occurred. This is the expected behavior if the handshake protocol is not enabled. However, consider the behavior where the BDM is running in a frequency much greater than the CPU frequency. In this case, the command could time out before the data is ready to be retrieved. In order to allow the data to be retrieved even with a large clock frequency mismatch (between BDM and CPU) when the hardware handshake protocol is enabled, the time out between a read command and the data retrieval is disabled. Therefore, the host could wait for more then 512 serial clock cycles and still be able to retrieve the data from an issued read command. However, once the handshake pulse (ACK pulse) is issued, the time-out feature is re-activated, meaning that the target will time out after 512 clock cycles. Therefore, the host needs to retrieve the data within a 512 serial clock cycles time frame after the ACK pulse had been issued. After that period, the read command is discarded and the data is no longer available for retrieval. Any negative edge in the BKGD pin after the time-out period is considered to be a new command or a SYNC request.

Note that whenever a partially issued command, or partially retrieved data, has occurred the time out in the serial communication is active. This means that if a time frame higher than 512 serial clock cycles is observed between two consecutive negative edges and the command being issued or data being retrieved is not complete, a soft-reset will occur causing the partially received command or data retrieved to be disregarded. The next negative edge in the BKGD pin, after a soft-reset has occurred, is considered by the target as the start of a new BDM command, or the start of a SYNC request pulse.

Chapter 22
S12X Debug (S12XDBGV3) Module

Table 22-1. Revision History

Revision Number	Revision Date	Sections Affected	Description of Changes
V03.20	14 Sep 2007	22.3.2.7/22-777	- Clarified reserved State Sequencer encodings.
V03.21	23 Oct 2007	22.4.2.2/22-789 22.4.2.4/22-790	- Added single databyte comparison limitation information - Added statement about interrupt vector fetches whilst tagging.
V03.22	12 Nov 2007	22.4.5.2/22-794 22.4.5.5/22-801	- Removed LOOP1 tracing restriction NOTE. - Added pin reset effect NOTE.
V03.23	13 Nov 2007	General	- Text readability improved, typo removed.
V03.24	04 Jan 2008	22.4.5.3/22-796	- Corrected bit name.
V03.25	14 May 2008		- Updated Revision History Table format. Corrected other paragraph formats.

22.1 Introduction

The S12XDBG module provides an on-chip trace buffer with flexible triggering capability to allow non-intrusive debug of application software. The S12XDBG module is optimized for the S12X 16-bit architecture and allows debugging of CPU12Xand XGATE module operations.

Typically the S12XDBG module is used in conjunction with the S12XBDM module, whereby the user configures the S12XDBG module for a debugging session over the BDM interface. Once configured the S12XDBG module is armed and the device leaves BDM Mode returning control to the user program, which is then monitored by the S12XDBG module. Alternatively the S12XDBG module can be configured over a serial interface using SWI routines.

22.1.1 Glossary

Table 22-2. Glossary Of Terms

Term	Definition
COF	Change Of Flow. Change in the program flow due to a conditional branch, indexed jump or interrupt
BDM	Background Debug Mode
DUG	Device User Guide, describing the features of the device into which the DBG is integrated
WORD	16 bit data entity
Data Line	64 bit data entity

Table 22-2. Glossary Of Terms (continued)

Term	Definition
CPU	CPU12X module
Tag	Tags can be attached to XGATE or CPU opcodes as they enter the instruction pipe. If the tagged opcode reaches the execution stage a tag hit occurs.

22.1.2 Overview

The comparators monitor the bus activity of the CPU12X and XGATE. When a match occurs the control logic can trigger the state sequencer to a new state. On a transition to the Final State, bus tracing is triggered and/or a breakpoint can be generated.

Independent of comparator matches a transition to Final State with associated tracing and breakpoint can be triggered by the external $\overline{\text{TAGHI}}$ and $\overline{\text{TAGLO}}$ signals, or by an XGATE module S/W breakpoint request or by writing to the TRIG control bit.

The trace buffer is visible through a 2-byte window in the register address map and can be read out using standard 16-bit word reads. Tracing is disabled when the MCU system is secured.

22.1.3 Features

- Four comparators (A, B, C, and D)
  - Comparators A and C compare the full address bus and full 16-bit data bus
  - Comparators A and C feature a data bus mask register
  - Comparators B and D compare the full address bus only
  - Each comparator can be configured to monitor CPU12X or XGATE buses
  - Each comparator features selection of read or write access cycles
  - Comparators B and D allow selection of byte or word access cycles
  - Comparisons can be used as triggers for the state sequencer
- Three comparator modes
  - Simple address/data comparator match mode
  - Inside address range mode, Addmin ≤ Address ≤ Addmax
  - Outside address range match mode, Address < Addmin or Address > Addmax
- Two types of triggers
  - Tagged — This triggers just before a specific instruction begins execution
  - Force — This triggers on the first instruction boundary after a match occurs.
- The following types of breakpoints
  - CPU12X breakpoint entering BDM on breakpoint (BDM)
  - CPU12X breakpoint executing SWI on breakpoint (SWI)
  - XGATE breakpoint
- External CPU12X instruction tagging trigger independent of comparators
- XGATE S/W breakpoint request trigger independent of comparators

- TRIG Immediate software trigger independent of comparators
- Four trace modes
  - Normal: change of flow (COF) PC information is stored (see Section 22.4.5.2.1) for change of flow definition.
  - Loop1: same as Normal but inhibits consecutive duplicate source address entries
  - Detail: address and data for all cycles except free cycles and opcode fetches are stored
  - Pure PC: All program counter addresses are stored.
- 4-stage state sequencer for trace buffer control
  - Tracing session trigger linked to Final State of state sequencer
  - Begin, End, and Mid alignment of tracing to trigger

22.1.4 Modes of Operation

The S12XDBG module can be used in all MCU functional modes.

During BDM hardware accesses and whilst the BDM module is active, CPU12X monitoring is disabled. Thus breakpoints, comparators, and CPU12X bus tracing are disabled but XGATE bus monitoring accessing the S12XDBG registers, including comparator registers, is still possible. While in active BDM or during hardware BDM accesses, XGATE activity can still be compared, traced and can be used to generate a breakpoint to the XGATE module. When the CPU12X enters active BDM Mode through a BACKGROUND command, with the S12XDBG module armed, the S12XDBG remains armed.

The S12XDBG module tracing is disabled if the MCU is secure. However, breakpoints can still be generated if the MCU is secure.

Table 22-3. Mode Dependent Restriction Summary

BDM Enable	BDM Active	MCU Secure	Comparator Matches Enabled	Breakpoints Possible	Tagging Possible	Tracing Possible
x	x	1	Yes	Yes	Yes	No
0	0	0	Yes	Only SWI	Yes	Yes
0	1	0	Active BDM not possible when not enabled			
1	0	0	Yes	Yes	Yes	Yes
1	1	0	XGATE only	XGATE only	XGATE only	XGATE only

22.1.5 Block Diagram

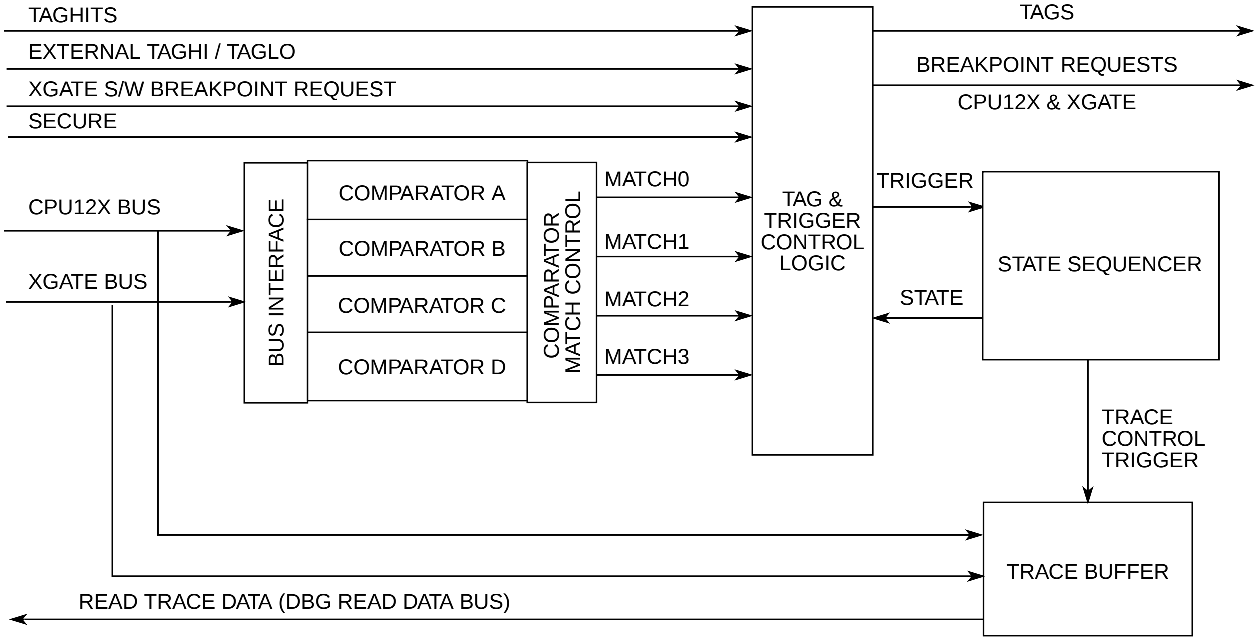


Figure 22-1. Debug Module Block Diagram

22.2 External Signal Description

The S12XDBG sub-module features two external tag input signals. See Device User Guide (DUG) for the mapping of these signals to device pins. These tag pins may be used for the external tagging in emulation modes only.

Table 22-4. External System Pins Associated With S12XDBG

Pin Name	Pin Functions	Description
$\overline{\text{TAGHI}}$ (See DUG)	TAGHI	When instruction tagging is on, tags the high half of the instruction word being read into the instruction queue.
$\overline{\text{TAGLO}}$ (See DUG)	TAGLO	When instruction tagging is on, tags the low half of the instruction word being read into the instruction queue.
$\overline{\text{TAGLO}}$ (See DUG)	Unconditional Tagging Enable	In emulation modes, a low assertion on this pin in the 7th or 8th cycle after the end of reset enables the Unconditional Tagging function.

22.3 Memory Map and Registers

22.3.1 Module Memory Map

A summary of the registers associated with the S12XDBG sub-block is shown in Table 22-2. Detailed descriptions of the registers and bits are given in the subsections that follow.

Address	Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0020	DBGC1	R	ARM	0	XGSBPE	BDM	DBGBRK		COMRV	
		W		TRIG						
0x0021	DBGSR	R	TBF	EXTF	0	0	0	SSF2	SSF1	SSF0
		W								
0x0022	DBGTCR	R	TSOURCE		TRANGE		TRCMOD		TALIGN	
		W								
0x0023	DBGC2	R	0	0	0	0	CDCM		ABCM	
		W								
0x0024	DBGTBH	R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		W								
0x0025	DBGTBL	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		W								
0x0026	DBGCNT	R	0	CNT						
		W								
0x0027	DBGSCRX	R	0	0	0	0	SC3	SC2	SC1	SC0
		W								
0x0027	DBGMFR	R	0	0	0	0	MC3	MC2	MC1	MC0
		W								
0x0028[1]	DBGXCTL (COMPA/C)	R	0	NDB	TAG	BRK	RW	RWE	SRC	COMPE
		W								
0x0028[2]	DBGXCTL (COMPB/D)	R	SZE	SZ	TAG	BRK	RW	RWE	SRC	COMPE
		W								
0x0029	DBGXAH	R	0	Bit 22	21	20	19	18	17	Bit 16
		W								
0x002A	DBGXAM	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x002B	DBGXAL	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x002C	DBGXDH	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x002D	DBGXDL	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x002E	DBGXDHM	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x002F	DBGXDLM	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								

[1] This represents the contents if the Comparator A or C control register is blended into this address.

[2] This represents the contents if the Comparator B or D control register is blended into this address

Figure 22-2. Quick Reference to S12XDBG Registers

22.3.2 Register Descriptions

This section consists of the S12XDBG control and trace buffer register descriptions in address order. Each comparator has a bank of registers that are visible through an 8-byte window between 0x0028 and 0x002F in the S12XDBG module register address map. When ARM is set in DBGC1, the only bits in the S12XDBG module registers that can be written are ARM, TRIG, and COMRV[1:0]

22.3.2.1 Debug Control Register 1 (DBGC1)

Address: 0x0020

	7	6	5	4	3	2	1	0
R	ARM	0	XGSBPE	BDM	DBGBRK		COMRV	
W		TRIG						
Reset	0	0	0	0	0	0	0	0

Figure 22-3. Debug Control Register (DBGC1)

Read: Anytime

Write: Bits 7, 1, 0 anytime
Bit 6 can be written anytime but always reads back as 0.
Bits 5:2 anytime S12XDBG is not armed.

NOTE

If a write access to DBGC1 with the ARM bit position set occurs simultaneously to a hardware disarm from an internal trigger event, then the ARM bit is cleared due to the hardware disarm.

NOTE

When disarming the S12XDBG by clearing ARM with software, the contents of bits[5:2] are not affected by the write, since up until the write operation, ARM = 1 preventing these bits from being written. These bits must be cleared using a second write if required.

Table 22-5. DBGC1 Field Descriptions

Field	Description
7 ARM	**Arm Bit** — The ARM bit controls whether the S12XDBG module is armed. This bit can be set and cleared by user software and is automatically cleared on completion of a tracing session, or if a breakpoint is generated with tracing not enabled. On setting this bit the state sequencer enters State1. 0 Debugger disarmed 1 Debugger armed
6 TRIG	**Immediate Trigger Request Bit** — This bit when written to 1 requests an immediate trigger independent of comparator or external tag signal status. When tracing is complete a forced breakpoint may be generated depending upon DBGBRK and BDM bit settings. This bit always reads back a 0. Writing a 0 to this bit has no effect. If TSOURCE are clear no tracing is carried out. If tracing has already commenced using BEGIN- or MID trigger alignment, it continues until the end of the tracing session as defined by the TALIGN bit settings, thus TRIG has no affect. In secure mode tracing is disabled and writing to this bit has no effect. 0 Do not trigger until the state sequencer enters the Final State. 1 Trigger immediately .

Table 22-5. DBGC1 Field Descriptions (continued)

Field	Description
5 XGSBPE	**XGATE S/W Breakpoint Enable** — The XGSBPE bit controls whether an XGATE S/W breakpoint request is passed to the CPU12X. The XGATE S/W breakpoint request is handled by the S12XDBG module, which can request an CPU12X breakpoint depending on the state of this bit. 0 XGATE S/W breakpoint request is disabled 1 XGATE S/W breakpoint request is enabled
4 BDM	**Background Debug Mode Enable** — This bit determines if an S12X breakpoint causes the system to enter Background Debug Mode (BDM) or initiate a Software Interrupt (SWI). If this bit is set but the BDM is not enabled by the ENBDM bit in the BDM module, then breakpoints default to SWI. 0 Breakpoint to Software Interrupt if BDM inactive. Otherwise no breakpoint. 1 Breakpoint to BDM, if BDM enabled. Otherwise breakpoint to SWI
3–2 DBGBRK	**S12XDBG Breakpoint Enable Bits** — The DBGBRK bits control whether the debugger will request a breakpoint to either CPU12X or XGATE or both upon reaching the state sequencer Final State. If tracing is enabled, the breakpoint is generated on completion of the tracing session. If tracing is not enabled, the breakpoint is generated immediately. Please refer to Section 22.4.7 for further details. XGATE software breakpoints are independent of the DBGBRK bits. XGATE software breakpoints force a breakpoint to the CPU12X independent of the DBGBRK bit field configuration. See Table 22-6.
1–0 COMRV	**Comparator Register Visibility Bits** — These bits determine which bank of comparator register is visible in the 8-byte window of the S12XDBG module address map, located between 0x0028 to 0x002F. Furthermore these bits determine which register is visible at the address 0x0027. See Table 22-7.

Table 22-6. DBGBRK Encoding

DBGBRK	Resource Halted by Breakpoint
00	No breakpoint generated
01	XGATE breakpoint generated
10	CPU12X breakpoint generated
11	Breakpoints generated for CPU12X and XGATE

Table 22-7. COMRV Encoding

COMRV	Visible Comparator	Visible Register at 0x0027
00	Comparator A	DBGSCR1
01	Comparator B	DBGSCR2
10	Comparator C	DBGSCR3
11	Comparator D	DBGMFR

22.3.2.2 Debug Status Register (DBGSR)

Address: 0x0021

	7	6	5	4	3	2	1	0
R	TBF	EXTF	0	0	0	SSF2	SSF1	SSF0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	—	0	0	0	0	0	0	0
POR	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 22-4. Debug Status Register (DBGSR)

Read: Anytime

Write: Never

Table 22-8. DBGSR Field Descriptions

Field	Description
7 TBF	**Trace Buffer Full** — The TBF bit indicates that the trace buffer has stored 64 or more lines of data since it was last armed. If this bit is set, then all 64 lines will be valid data, regardless of the value of DBGCNT bits CNT[6:0]. The TBF bit is cleared when ARM in DBGC1 is written to a one. The TBF is cleared by the power on reset initialization. Other system generated resets have no affect on this bit
6 EXTF	**External Tag Hit Flag** — The EXTF bit indicates if a tag hit condition from an external $\overline{\text{TAGHI}}$/$\overline{\text{TAGLO}}$ tag was met since arming. This bit is cleared when ARM in DBGC1 is written to a one. 0 External tag hit has not occurred 1 External tag hit has occurred
2–0 SSF[2:0]	**State Sequencer Flag Bits** — The SSF bits indicate in which state the State Sequencer is currently in. During a debug session on each transition to a new state these bits are updated. If the debug session is ended by software clearing the ARM bit, then these bits retain their value to reflect the last state of the state sequencer before disarming. If a debug session is ended by an internal trigger, then the state sequencer returns to state0 and these bits are cleared to indicate that state0 was entered during the session. On arming the module the state sequencer enters state1 and these bits are forced to SSF[2:0] = 001. See Table 22-9**.**

Table 22-9. SSF[2:0] — State Sequence Flag Bit Encoding

SSF[2:0]	Current State
000	State0 (disarmed)
001	State1
010	State2
011	State3
100	Final State
101,110,111	Reserved

22.3.2.3 Debug Trace Control Register (DBGTCR)

Address: 0x0022

	7	6	5	4	3	2	1	0
R W	TSOURCE		TRANGE		TRCMOD		TALIGN	
Reset	0	0	0	0	0	0	0	0

Figure 22-5. Debug Trace Control Register (DBGTCR)

Read: Anytime

Write: Bits 7:6 only when S12XDBG is neither secure nor armed.
Bits 5:0 anytime the module is disarmed.

Table 22-10. DBGTCR Field Descriptions

Field	Description
7–6 TSOURCE	**Trace Source Control Bits** — The TSOURCE bits select the data source for the tracing session. If the MCU system is secured, these bits cannot be set and tracing is inhibited. See Table 22-11.
5–4 TRANGE	**Trace Range Bits** — The TRANGE bits allow filtering of trace information from a selected address range when tracing from the CPU12X in Detail Mode. The XGATE tracing range cannot be narrowed using these bits. To use a comparator for range filtering, the corresponding COMPE and SRC bits must remain cleared. If the COMPE bit is not clear then the comparator will also be used to generate state sequence triggers. If the corresponding SRC bit is set the comparator is mapped to the XGATE buses, the TRANGE bits have no effect on the valid address range, memory accesses within the whole memory map are traced. See Table 22-12.
3–2 TRCMOD	**Trace Mode Bits** — See Section 22.4.5.2 for detailed Trace Mode descriptions. In Normal Mode, change of flow information is stored. In Loop1 Mode, change of flow information is stored but redundant entries into trace memory are inhibited. In Detail Mode, address and data for all memory and register accesses is stored. See Table 22-13.
1–0 TALIGN	**Trigger Align Bits** — These bits control whether the trigger is aligned to the beginning, end or the middle of a tracing session. See Table 22-14.

Table 22-11. TSOURCE — Trace Source Bit Encoding

TSOURCE	Tracing Source
00	No tracing requested
01	CPU12X
10(1)	XGATE
11[1],(2)	Both CPU12X and XGATE

1. No range limitations are allowed. Thus tracing operates as if TRANGE = 00.
2. No Detail Mode tracing supported. If TRCMOD = 10, no information is stored.

Table 22-12. TRANGE Trace Range Encoding

TRANGE	Tracing Range
00	Trace from all addresses (No filter)
01	Trace only in address range from $00000 to Comparator D
10	Trace only in address range from Comparator C to $7FFFFF
11	Trace only in range from Comparator C to Comparator D

Table 22-13. TRCMOD Trace Mode Bit Encoding

TRCMOD	Description
00	Normal
01	Loop1
10	Detail
11	Pure PC

Table 22-14. TALIGN Trace Alignment Encoding

TALIGN	Description
00	Trigger at end of stored data
01	Trigger before storing data
10	Trace buffer entries before and after trigger
11	Reserved

22.3.2.4 Debug Control Register2 (DBGC2)

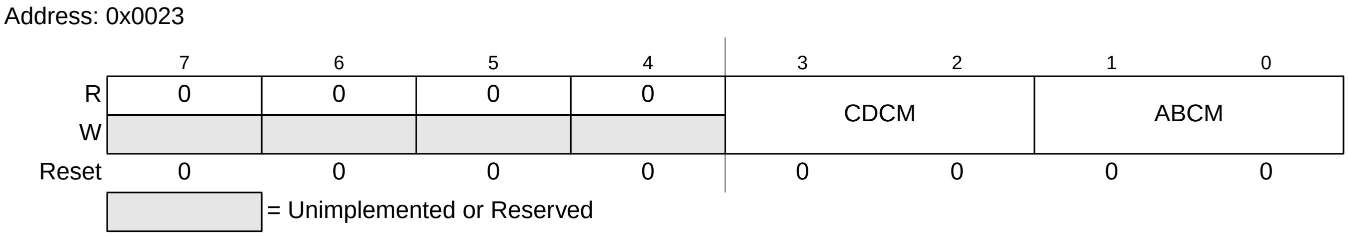


Figure 22-6. Debug Control Register2 (DBGC2)

Read: Anytime

Write: Anytime the module is disarmed.

This register configures the comparators for range matching.

Table 22-15. DBGC2 Field Descriptions

Field	Description
3–2 CDCM[1:0]	**C and D Comparator Match Control** — These bits determine the C and D comparator match mapping as described in Table 22-16.
1–0 ABCM[1:0]	**A and B Comparator Match Control** — These bits determine the A and B comparator match mapping as described in Table 22-17.

Table 22-16. CDCM Encoding

CDCM	Description
00	Match2 mapped to comparator C match....... Match3 mapped to comparator D match.
01	Match2 mapped to comparator C/D inside range....... Match3 disabled.
10	Match2 mapped to comparator C/D outside range....... Match3 disabled.
11	Reserved(1)

1. Currently defaults to Match2 mapped to comparator C : Match3 mapped to comparator D

Table 22-17. ABCM Encoding

ABCM	Description
00	Match0 mapped to comparator A match....... Match1 mapped to comparator B match.
01	Match 0 mapped to comparator A/B inside range....... Match1 disabled.
10	Match 0 mapped to comparator A/B outside range....... Match1 disabled.
11	Reserved(1)

1. Currently defaults to Match0 mapped to comparator A : Match1 mapped to comparator B

22.3.2.5 Debug Trace Buffer Register (DBGTBH:DBGTBL)

Address: 0x0024, 0x0025

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
POR	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Other Resets	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 22-7. Debug Trace Buffer Register (DBGTB)

Read: Only when unlocked AND not secured AND not armed AND with a TSOURCE bit set.

Write: Aligned word writes when disarmed unlock the trace buffer for reading but do not affect trace buffer contents.

Table 22-18. DBGTB Field Descriptions

Field	Description
15–0 Bit[15:0]	**Trace Buffer Data Bits** — The Trace Buffer Register is a window through which the 64-bit wide data lines of the Trace Buffer may be read 16 bits at a time. Each valid read of DBGTB increments an internal trace buffer pointer which points to the next address to be read. When the ARM bit is written to 1 the trace buffer is locked to prevent reading. The trace buffer can only be unlocked for reading by writing to DBGTB with an aligned word write when the module is disarmed. The DBGTB register can be read only as an aligned word, any byte reads or misaligned access of these registers will return 0 and will not cause the trace buffer pointer to increment to the next trace buffer address. The same is true for word reads while the debugger is armed. The POR state is undefined Other resets do not affect the trace buffer contents. .

22.3.2.6 Debug Count Register (DBGCNT)

Address: 0x0026

	7	6	5	4	3	2	1	0
R	0	CNT						
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	—	—	—	—	—	—	—
POR	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 22-8. Debug Count Register (DBGCNT)

Read: Anytime

Write: Never

Table 22-19. DBGCNT Field Descriptions

Field	Description
6–0 CNT[6:0]	**Count Value** — The CNT bits [6:0] indicate the number of valid data 64-bit data lines stored in the Trace Buffer. Table 22-20 shows the correlation between the CNT bits and the number of valid data lines in the Trace Buffer. When the CNT rolls over to zero, the TBF bit in DBGSR is set and incrementing of CNT will continue in end-trigger or mid-trigger mode. The DBGCNT register is cleared when ARM in DBGC1 is written to a one. The DBGCNT register is cleared by power-on-reset initialization but is not cleared by other system resets. Thus should a reset occur during a debug session, the DBGCNT register still indicates after the reset, the number of valid trace buffer entries stored before the reset occurred. The DBGCNT register is not decremented when reading from the trace buffer.

Table 22-20. CNT Decoding Table

TBF (DBGSR)	CNT[6:0]	Description
0	0000000	No data valid
0	0000001	32 bits of one line valid(1)
0	0000010 0000100 0000110 .. 1111100	1 line valid 2 lines valid 3 lines valid .. 62 lines valid
0	1111110	63 lines valid
1	0000000	64 lines valid; if using Begin trigger alignment, ARM bit will be cleared and the tracing session ends.
1	0000010 1111110	64 lines valid, oldest data has been overwritten by most recent data

1. This applies to Normal/Loop1/PurePC Modes when tracing from either CPU12X or XGATE only.

22.3.2.7 Debug State Control Registers

There is a dedicated control register for each of the state sequencer states 1 to 3 that determines if transitions from that state are allowed, depending upon comparator matches or tag hits, and defines the next state for the state sequencer following a match. The three debug state control registers are located at the same address in the register address map (0x0027). Each register can be accessed using the COMRV bits in DBGC1 to blend in the required register. The COMRV = 11 value blends in the match flag register (DBGMFR).

Table 22-21. State Control Register Access Encoding

COMRV	Visible State Control Register
00	DBGSCR1
01	DBGSCR2
10	DBGSCR3
11	DBGMFR

22.3.2.7.1 Debug State Control Register 1 (DBGSCR1)

Address: 0x0027

	7	6	5	4	3	2	1	0
R	0	0	0	0	SC3	SC2	SC1	SC0
W	▨	▨	▨	▨				
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 22-9. Debug State Control Register 1 (DBGSCR1)

Read: If COMRV[1:0] = 00

Write: If COMRV[1:0] = 00 and S12XDBG is not armed.

This register is visible at 0x0027 only with COMRV[1:0] = 00. The state control register 1 selects the targeted next state whilst in State1. The matches refer to the match channels of the comparator match control logic as depicted in Figure 22-1 and described in Section 22.3.2.8.1”. Comparators must be enabled by setting the comparator enable bit in the associated DBGXCTL control register.

Table 22-22. DBGSCR1 Field Descriptions

Field	Description
3–0 SC[3:0]	These bits select the targeted next state whilst in State1, based upon the match event.

Table 22-23. State1 Sequencer Next State Selection

SC[3:0]	Description
0000	Any match triggers to state2
0001	Any match triggers to state3
0010	Any match triggers to Final State
0011	Match2 triggers to State2....... Other matches have no effect

Table 22-23. State1 Sequencer Next State Selection (continued)

SC[3:0]	Description
0100	Match2 triggers to State3....... Other matches have no effect
0101	Match2 triggers to Final State....... Other matches have no effect
0110	Match0 triggers to State2....... Match1 triggers to State3....... Other matches have no effect
0111	Match1 triggers to State3....... Match0 triggers Final State....... Other matches have no effect
1000	Match0 triggers to State2....... Match2 triggers to State3....... Other matches have no effect
1001	Match2 triggers to State3....... Match0 triggers Final State....... Other matches have no effect
1010	Match1 triggers to State2....... Match3 triggers to State3....... Other matches have no effect
1011	Match3 triggers to State3....... Match1 triggers to Final State....... Other matches have no effect
1100	Match3 has no effect....... All other matches (M0,M1,M2) trigger to State2
1101	Reserved. (No match triggers state sequencer transition)
1110	Reserved. (No match triggers state sequencer transition)
1111	Reserved. (No match triggers state sequencer transition)

The trigger priorities described in Table 22-42 dictate that in the case of simultaneous matches, the match on the lower channel number (0,1,2,3) has priority. The SC[3:0] encoding ensures that a match leading to final state has priority over all other matches.

22.3.2.7.2 Debug State Control Register 2 (DBGSCR2)

Address: 0x0027

	7	6	5	4	3	2	1	0
R	0	0	0	0	SC3	SC2	SC1	SC0
W	▨	▨	▨	▨				
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 22-10. Debug State Control Register 2 (DBGSCR2)

Read: If COMRV[1:0] = 01

Write: If COMRV[1:0] = 01 and S12XDBG is not armed.

This register is visible at 0x0027 only with COMRV[1:0] = 01. The state control register 2 selects the targeted next state whilst in State2. The matches refer to the match channels of the comparator match control logic as depicted in Figure 22-1 and described in Section 22.3.2.8.1”. Comparators must be enabled by setting the comparator enable bit in the associated DBGXCTL control register.

Table 22-24. DBGSCR2 Field Descriptions

Field	Description
3–0 SC[3:0]	These bits select the targeted next state whilst in State2, based upon the match event.

Table 22-25. State2 —Sequencer Next State Selection

SC[3:0]	Description
0000	Any match triggers to state1
0001	Any match triggers to state3

Table 22-25. State2 —Sequencer Next State Selection (continued)

SC[3:0]	Description
0010	Any match triggers to Final State
0011	Match3 triggers to State1....... Other matches have no effect
0100	Match3 triggers to State3....... Other matches have no effect
0101	Match3 triggers to Final State....... Other matches have no effect
0110	Match0 triggers to State1....... Match1 triggers to State3....... Other matches have no effect
0111	Match1 triggers to State3....... Match0 triggers Final State....... Other matches have no effect
1000	Match0 triggers to State1....... Match2 triggers to State3....... Other matches have no effect
1001	Match2 triggers to State3....... Match0 triggers Final State....... Other matches have no effect
1010	Match1 triggers to State1....... Match3 triggers to State3....... Other matches have no effect
1011	Match3 triggers to State3....... Match1 triggers Final State....... Other matches have no effect
1100	Match2 triggers to State1..... Match3 trigger to Final State
1101	Match2 has no affect, all other matches (M0,M1,M3) trigger to Final State
1110	Reserved. (No match triggers state sequencer transition)
1111	Reserved. (No match triggers state sequencer transition)

The trigger priorities described in Table 22-42 dictate that in the case of simultaneous matches, the match on the lower channel number (0,1,2,3) has priority. The SC[3:0] encoding ensures that a match leading to final state has priority over all other matches.

22.3.2.7.3 Debug State Control Register 3 (DBGSCR3)

Address: 0x0027

	7	6	5	4	3	2	1	0
R	0	0	0	0	SC3	SC2	SC1	SC0
W	▨	▨	▨	▨				
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 22-11. Debug State Control Register 3 (DBGSCR3)

Read: If COMRV[1:0] = 10

Write: If COMRV[1:0] = 10 and S12XDBG is not armed.

This register is visible at 0x0027 only with COMRV[1:0] = 10. The state control register three selects the targeted next state whilst in State3. The matches refer to the match channels of the comparator match control logic as depicted in Figure 22-1 and described in Section 22.3.2.8.1”. Comparators must be enabled by setting the comparator enable bit in the associated DBGXCTL control register.

Table 22-26. DBGSCR3 Field Descriptions

Field	Description
3–0 SC[3:0]	These bits select the targeted next state whilst in State3, based upon the match event.

Table 22-27. State3 — Sequencer Next State Selection

SC[3:0]	Description
0000	Any match triggers to state1
0001	Any match triggers to state2
0010	Any match triggers to Final State
0011	Match0 triggers to State1....... Other matches have no effect
0100	Match0 triggers to State2....... Other matches have no effect
0101	Match0 triggers to Final State.......Match1 triggers to State1...Other matches have no effect
0110	Match1 triggers to State1....... Other matches have no effect
0111	Match1 triggers to State2....... Other matches have no effect
1000	Match1 triggers to Final State....... Other matches have no effect
1001	Match2 triggers to State2....... Match0 triggers to Final State....... Other matches have no effect
1010	Match1 triggers to State1....... Match3 triggers to State2....... Other matches have no effect
1011	Match3 triggers to State2....... Match1 triggers to Final State....... Other matches have no effect
1100	Match2 triggers to Final State....... Other matches have no effect
1101	Match3 triggers to Final State....... Other matches have no effect
1110	Reserved. (No match triggers state sequencer transition)
1111	Reserved. (No match triggers state sequencer transition)

The trigger priorities described in Table 22-42 dictate that in the case of simultaneous matches, the match on the lower channel number (0,1,2,3) has priority. The SC[3:0] encoding ensures that a match leading to final state has priority over all other matches.

22.3.2.7.4 Debug Match Flag Register (DBGMFR)

Address: 0x0027

	7	6	5	4	3	2	1	0
R	0	0	0	0	MC3	MC2	MC1	MC0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 22-12. Debug Match Flag Register (DBGMFR)

Read: If COMRV[1:0] = 11

Write: Never

DBGMFR is visible at 0x0027 only with COMRV[1:0] = 11. It features four flag bits each mapped directly to a channel. Should a match occur on the channel during the debug session, then the corresponding flag is set and remains set until the next time the module is armed by writing to the ARM bit. Thus the contents are retained after a debug session for evaluation purposes. These flags cannot be cleared by software, they are cleared only when arming the module. A set flag does not inhibit the setting of other flags. Once a flag is set, further triggers on the same channel have no affect.

22.3.2.8 Comparator Register Descriptions

Each comparator has a bank of registers that are visible through an 8-byte window in the S12XDBG module register address map. Comparators A and C consist of 8 register bytes (3 address bus compare registers, two data bus compare registers, two data bus mask registers and a control register).

Comparators B and D consist of four register bytes (three address bus compare registers and a control register).

Each set of comparator registers is accessible in the same 8-byte window of the register address map and can be accessed using the COMRV bits in the DBGC1 register. If the Comparators B or D are accessed through the 8-byte window, then only the address and control bytes are visible, the 4 bytes associated with data bus and data bus masking read as zero and cannot be written. Furthermore the control registers for comparators B and D differ from those of comparators A and C.

Table 22-28. Comparator Register Layout

0x0028	CONTROL	Read/Write	Comparators A,B,C,D
0x0029	ADDRESS HIGH	Read/Write	Comparators A,B,C,D
0x002A	ADDRESS MEDIUM	Read/Write	Comparators A,B,C,D
0x002B	ADDRESS LOW	Read/Write	Comparators A,B,C,D
0x002C	DATA HIGH COMPARATOR	Read/Write	Comparator A and C only
0x002D	DATA LOW COMPARATOR	Read/Write	Comparator A and C only
0x002E	DATA HIGH MASK	Read/Write	Comparator A and C only
0x002F	DATA LOW MASK	Read/Write	Comparator A and C only

22.3.2.8.1 Debug Comparator Control Register (DBGXCTL)

The contents of this register bits 7 and 6 differ depending upon which comparator registers are visible in the 8-byte window of the DBG module register address map.

Address: 0x0028

	7	6	5	4	3	2	1	0
R	0	NDB	TAG	BRK	RW	RWE	SRC	COMPE
W	▨							
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 22-13. Debug Comparator Control Register (Comparators A and C)

Address: 0x0028

	7	6	5	4	3	2	1	0
R W	SZE	SZ	TAG	BRK	RW	RWE	SRC	COMPE
Reset	0	0	0	0	0	0	0	0

Figure 22-14. Debug Comparator Control Register (Comparators B and D)

Read: Anytime. See Table 22-29 for visible register encoding.

Write: If DBG not armed. See Table 22-29 for visible register encoding.

The DBGC1_COMRV bits determine which comparator control, address, data and datamask registers are visible in the 8-byte window from 0x0028 to 0x002F as shown in Section Table 22-29.

Table 22-29. Comparator Address Register Visibility

COMRV	Visible Comparator
00	DBGACTL, DBGAAH ,DBGAAM, DBGAAL, DBGADH, DBGADL, DBGADHM, DBGADLM
01	DBGBCTL, DBGBAH, DBGBAM, DBGBAL
10	DBGCCTL, DBGCAH, DBGCAM, DBGCAL, DBGCDH, DBGCDL, DBGCDHM, DBGCDLM
11	DBGDCTL, DBGDAH, DBGDAM, DBGDAL

Table 22-30. DBGXCTL Field Descriptions

Field	Description
7 SZE (Comparators B and D)	**Size Comparator Enable Bit** — The SZE bit controls whether access size comparison is enabled for the associated comparator. This bit is ignored if the TAG bit in the same register is set. 0 Word/Byte access size is not used in comparison 1 Word/Byte access size is used in comparison
6 NDB (Comparators A and C	**Not Data Bus** — The NDB bit controls whether the match occurs when the data bus matches the comparator register value or when the data bus differs from the register value. Furthermore data bus bits can be individually masked using the comparator data mask registers. This bit is only available for comparators A and C. This bit is ignored if the TAG bit in the same register is set. This bit position has an SZ functionality for comparators B and D. 0 Match on data bus equivalence to comparator register contents 1 Match on data bus difference to comparator register contents
6 SZ (Comparators B and D)	**Size Comparator Value Bit** — The SZ bit selects either word or byte access size in comparison for the associated comparator. This bit is ignored if the SZE bit is cleared or if the TAG bit in the same register is set. This bit position has NDB functionality for comparators A and C 0 Word access size will be compared 1 Byte access size will be compared
5 TAG	**Tag Select** — This bit controls whether the comparator match will cause a trigger or tag the opcode at the matched address. Tagged opcodes trigger only if they reach the execution stage of the instruction queue. 0 Trigger immediately on match 1 On match, tag the opcode. If the opcode is about to be executed a trigger is generated
4 BRK	**Break** — This bit controls whether a channel match terminates a debug session immediately, independent of state sequencer state. To generate an immediate breakpoint the module breakpoints must be enabled using DBGBRK. 0 The debug session termination is dependent upon the state sequencer and trigger conditions. 1 A match on this channel terminates the debug session immediately; breakpoints if active are generated, tracing, if active, is terminated and the module disarmed.
3 RW	**Read/Write Comparator Value Bit** — The RW bit controls whether read or write is used in compare for the associated comparator . The RW bit is not used if RWE = 0. 0 Write cycle will be matched 1 Read cycle will be matched
2 RWE	**Read/Write Enable Bit** — The RWE bit controls whether read or write comparison is enabled for the associated comparator. This bit is not used for tagged operations. 0 Read/Write is not used in comparison 1 Read/Write is used in comparison

Table 22-30. DBGXCTL Field Descriptions (continued)

Field	Description
1 SRC	Determines mapping of comparator to CPU12X or XGATE 0 The comparator is mapped to CPU12X buses 1 The comparator is mapped to XGATE address and data buses
0 COMPE	Determines if comparator is enabled 0 The comparator is not enabled 1 The comparator is enabled for state sequence triggers or tag generation

Table 22-31 shows the effect for RWE and RW on the comparison conditions. These bits are not useful for tagged operations since the trigger occurs based on the tagged opcode reaching the execution stage of the instruction queue. Thus these bits are ignored if tagged triggering is selected.

Table 22-31. Read or Write Comparison Logic Table

RWE Bit	RW Bit	RW Signal	Comment
0	x	0	RW not used in comparison
0	x	1	RW not used in comparison
1	0	0	Write
1	0	1	No match
1	1	0	No match
1	1	1	Read

22.3.2.8.2 Debug Comparator Address High Register (DBGXAH)

Address: 0x0029

	7	6	5	4	3	2	1	0
R	0	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
W	▨							
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 22-15. Debug Comparator Address High Register (DBGXAH)

Read: Anytime. See Table 22-29 for visible register encoding.

Write: If DBG not armed. See Table 22-29 for visible register encoding.

Table 22-32. DBGXAH Field Descriptions

Field	Description
6–0 Bit[22:16]	**Comparator Address High Compare Bits** — The Comparator address high compare bits control whether the selected comparator will compare the address bus bits [22:16] to a logic one or logic zero. This register byte is ignored for XGATE compares. 0 Compare corresponding address bit to a logic zero 1 Compare corresponding address bit to a logic one

22.3.2.8.3 Debug Comparator Address Mid Register (DBGXAM)

Address: 0x002A

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

Figure 22-16. Debug Comparator Address Mid Register (DBGXAM)

Read: Anytime. See Table 22-29 for visible register encoding.

Write: If DBG not armed. See Table 22-29 for visible register encoding.

Table 22-33. DBGXAM Field Descriptions

Field	Description
7–0 Bit[15:8]	**Comparator Address Mid Compare Bits**— The Comparator address mid compare bits control whether the selected comparator will compare the address bus bits [15:8] to a logic one or logic zero. 0 Compare corresponding address bit to a logic zero 1 Compare corresponding address bit to a logic one

22.3.2.8.4 Debug Comparator Address Low Register (DBGXAL)

Address: 0x002B

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 22-17. Debug Comparator Address Low Register (DBGXAL)

Read: Anytime. See Table 22-29 for visible register encoding.

Write: If DBG not armed. See Table 22-29 for visible register encoding.

Table 22-34. DBGXAL Field Descriptions

Field	Description
7–0 Bits[7:0]	**Comparator Address Low Compare Bits** — The Comparator address low compare bits control whether the selected comparator will compare the address bus bits [7:0] to a logic one or logic zero. 0 Compare corresponding address bit to a logic zero 1 Compare corresponding address bit to a logic one

22.3.2.8.5 Debug Comparator Data High Register (DBGXDH)

Address: 0x002C

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

Figure 22-18. Debug Comparator Data High Register (DBGXDH)

Read: Anytime. See Table 22-29 for visible register encoding.

Write: If DBG not armed. See Table 22-29 for visible register encoding.

Table 22-35. DBGXAH Field Descriptions

Field	Description
7–0 Bits[15:8]	**Comparator Data High Compare Bits** — The Comparator data high compare bits control whether the selected comparator compares the data bus bits [15:8] to a logic one or logic zero. The comparator data compare bits are only used in comparison if the corresponding data mask bit is logic 1. This register is available only for comparators A and C. 0 Compare corresponding data bit to a logic zero 1 Compare corresponding data bit to a logic one

22.3.2.8.6 Debug Comparator Data Low Register (DBGXDL)

Address: 0x002D

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 22-19. Debug Comparator Data Low Register (DBGXDL)

Read: Anytime. See Table 22-29 for visible register encoding.

Write: If DBG not armed. See Table 22-29 for visible register encoding.

Table 22-36. DBGXDL Field Descriptions

Field	Description
7–0 Bits[7:0]	**Comparator Data Low Compare Bits** — The Comparator data low compare bits control whether the selected comparator compares the data bus bits [7:0] to a logic one or logic zero. The comparator data compare bits are only used in comparison if the corresponding data mask bit is logic 1. This register is available only for comparators A and C. 0 Compare corresponding data bit to a logic zero 1 Compare corresponding data bit to a logic one

22.3.2.8.7 Debug Comparator Data High Mask Register (DBGXDHM)

Address: 0x002E

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

Figure 22-20. Debug Comparator Data High Mask Register (DBGXDHM)

Read: Anytime. See Table 22-29 for visible register encoding.

Write: If DBG not armed. See Table 22-29 for visible register encoding.

Table 22-37. DBGXDHM Field Descriptions

Field	Description
7–0 Bits[15:8]	**Comparator Data High Mask Bits** — The Comparator data high mask bits control whether the selected comparator compares the data bus bits [15:8] to the corresponding comparator data compare bits. This register is available only for comparators A and C. 0 Do not compare corresponding data bit 1 Compare corresponding data bit

22.3.2.8.8 Debug Comparator Data Low Mask Register (DBGXDLM)

Address: 0x002F

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 22-21. Debug Comparator Data Low Mask Register (DBGXDLM)

Read: Anytime. See Table 22-29 for visible register encoding.

Write: If DBG not armed. See Table 22-29 for visible register encoding.

Table 22-38. DBGXDLM Field Descriptions

Field	Description
7–0 Bits[7:0]	**Comparator Data Low Mask Bits** — The Comparator data low mask bits control whether the selected comparator compares the data bus bits [7:0] to the corresponding comparator data compare bits. This register is available only for comparators A and C. 0 Do not compare corresponding data bit 1 Compare corresponding data bit

22.4 Functional Description

This section provides a complete functional description of the S12XDBG module. If the part is in secure mode, the S12XDBG module can generate breakpoints but tracing is not possible.

22.4.1 S12XDBG Operation

Arming the S12XDBG module by setting ARM in DBGC1 allows triggering, and storing of data in the trace buffer and can be used to cause breakpoints to the CPU12X or the XGATE module. The DBG module is made up of four main blocks, the comparators, control logic, the state sequencer, and the trace buffer.

The comparators monitor the bus activity of the CPU12X and XGATE. Comparators can be configured to monitor address and databus. Comparators can also be configured to mask out individual data bus bits during a compare and to use R/W and word/byte access qualification in the comparison. When a match with a comparator register value occurs the associated control logic can trigger the state sequencer to another state (see Figure 22-22). Either forced or tagged triggers are possible. Using a forced trigger, the trigger is generated immediately on a comparator match. Using a tagged trigger, at a comparator match, the instruction opcode is tagged and only if the instruction reaches the execution stage of the instruction queue is a trigger generated. In the case of a transition to Final State, bus tracing is triggered and/or a breakpoint can be generated. Tracing of both CPU12X and/or XGATE bus activity is possible.

Independent of the state sequencer, a breakpoint can be triggered by the external $\overline{\text{TAGHI}}$ / $\overline{\text{TAGLO}}$ signals or by an XGATE S/W breakpoint request or by writing to the TRIG bit in the DBGC1 control register.

The trace buffer is visible through a 2-byte window in the register address map and can be read out using standard 16-bit word reads.

22.4.2 Comparator Modes

The S12XDBG contains four comparators, A, B, C, and D. Each comparator can be configured to monitor CPU12X or XGATE buses. Each comparator compares the selected address bus with the address stored in DBGXAH, DBGXAM, and DBGXAL. Furthermore, comparators A and C also compare the data buses to the data stored in DBGXDH, DBGXDL and allow masking of individual data bus bits.

S12X comparator matches are disabled in BDM and during BDM accesses.

The comparator match control logic configures comparators to monitor the buses for an exact address or an address range. The comparator configuration is controlled by the control register contents and the range control by the DBGC2 contents.

On a match a trigger can initiate a transition to another state sequencer state (see Section 22.4.3”). The comparator control register also allows the type of access to be included in the comparison through the use of the RWE, RW, SZE, and SZ bits. The RWE bit controls whether read or write comparison is enabled for the associated comparator and the RW bit selects either a read or write access for a valid match. Similarly the SZE and SZ bits allows the size of access (word or byte) to be considered in the compare. Only comparators B and D feature SZE and SZ.

The TAG bit in each comparator control register is used to determine the triggering condition. By setting TAG, the comparator will qualify a match with the output of opcode tracking logic and a trigger occurs before the tagged instruction executes (tagged-type trigger). Whilst tagging, the RW, RWE, SZE, and SZ bits are ignored and the comparator register must be loaded with the exact opcode address.

If the TAG bit is clear (forced type trigger) a comparator match is generated when the selected address appears on the system address bus. If the selected address is an opcode address, the match is generated when the opcode is fetched from the memory. This precedes the instruction execution by an indefinite number of cycles due to instruction pipe lining. For a comparator match of an opcode at an odd address when TAG = 0, the corresponding even address must be contained in the comparator register. Thus for an opcode at odd address (n), the comparator register must contain address (n–1).

Once a successful comparator match has occurred, the condition that caused the original match is not verified again on subsequent matches. Thus if a particular data value is verified at a given address, this address may not still contain that data value when a subsequent match occurs.

Comparators C and D can also be used to select an address range to trace from. This is determined by the TRANGE bits in the DBGTCR register. The TRANGE encoding is shown in Table 22-12. If the TRANGE bits select a range definition using comparator D, then comparator D is configured for trace range definition and cannot be used for address bus comparisons. Similarly if the TRANGE bits select a range definition using comparator C, then comparator C is configured for trace range definition and cannot be used for address bus comparisons.

Match[0, 1, 2, 3] map directly to Comparators[A, B, C, D] respectively, except in range modes (see Section 22.3.2.4”). Comparator priority rules are described in the trigger priority section (Section 22.4.3.6”).

22.4.2.1 Exact Address Comparator Match (Comparators A and C)

With range comparisons disabled, the match condition is an exact equivalence of address/data bus with the value stored in the comparator address/data registers. Further qualification of the type of access (R/W, word/byte) is possible.

Comparators A and C do not feature SZE or SZ control bits, thus the access size is not compared. Table 22-40 lists access considerations without data bus compare. Table 22-39 lists access considerations with data bus comparison. To compare byte accesses DBGxDH must be loaded with the data byte, the low byte must be masked out using the DBGxDLM mask register. On word accesses the data byte of the lower address is mapped to DBGxDH.

Table 22-39. Comparator A and C Data Bus Considerations

Access	Address	DBGxDH	DBGxDL	DBGxDHM	DBGxDLM	Example Valid Match	
Word	ADDR[n]	Data[n]	Data[n+1]	$FF	$FF	MOVW #$WORD ADDR[n]	config1
Byte	ADDR[n]	Data[n]	x	$FF	$00	MOVB #$BYTE ADDR[n]	config2
Word	ADDR[n]	Data[n]	x	$FF	$00	MOVW #$WORD ADDR[n]	config2
Word	ADDR[n]	x	Data[n+1]	$00	$FF	MOVW #$WORD ADDR[n]	config3

Code may contain various access forms of the same address, i.e. a word access of ADDR[n] or byte access of ADDR[n+1] both access n+1. At a word access of ADDR[n], address ADDR[n+1] does not appear on the address bus and so cannot cause a comparator match if the comparator contains ADDR[n]. Thus it is not possible to monitor all data accesses of ADDR[n+1] with one comparator.

To detect an access of ADDR[n+1] through a word access of ADDR[n] the comparator can be configured to ADDR[n], DBGxDL is loaded with the data pattern and DBGxDHM is cleared so only the data[n+1] is compared on accesses of ADDR[n].

NOTE

Using this configuration, a byte access of ADDR[n] can cause a comparator match if the databus low byte by chance contains the same value as ADDR[n+1] because the databus comparator does not feature access size comparison and uses the mask as a “don’t care” function. Thus masked bits do not prevent a match.

Comparators A and C feature an NDB control bit to determine if a match occurs when the data bus differs to comparator register contents or when the data bus is equivalent to the comparator register contents.

22.4.2.2 Exact Address Comparator Match (Comparators B and D)

Comparators B and D feature SZ and SZE control bits. If SZE is clear, then the comparator address match qualification functions the same as for comparators A and C.

If the SZE bit is set the access size (word or byte) is compared with the SZ bit value such that only the specified type of access causes a match. Thus if configured for a byte access of a particular address, a word access covering the same address does not lead to match.

Table 22-40. Comparator Access Size Considerations

Comparator	Address	SZE	SZ8	Condition For Valid Match
Comparators A and C	ADDR[n]	—	—	Word and byte accesses of ADDR[n](1) MOVB #$BYTE ADDR[n] MOVW #$WORD ADDR[n]
Comparators B and D	ADDR[n]	0	X	Word and byte accesses of ADDR[n][1] MOVB #$BYTE ADDR[n] MOVW #$WORD ADDR[n]
Comparators B and D	ADDR[n]	1	0	Word accesses of ADDR[n][1] MOVW #$WORD ADDR[n]
Comparators B and D	ADDR[n]	1	1	Byte accesses of ADDR[n] MOVB #$BYTE ADDR[n]

1. A word access of ADDR[n-1] also accesses ADDR[n] but does not generate a match.
The comparator address register must contain the exact address used in the code.

22.4.2.3 Data Bus Comparison NDB Dependency

Comparators A and C each feature an NDB control bit, which allows data bus comparators to be configured to either trigger on equivalence or trigger on difference. This allows monitoring of a difference in the contents of an address location from an expected value.

When matching on an equivalence (NDB=0), each individual data bus bit position can be masked out by clearing the corresponding mask bit (DBGxDHM/DBGxDLM), so that it is ignored in the comparison. A match occurs when all data bus bits with corresponding mask bits set are equivalent. If all mask register bits are clear, then a match is based on the address bus only, the data bus is ignored.

When matching on a difference, mask bits can be cleared to ignore bit positions. A match occurs when any data bus bit with corresponding mask bit set is different. Clearing all mask bits, causes all bits to be ignored and prevents a match because no difference can be detected. In this case address bus equivalence does not cause a match.

Table 22-41. NDB and MASK bit dependency

NDB	DBGxDHM[n] / DBGxDLM[n]	Comment
0	0	Do not compare data bus bit.
0	1	Compare data bus bit. Match on equivalence.
1	0	Do not compare data bus bit.
1	1	Compare data bus bit. Match on difference.

22.4.2.4 Range Comparisons

When using the AB comparator pair for a range comparison, the data bus can also be used for qualification by using the comparator A data and data mask registers. Furthermore the DBGACTL RW and RWE bits can be used to qualify the range comparison on either a read or a write access. The corresponding DBGBCTL bits are ignored. Similarly when using the CD comparator pair for a range comparison, the data bus can also be used for qualification by using the comparator C data and data mask registers. Furthermore the DBGCCTL RW and RWE bits can be used to qualify the range comparison on either a read or a write access if tagging is not selected. The corresponding DBGDCTL bits are ignored. The SZE and SZ control bits are ignored in range mode. The comparator A and C TAG bits are used to tag range comparisons for the AB and CD ranges respectively. The comparator B and D TAG bits are ignored in range modes. In order for a range comparison using comparators A and B, both COMPEA and COMPEB must be set; to disable range comparisons both must be cleared. Similarly for a range CD comparison, both COMPEC and COMPED must be set. If a range mode is selected SRCA and SRCC select the source (S12X or XGATE), SRCB and SRCD are ignored. The comparator A and C BRK bits are used for the AB and CD ranges respectively, the comparator B and D BRK bits are ignored in range mode. When configured for range comparisons and tagging, the ranges are accurate only to word boundaries.

22.4.2.4.1 Inside Range (CompAC_Addr $\leq$ address $\leq$ CompBD_Addr)

In the Inside Range comparator mode, either comparator pair A and B or comparator pair C and D can be configured for range comparisons by the control register (DBGC2). The match condition requires that a valid match for both comparators happens on the same bus cycle. A match condition on only one comparator is not valid. An aligned word access which straddles the range boundary will cause a trigger only if the aligned address is inside the range.

22.4.2.4.2 Outside Range (address < CompAC_Addr or address > CompBD_Addr)

In the Outside Range comparator mode, either comparator pair A and B or comparator pair C and D can be configured for range comparisons. A single match condition on either of the comparators is recognized as valid. An aligned word access which straddles the range boundary will cause a trigger only if the aligned address is outside the range.

Outside range mode in combination with tagged triggers can be used to detect if the opcode fetches are from an unexpected range. In forced trigger modes the outside range trigger would typically be activated at any interrupt vector fetch or register access. This can be avoided by setting the upper or lower range limit to $7FFFFF or $000000 respectively. Interrupt vector fetches do not cause taghits

When comparing the XGATE address bus in outside range mode, the initial vector fetch as determined by the vector contained in the XGATE XGVBR register should be taken into consideration. The XGVBR register and hence vector address can be modified.

22.4.3 Trigger Modes

Trigger modes are used as qualifiers for a state sequencer change of state. The control logic determines the trigger mode and provides a trigger to the state sequencer. The individual trigger modes are described in the following sections.

22.4.3.1 Forced Trigger On Comparator Match

If a forced trigger comparator match occurs, the trigger immediately initiates a transition to the next state sequencer state whereby the corresponding flags in DBGSR are set. The state control register for the current state determines the next state for each trigger. Forced triggers are generated as soon as the matching address appears on the address bus, which in the case of opcode fetches occurs several cycles before the opcode execution. For this reason a forced trigger at an opcode address precedes a tagged trigger at the same address by several cycles.

22.4.3.2 Trigger On Comparator Related Taghit

If a CPU12X or XGATE taghit occurs, a transition to another state sequencer state is initiated and the corresponding DBGSR flags are set. For a comparator related taghit to occur, the S12XDBG must first generate tags based on comparator matches. When the tagged instruction reaches the execution stage of the instruction queue a taghit is generated by the CPU12X/XGATE. The state control register for the current state determines the next state for each trigger.

22.4.3.3 External Tagging Trigger

The $\overline{\text{TAGLO}}$ and $\overline{\text{TAGHI}}$ pins (mapped to device pins) can be used to tag an instruction. This function can be used as another breakpoint source. When the tagged opcode reaches the execution stage of the instruction queue a transition to the disarmed state0 occurs, ending the debug session and generating a breakpoint, if breakpoints are enabled. External tagging is only possible in device emulation modes.

22.4.3.4 Trigger On XGATE S/W Breakpoint Request

The XGATE S/W breakpoint request issues a forced breakpoint request to the CPU12X immediately and triggers the state sequencer into the disarmed state. Active tracing sessions are terminated immediately, thus if tracing has not yet begun, no trace information is stored. XGATE generated breakpoints are independent of the DBGBRK bits. The XGSBPE bit in DBGC1 determines if the XGATE S/W breakpoint function is enabled. The BDM bit in DBGC1 determines if the XGATE requested breakpoint causes the system to enter BDM Mode or initiate a software interrupt (SWI).

22.4.3.5 TRIG Immediate Trigger

Independent of comparator matches or external tag signals it is possible to initiate a tracing session and/or breakpoint by writing the TRIG bit in DBGC1 to a logic “1”. If configured for begin or mid aligned tracing, this triggers the state sequencer into the Final State, if configured for end alignment, setting the TRIG bit disarms the module, ending the session. If breakpoints are enabled, a forced breakpoint request is issued immediately (end alignment) or when tracing has completed (begin or mid alignment).

22.4.3.6 Trigger Priorities

In case of simultaneous triggers, the priority is resolved according to Table 22-42. The lower priority trigger is suppressed. It is thus possible to miss a lower priority trigger if it occurs simultaneously with a trigger of a higher priority. The trigger priorities described in Table 22-42 dictate that in the case of simultaneous matches, the match on the lower channel number (0,1,2,3) has priority. The SC[3:0] encoding ensures that a match leading to final state has priority over all other matches in each state sequencer state. When configured for range modes a simultaneous match of comparators A and C generates an active match0 whilst match2 is suppressed.

If a write access to DBGC1 with the ARM bit position set occurs simultaneously to a hardware disarm from an internal trigger event, then the ARM bit is cleared due to the hardware disarm.

Table 22-42. Trigger Priorities

Priority	Source	Action
Highest	**XGATE BKP**	Immediate forced breakpoint......(Tracing terminated immediately).
	TRIG	Trigger immediately to final state (begin or mid aligned tracing enabled) Trigger immediately to state 0 (end aligned or no tracing enabled)
	External TAGHI/TAGLO	Enter State0
	Match0 (force or tag hit)	Trigger to next state as defined by state control registers
	Match1 (force or tag hit)	Trigger to next state as defined by state control registers
	Match2 (force or tag hit)	Trigger to next state as defined by state control registers
Lowest	**Match3 (force or tag hit)**	Trigger to next state as defined by state control registers

22.4.4 State Sequence Control

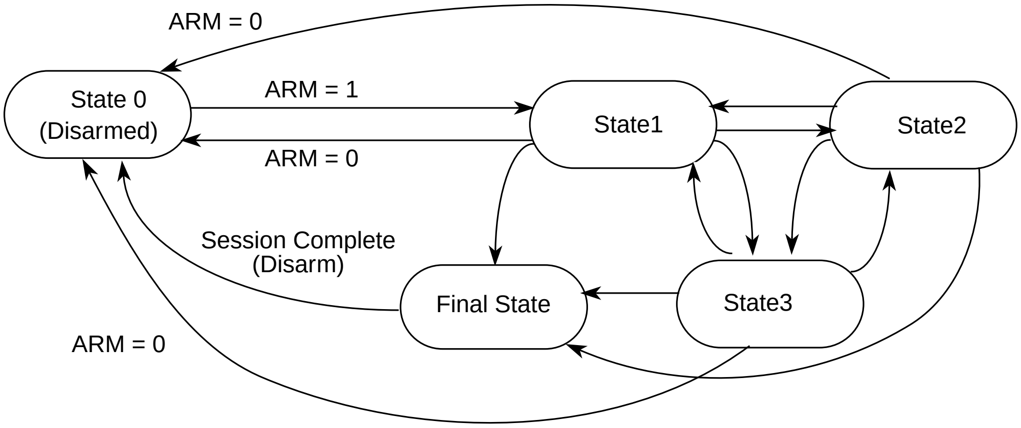


Figure 22-22. State Sequencer Diagram

The state sequencer allows a defined sequence of events to provide a trigger point for tracing of data in the trace buffer. Once the S12XDBG module has been armed by setting the ARM bit in the DBGC1 register, then state1 of the state sequencer is entered. Further transitions between the states are then controlled by the state control registers and depend upon a selected trigger mode condition being met. From Final State the only permitted transition is back to the disarmed state0. Transition between any of the states 1 to 3 is not restricted. Each transition updates the SSF[2:0] flags in DBGSR accordingly to indicate the current state.

Alternatively by setting the TRIG bit in DBGSC1, the state machine can be triggered to state0 or Final State depending on tracing alignment.

A tag hit through $\overline{\text{TAGHI}}/\overline{\text{TAGLO}}$ brings the state sequencer immediately into state0, causes a breakpoint, if breakpoints are enabled, and ends tracing immediately independent of the trigger alignment bits TALIGN[1:0].

Independent of the state sequencer, each comparator channel can be individually configured to generate an immediate breakpoint when a match occurs through the use of the BRK bits in the DBGxCTL registers. Thus it is possible to generate an immediate breakpoint on selected channels, whilst a state sequencer transition can be initiated by a match on other channels. If a debug session is ended by a trigger on a channel with BRK = 1, the state sequencer transitions through Final State for a clock cycle to state0. This is independent of tracing and breakpoint activity, thus with tracing and breakpoints disabled, the state sequencer enters state0 and the debug module is disarmed.

An XGATE S/W breakpoint request, if enabled causes a transition to the State0 and generates a breakpoint request to the CPU12X immediately

22.4.4.1 Final State

On entering Final State a trigger may be issued to the trace buffer according to the trace position control as defined by the TALIGN field (see Section 22.3.2.3”). If TSOURCE in the trace control register DBGTCR are cleared then the trace buffer is disabled and the transition to Final State can only generate a breakpoint request. In this case or upon completion of a tracing session when tracing is enabled, the ARM bit in the DBGC1 register is cleared, returning the module to the disarmed state0. If tracing is enabled, a breakpoint request can occur at the end of the tracing session. If neither tracing nor breakpoints are enabled then when the final state is reached it returns automatically to state0 and the debug module is disarmed.

22.4.5 Trace Buffer Operation

The trace buffer is a 64 lines deep by 64-bits wide RAM array. The S12XDBG module stores trace information in the RAM array in a circular buffer format. The RAM array can be accessed through a register window (DBGTBH:DBGTBL) using 16-bit wide word accesses. After each complete 64-bit trace buffer line is read, an internal pointer into the RAM is incremented so that the next read will receive fresh information. Data is stored in the format shown in Table 22-43. After each store the counter register bits DBGCNT[6:0] are incremented. Tracing of CPU12X activity is disabled when the BDM is active but tracing of XGATE activity is still possible. Reading the trace buffer whilst the DBG is armed returns invalid data and the trace buffer pointer is not incremented.

22.4.5.1 Trace Trigger Alignment

Using the TALIGN bits (see Section 22.3.2.3”) it is possible to align the trigger with the end, the middle, or the beginning of a tracing session.

If End or Mid tracing is selected, tracing begins when the ARM bit in DBGC1 is set and State1 is entered. The transition to Final State if End is selected signals the end of the tracing session. The transition to Final State if Mid is selected signals that another 32 lines will be traced before ending the tracing session. Tracing with Begin-Trigger starts at the opcode of the trigger.

22.4.5.1.1 Storing with Begin-Trigger

Storing with Begin-Trigger, data is not stored in the Trace Buffer until the Final State is entered. Once the trigger condition is met the S12XDBG module will remain armed until 64 lines are stored in the Trace Buffer. If the trigger is at the address of the change-of-flow instruction the change of flow associated with the trigger will be stored in the Trace Buffer. Using Begin-trigger together with tagging, if the tagged instruction is about to be executed then the trace is started. Upon completion of the tracing session the breakpoint is generated, thus the breakpoint does not occur at the tagged instruction boundary.

22.4.5.1.2 Storing with Mid-Trigger

Storing with Mid-Trigger, data is stored in the Trace Buffer as soon as the S12XDBG module is armed. When the trigger condition is met, another 32 lines will be traced before ending the tracing session, irrespective of the number of lines stored before the trigger occurred, then the S12XDBG module is disarmed and no more data is stored. Using Mid-trigger with tagging, if the tagged instruction is about to be executed then the trace is continued for another 32 lines. Upon tracing completion the breakpoint is generated, thus the breakpoint does not occur at the tagged instruction boundary.

22.4.5.1.3 Storing with End-Trigger

Storing with End-Trigger, data is stored in the Trace Buffer until the Final State is entered, at which point the S12XDBG module will become disarmed and no more data will be stored. If the trigger is at the address of a change of flow instruction the trigger event will not be stored in the Trace Buffer.

22.4.5.2 Trace Modes

The S12XDBG module can operate in four trace modes. The mode is selected using the TRCMOD bits in the DBGTCR register. In each mode tracing of XGATE or CPU12X information is possible. The source for the trace is selected using the TSOURCE bits in the DBGTCR register. The modes are described in the following subsections. The trace buffer organization is shown in Table 22-43.

22.4.5.2.1 Normal Mode

In Normal Mode, change of flow (COF) program counter (PC) addresses will be stored.

COF addresses are defined as follows for the CPU12X:

- Source address of taken conditional branches (long, short, bit-conditional, and loop primitives)
- Destination address of indexed JMP, JSR, and CALL instruction

- Destination address of RTI, RTS, and RTC instructions.
- Vector address of interrupts, except for SWI and BDM vectors

LBRA, BRA, BSR, BGND as well as non-indexed JMP, JSR, and CALL instructions are not classified as change of flow and are not stored in the trace buffer.

COF addresses are defined as follows for the XGATE:

- Source address of taken conditional branches
- Destination address of indexed JAL instructions.
- First XGATE code address in a thread

Change-of-flow addresses stored include the full 23-bit address bus of CPU12X, the 16-bit address bus for the XGATE module and an information byte, which contains a source/destination bit to indicate whether the stored address was a source address or destination address.

NOTE

When an CPU12X COF instruction with destination address is executed, the destination address is stored to the trace buffer on instruction completion, indicating the COF has taken place. If an interrupt occurs simultaneously then the next instruction carried out is actually from the interrupt service routine. The instruction at the destination address of the original program flow gets exectuted after the interrupt service routine.

In the following example an IRQ interrupt occurs during execution of the indexed JMP at address MARK1. The BRN at the destination (SUB_1) is not executed until after the IRQ service routine but the destination address is entered into the trace buffer to indicate that the indexed JMP COF has taken place.

```
        LDX     #SUB_1
MARK1   JMP     0,X                     ; IRQ interrupt occurs during execution of this
MARK2   NOP                             ;

SUB_1   BRN     *                       ; JMP Destination address TRACE BUFFER ENTRY 1
                                        ; RTI Destination address TRACE BUFFER ENTRY 3
        NOP                             ;
ADDR1   DBNE    A,PART5                 ; Source address TRACE BUFFER ENTRY 4

IRQ_ISR LDAB    #$F0                    ; IRQ Vector $FFF2 = TRACE BUFFER ENTRY 2
        STAB    VAR_C1
        RTI                             ;
```

The execution flow taking into account the IRQ is as follows

```
        LDX     #SUB_1
MARK1   JMP     0,X                     ;
IRQ_ISR LDAB    #$F0                    ;
        STAB    VAR_C1
        RTI                             ;
SUB_1   BRN     *
        NOP                             ;
ADDR1   DBNE    A,PART5                 ;
```

22.4.5.2.2 Loop1 Mode

Loop1 Mode, similarly to Normal Mode also stores only COF address information to the trace buffer, it however allows the filtering out of redundant information.

The intent of Loop1 Mode is to prevent the Trace Buffer from being filled entirely with duplicate information from a looping construct such as delays using the DBNE instruction or polling loops using BRSET/BRCLR instructions. Immediately after address information is placed in the Trace Buffer, the S12XDBG module writes this value into a background register. This prevents consecutive duplicate address entries in the Trace Buffer resulting from repeated branches.

Loop1 Mode only inhibits consecutive duplicate source address entries that would typically be stored in most tight looping constructs. It does not inhibit repeated entries of destination addresses or vector addresses, since repeated entries of these would most likely indicate a bug in the user's code that the S12XDBG module is designed to help find.

22.4.5.2.3 Detail Mode

In Detail Mode, address and data for all memory and register accesses is stored in the trace buffer. In the case of XGATE tracing this means that initialization of the R1 register during a vector fetch is not traced. This mode also features information byte entries to the trace buffer, for each address byte entry. The information byte indicates the size of access (word or byte) and the type of access (read or write).

When tracing CPU12X activity in Detail Mode, all cycles are traced except those when the CPU12X is either in a free or opcode fetch cycle. In this mode the XGATE program counter is also traced to provide a snapshot of the XGATE activity. CXINF information byte bits indicate the type of XGATE activity occurring at the time of the trace buffer entry. When tracing CPU12X activity alone in Detail Mode, the address range can be limited to a range specified by the TRANGE bits in DBGTCR. This function uses comparators C and D to define an address range inside which CPU12X activity should be traced (see Table 22-43). Thus the traced CPU12X activity can be restricted to particular register range accesses.

When tracing XGATE activity in Detail Mode, all load and store cycles are traced. Additionally the CPU12X program counter is stored at the time of the XGATE trace buffer entry to provide a snapshot of CPU12X activity.

22.4.5.2.4 Pure PC Mode

In Pure PC Mode, tracing from the CPU the PC addresses of all executed opcodes, including illegal opcodes, are stored. In Pure PC Mode, tracing from the XGATE the PC addresses of all executed opcodes are stored.

22.4.5.3 Trace Buffer Organization

Referring to Table 22-43. An X prefix denotes information from the XGATE module, a C prefix denotes information from the CPU12X. ADRH, ADRM, ADRL denote address high, middle and low byte respectively. INF bytes contain control information (R/W, S/D etc.). The numerical suffix indicates which tracing step. The information format for Loop1 Mode and PurePC Mode is the same as that of Normal Mode. Whilst tracing from XGATE or CPU12X only, in Normal or Loop1 modes each array line contains

2 data entries, thus in this case the DBGCNT[0] is incremented after each separate entry. In Detail mode DBGCNT[0] remains cleared whilst the other DBGCNT bits are incremented on each trace buffer entry.

XGATE and CPU12X COFs occur independently of each other and the profile of COFs for the two sources is totally different. When both sources are being traced in Normal or Loop1 mode, for each COF from one source, there may be many COFs from the other source, depending on user code. COF events could occur far from each other in the time domain, on consecutive cycles or simultaneously. When a COF occurs in either source (S12X or XGATE) a trace buffer entry is made and the corresponding CDV or XDV bit is set. The current PC of the other source is simultaneously stored to the trace buffer even if no COF has occurred, in which case CDV/XDV remains cleared indicating the address is not associated with a COF, but is simply a snapshot of the PC contents at the time of the COF from the other source.

Single byte data accesses in Detail Mode are always stored to the low byte of the trace buffer (CDATAL or XDATAL) and the high byte is cleared. When tracing word accesses, the byte at the lower address is always stored to trace buffer byte3 and the byte at the higher address is stored to byte2

Table 22-43. Trace Buffer Organization

Mode	**8-Byte Wide Word Buffer**							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
XGATE Detail	CXINF1	CADRH1	CADRM1	CADRL1	XDATAH1	XDATAL1	XADRM1	XADRL1
	CXINF2	CADRH2	CADRM2	CADRL2	XDATAH2	XDATAL2	XADRM2	XADRL2
CPU12X Detail	CXINF1	CADRH1	CADRM1	CADRL1	CDATAH1	CDATAL1	XADRM1	XADRL1
	CXINF2	CADRH2	CADRM2	CADRL2	CDATAH2	CDATAL2	XADRM2	XADRL2
Both Other Modes	XINF0		XPCM0	XPCL0	CINF0	CPCH0	CPCM0	CPCL0
	XINF1		XPCM1	XPCL1	CINF1	CPCH1	CPCM1	CPCL1
XGATE Other Modes	XINF1		XPCM1	XPCL1	XINF0		XPCM0	XPCL0
	XINF3		XPCM3	XPCL3	XINF2		XPCM2	XPCL2
CPU12X Other Modes	CINF1	CPCH1	CPCM1	CPCL1	CINF0	CPCH0	CPCM0	CPCL0
	CINF3	CPCH3	CPCM3	CPCL3	CINF2	CPCH2	CPCM2	CPCL2

22.4.5.3.1 Information Byte Organization

The format of the control information byte is dependent upon the active trace mode as described below. In Normal, Loop1, or Pure PC modes tracing of XGATE activity, XINF is used to store control information. In Normal, Loop1, or Pure PC modes tracing of CPU12X activity, CINF is used to store control information. In Detail Mode, CXINF contains the control information

XGATE Information Byte

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
XSD	XSOT	XCOT	XDV	0	0	0	0

Figure 22-23. XGATE Information Byte XINF

Table 22-44. XINF Field Descriptions

Field	Description
7 XSD	**Source Destination Indicator** — This bit indicates if the corresponding stored address is a source or destination address. This is only used in Normal and Loop1 mode tracing. 0 Source address 1 Destination address or Start of Thread or Continuation of Thread
6 XSOT	**Start Of Thread Indicator** — This bit indicates that the corresponding stored address is a start of thread address. This is only used in Normal and Loop1 mode tracing. **NOTE. This bit only has effect on devices where the XGATE module supports multiple interrupt levels**. 0 Stored address not from a start of thread 1 Stored address from a start of thread
5 XCOT	**Continuation Of Thread Indicator** — This bit indicates that the corresponding stored address is the first address following a return from a higher priority thread. This is only used in Normal and Loop1 mode tracing. **NOTE. This bit only has effect on devices where the XGATE module supports multiple interrupt levels.** 0 Stored address not from a continuation of thread 1 Stored address from a continuation of thread
4 XDV	**Data Invalid Indicator** — This bit indicates if the trace buffer entry is invalid. It is only used when tracing from both sources in Normal, Loop1 and Pure PC modes, to indicate that the XGATE trace buffer entry is valid. 0 Trace buffer entry is invalid 1 Trace buffer entry is valid

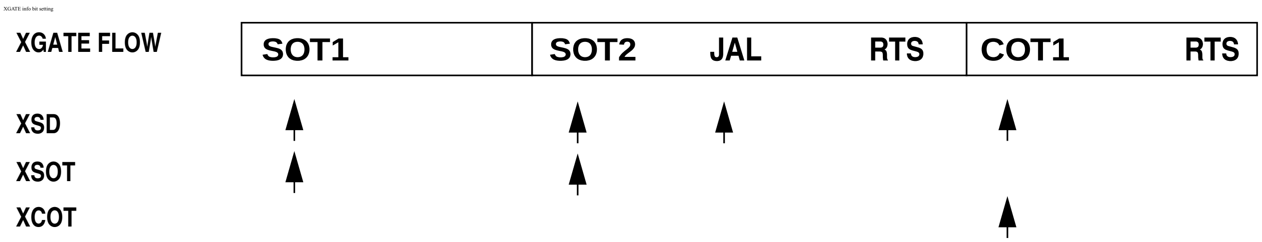


Figure 22-24. XGATE info bit setting

Figure 22-24 indicates the XGATE information bit setting when switching between threads, the initial thread starting at SOT1 and continuing at COT1 after the higher priority thread2 has ended.

CPU12X Information Byte

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CSD	CVA	0	CDV	0	0	0	0

Figure 22-25. CPU12X Information Byte CINF

Table 22-45. CINF Field Descriptions

Field	Description
7 CSD	**Source Destination Indicator** — This bit indicates if the corresponding stored address is a source or destination address. This is only used in Normal and Loop1 mode tracing. 0 Source address 1 Destination address
6 CVA	**Vector Indicator** — This bit indicates if the corresponding stored address is a vector address.. Vector addresses are destination addresses, thus if CVA is set, then the corresponding CSD is also set. This is only used in Normal and Loop1 mode tracing. This bit has no meaning in Pure PC mode. 0 Indexed jump destination address 1 Vector destination address
4 CDV	**Data Invalid Indicator** — This bit indicates if the trace buffer entry is invalid. It is only used when tracing from both sources in Normal, Loop1 and Pure PC modes, to indicate that the CPU12X trace buffer entry is valid. 0 Trace buffer entry is invalid 1 Trace buffer entry is valid

CXINF Information Byte

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CFREE	CSZ	CRW	COCF	XACK	XSZ	XRW	XOCF

Figure 22-26. Information Byte CXINF

This describes the format of the information byte used only when tracing in Detail Mode. When tracing from the CPU12X in Detail Mode, information is stored to the trace buffer on all cycles except opcode fetch and free cycles. The XGATE entry stored on the same line is a snapshot of the XGATE program counter. In this case the CSZ and CRW bits indicate the type of access being made by the CPU12X, whilst the XACK and XOCF bits indicate if the simultaneous XGATE cycle is a free cycle (no bus acknowledge) or opcode fetch cycle. Similarly when tracing from the XGATE in Detail Mode, information is stored to the trace buffer on all cycles except opcode fetch and free cycles. The CPU12X entry stored on the same line is a snapshot of the CPU12X program counter. In this case the XSZ and XRW bits indicate the type of access being made by the XGATE, whilst the CFREE and COCF bits indicate if the simultaneous CPU12X cycle is a free cycle or opcode fetch cycle.

Table 22-46. CXINF Field Descriptions

Field	Description
7 CFREE	**CPU12X Free Cycle Indicato**r — This bit indicates if the stored CPU12X address corresponds to a free cycle. This bit only contains valid information when tracing the XGATE accesses in Detail Mode. 0 Stored information corresponds to free cycle 1 Stored information does not correspond to free cycle

Table 22-46. CXINF Field Descriptions (continued)

Field	Description
6 CSZ	**Access Type Indicator** — This bit indicates if the access was a byte or word size access.This bit only contains valid information when tracing CPU12X activity in Detail Mode. 0 Word Access 1 Byte Access
5 CRW	**Read Write Indicator** — This bit indicates if the corresponding stored address corresponds to a read or write access. This bit only contains valid information when tracing CPU12X activity in Detail Mode. 0 Write Access 1 Read Access
4 COCF	**CPU12X Opcode Fetch Indicator** — This bit indicates if the stored address corresponds to an opcode fetch cycle. This bit only contains valid information when tracing the XGATE accesses in Detail Mode. 0 Stored information does not correspond to opcode fetch cycle 1 Stored information corresponds to opcode fetch cycle
3 XACK	**XGATE Access Indicator** — This bit indicates if the stored XGATE address corresponds to a free cycle. This bit only contains valid information when tracing the CPU12X accesses in Detail Mode. 0 Stored information corresponds to free cycle 1 Stored information does not correspond to free cycle
2 XSZ	**Access Type Indicator** — This bit indicates if the access was a byte or word size access. This bit only contains valid information when tracing XGATE activity in Detail Mode. 0 Word Access 1 Byte Access
1 XRW	**Read Write Indicator** — This bit indicates if the corresponding stored address corresponds to a read or write access. This bit only contains valid information when tracing XGATE activity in Detail Mode. 0 Write Access 1 Read Access
0 XOCF	**XGATE Opcode Fetch Indicator** — This bit indicates if the stored address corresponds to an opcode fetch cycle.This bit only contains valid information when tracing the CPU12X accesses in Detail Mode. 0 Stored information does not correspond to opcode fetch cycle 1 Stored information corresponds to opcode fetch cycle

22.4.5.4 Reading Data from Trace Buffer

The data stored in the Trace Buffer can be read using either the background debug module (BDM) module, the XGATE or the CPU12X provided the S12XDBG module is not armed, is configured for tracing and the system not secured. When the ARM bit is written to 1 the trace buffer is locked to prevent reading. The trace buffer can only be unlocked for reading by an aligned word write to DBGTB when the module is disarmed.

The Trace Buffer can only be read through the DBGTB register using aligned word reads, any byte or misaligned reads return 0 and do not cause the trace buffer pointer to increment to the next trace buffer address. The Trace Buffer data is read out first-in first-out. By reading CNT in DBGCNT the number of valid 64-bit lines can be determined. DBGCNT will not decrement as data is read.

Whilst reading an internal pointer is used to determine the next line to be read. After a tracing session, the pointer points to the oldest data entry, thus if no overflow has occurred, the pointer points to line0, otherwise it points to the line with the oldest entry. The pointer is initialized by each aligned write to DBGTBH to point to the oldest data again. This enables an interrupted trace buffer read sequence to be easily restarted from the oldest data entry.

The least significant word of each 64-bit wide array line is read out first. This corresponds to the bytes 1 and 0 of Table 22-43. The bytes containing invalid information (shaded in Table 22-43) are also read out.

Reading the Trace Buffer while the S12XDBG module is armed will return invalid data and no shifting of the RAM pointer will occur.

22.4.5.5 Trace Buffer Reset State

The Trace Buffer contents are not initialized by a system reset. Thus should a system reset occur, the trace session information from immediately before the reset occurred can be read out. The DBGCNT bits are not cleared by a system reset. Thus should a reset occur, the number of valid lines in the trace buffer is indicated by DBGCNT. The internal pointer to the current trace buffer address is initialized by unlocking the trace buffer thus points to the oldest valid data even if a reset occurred during the tracing session. Generally debugging occurrences of system resets is best handled using mid or end trigger alignment since the reset may occur before the trace trigger, which in the begin trigger alignment case means no information would be stored in the trace buffer.

NOTE

An external pin RESET that occurs simultaneous to a trace buffer entry can, in very seldom cases, lead to either that entry being corrupted or the first entry of the session being corrupted. In such cases the other contents of the trace buffer still contain valid tracing information. The case occurs when the reset assertion coincides with the trace buffer entry clock edge.

22.4.6 Tagging

A tag follows program information as it advances through the instruction queue. When a tagged instruction reaches the head of the queue a tag hit occurs and triggers the state sequencer.

Each comparator control register features a TAG bit, which controls whether the comparator match will cause a trigger immediately or tag the opcode at the matched address. If a comparator is enabled for tagged comparisons, the address stored in the comparator match address registers must be an opcode address for the trigger to occur.

Both CPU12X and XGATE opcodes can be tagged with the comparator register TAG bits.

Using Begin trigger together with tagging, if the tagged instruction is about to be executed then the transition to the next state sequencer state occurs. If the transition is to the Final State, tracing is started. Only upon completion of the tracing session can a breakpoint be generated. Similarly using Mid trigger with tagging, if the tagged instruction is about to be executed then the trace is continued for another 32 lines. Upon tracing completion the breakpoint is generated. Using End trigger, when the tagged instruction is about to be executed and the next transition is to Final State then a breakpoint is generated immediately, before the tagged instruction is carried out.

Read/Write (R/W), access size (SZ) monitoring and data bus monitoring is not useful if tagged triggering is selected, since the tag is attached to the opcode at the matched address and is not dependent on the data bus nor on the type of access. Thus these bits are ignored if tagged triggering is selected.

When configured for range comparisons and tagging, the ranges are accurate only to word boundaries.

S12X tagging is disabled when the BDM becomes active. XGATE tagging is possible when the BDM is active.

22.4.6.1 External Tagging using TAGHI and TAGLO

External tagging using the external TAGHI and TAGLO pins can only be used to tag CPU12X opcodes; tagging of XGATE code using these pins is not possible. An external tag triggers the state sequencer into state0 when the tagged opcode reaches the execution stage of the instruction queue.

The pins operate independently, thus the state of one pin does not affect the function of the other. External tagging is possible in emulation modes only. The presence of logic level 0 on either pin at the rising edge of the external clock (ECLK) performs the function indicated in the Table 22-47. It is possible to tag both bytes of an instruction word. If a taghit occurs, a breakpoint can be generated as defined by the DBGBRK and BDM bits in DBGC1. Each time TAGHI or TAGLO are low on the rising edge of ECLK, the old tag is replaced by a new one.

Table 22-47. Tag Pin Function

TAGHI	TAGLO	Tag
1	1	No tag
1	0	Low byte
0	1	High byte
0	0	Both bytes

22.4.6.2 Unconditional Tagging Function

In emulation modes a low assertion of PE5/TAGLO/MODA in the 7th or 8th bus cycle after reset enables the unconditional tagging function, allowing immediate tagging via TAGHI/TAGLO with breakpoint to BDM independent of the ARM, BDM and DBGBRK bits. Conversely these bits are not affected by unconditional tagging. The unconditional tagging function remains enabled until the next reset. This function allows an immediate entry to BDM in emulation modes before user code execution. The TAGLO assertion must be in the 7th or 8th bus cycle following the end of reset, whereby the prior RESET pin assertion lasts the full 192 bus cycles.

22.4.7 Breakpoints

Breakpoints can be generated as follows.

- Through XGATE software breakpoint requests.
- From comparator channel triggers to final state.
- Using software to write to the TRIG bit in the DBGC1 register.
- From taghits generated using the external TAGHI and TAGLO pins.

Breakpoints generated by the XGATE module or via the BDM BACKGROUND command have no affect on the CPU12X in STOP or WAIT mode.

22.4.7.1 XGATE Software Breakpoints

The XGATE software breakpoint instruction BRK can request a CPU12X breakpoint, via the S12XDBG module. In this case, if the XGSBPE bit is set, the S12XDBG module immediately generates a forced breakpoint request to the CPU12X, the state sequencer is returned to state0 and tracing, if active, is terminated. If configured for BEGIN trigger and tracing has not yet been triggered from another source, the trace buffer contains no information. Breakpoint requests from the XGATE module do not depend upon the state of the DBGBRK or ARM bits in DBGC1. They depend solely on the state of the XGSBPE and BDM bits. Thus it is not necessary to ARM the DBG module to use XGATE software breakpoints to generate breakpoints in the CPU12X program flow, but it is necessary to set XGSBPE. Furthermore, if a breakpoint to BDM is required, the BDM bit must also be set. When the XGATE requests an CPU12X breakpoint, the XGATE program flow stops by default, independent of the S12XDBG module.

22.4.7.2 Breakpoints From Internal Comparator Channel Final State Triggers

Breakpoints can be generated when internal comparator channels trigger the state sequencer to the Final State. If configured for tagging, then the breakpoint is generated when the tagged opcode reaches the execution stage of the instruction queue.

If a tracing session is selected by TSOURCE, breakpoints are requested when the tracing session has completed, thus if Begin or Mid aligned triggering is selected, the breakpoint is requested only on completion of the subsequent trace (see Table 22-48). If no tracing session is selected, breakpoints are requested immediately.

If the BRK bit is set on the triggering channel, then the breakpoint is generated immediately independent of tracing trigger alignment.

Table 22-48. Breakpoint Setup For Both XGATE and CPU12X Breakpoints

BRK	TALIGN	DBGBRK[n]	Breakpoint Alignment
0	00	0	Fill Trace Buffer until trigger (no breakpoints — keep running)
0	00	1	Fill Trace Buffer until trigger, then breakpoint request occurs
0	01	0	Start Trace Buffer at trigger (no breakpoints — keep running)
0	01	1	Start Trace Buffer at trigger A breakpoint request occurs when Trace Buffer is full
0	10	0	Store a further 32 Trace Buffer line entries after trigger (no breakpoints — keep running)
0	10	1	Store a further 32 Trace Buffer line entries after trigger Request breakpoint after the 32 further Trace Buffer entries
1	00,01,10	1	Terminate tracing and generate breakpoint immediately on trigger
1	00,01,10	0	Terminate tracing immediately on trigger
x	11	x	Reserved

22.4.7.3 Breakpoints Generated Via The TRIG Bit

If a TRIG triggers occur, the Final State is entered. If a tracing session is selected by TSOURCE, breakpoints are requested when the tracing session has completed, thus if Begin or Mid aligned triggering is selected, the breakpoint is requested only on completion of the subsequent trace (see Table 22-48). If no tracing session is selected, breakpoints are requested immediately. TRIG breakpoints are possible even if the S12XDBG module is disarmed.

22.4.7.4 Breakpoints Via TAGHI Or TAGLO Pin Taghits

Tagging using the external $\overline{\text{TAGHI}}$/$\overline{\text{TAGLO}}$ pins always ends the session immediately at the tag hit. It is always end aligned, independent of internal channel trigger alignment configuration.

22.4.7.5 S12XDBG Breakpoint Priorities

XGATE software breakpoints have the highest priority. Active tracing sessions are terminated immediately.

If a TRIG trigger occurs after Begin or Mid aligned tracing has already been triggered by a comparator instigated transition to Final State, then TRIG no longer has an effect. When the associated tracing session is complete, the breakpoint occurs. Similarly if a TRIG is followed by a subsequent trigger from a comparator channel, it has no effect, since tracing has already started.

If a comparator tag hit occurs simultaneously with an external $\overline{\text{TAGHI}}$/$\overline{\text{TAGLO}}$ hit, the state sequencer enters state0. TAGHI/TAGLO triggers are always end aligned, to end tracing immediately, independent of the tracing trigger alignment bits TALIGN[1:0].

22.4.7.5.1 S12XDBG Breakpoint Priorities And BDM Interfacing

Breakpoint operation is dependent on the state of the S12XBDM module. If the S12XBDM module is active, the CPU12X is executing out of BDM firmware and S12X breakpoints are disabled. In addition, while executing a BDM TRACE command, tagging into BDM is disabled. If BDM is not active, the breakpoint will give priority to BDM requests over SWI requests if the breakpoint coincides with a SWI instruction in the user's code. On returning from BDM, the SWI from user code gets executed.

Table 22-49. Breakpoint Mapping Summary

DBGBRK[1] (DBGC1[3])	BDM Bit (DBGC1[4])	BDM Enabled	BDM Active	S12X Breakpoint Mapping
0	X	X	X	No Breakpoint
1	0	X	0	Breakpoint to SWI
1	0	X	1	No Breakpoint
1	1	0	X	Breakpoint to SWI
1	1	1	0	Breakpoint to BDM
1	1	1	1	No Breakpoint

BDM cannot be entered from a breakpoint unless the ENABLE bit is set in the BDM. If entry to BDM via a BGND instruction is attempted and the ENABLE bit in the BDM is cleared, the CPU12X actually executes the BDM firmware code. It checks the ENABLE and returns if ENABLE is not set. If not serviced by the monitor then the breakpoint is re-asserted when the BDM returns to normal CPU12X flow.

If the comparator register contents coincide with the SWI/BDM vector address then an SWI in user code and DBG breakpoint could occur simultaneously. The CPU12X ensures that BDM requests have a higher priority than SWI requests. Returning from the BDM/SWI service routine care must be taken to avoid re triggering a breakpoint.

NOTE

When program control returns from a tagged breakpoint using an RTI or BDM GO command without program counter modification it will return to the instruction whose tag generated the breakpoint. To avoid re triggering a breakpoint at the same location reconfigure the S12XDBG module in the SWI routine, if configured for an SWI breakpoint, or over the BDM interface by executing a TRACE command before the GO to increment the program flow past the tagged instruction.

An XGATE software breakpoint is forced immediately, the tracing session terminated and the XGATE module execution stops. The user can thus determine if an XGATE breakpoint has occurred by reading out the XGATE program counter over the BDM interface.

Chapter 23
External Bus Interface (S12XEBIV3)

23.1 Introduction

This document describes the functionality of the XEBI block controlling the external bus interface.

The XEBI controls the functionality of a non-multiplexed external bus (a.k.a. ‘expansion bus’) in relationship with the chip operation modes. Dependent on the mode, the external bus can be used for data exchange with external memory, peripherals or PRU, and provide visibility to the internal bus externally in combination with an emulator.

23.1.1 Glossary or Terms

bus clock	System Clock. Refer to CRG Block Guide.
expanded modes	Normal Expanded Mode Emulation Single-Chip Mode Emulation Expanded Mode Special Test Mode
single-chip modes	Normal Single-Chip Mode Special Single-Chip Mode
emulation modes	Emulation Single-Chip Mode Emulation Expanded Mode
normal modes	Normal Single-Chip Mode Normal Expanded Mode
special modes	Special Single-Chip Mode Special Test Mode
NS	Normal Single-Chip Mode
SS	Special Single-Chip Mode
NX	Normal Expanded Mode
ES	Emulation Single-Chip Mode
EX	Emulation Expanded Mode
ST	Special Test Mode
external resource	Addresses outside MCU
PRR	Port Replacement Registers
PRU	Port Replacement Unit
EMULMEM	External emulation memory
access source	CPU or BDM or XGATE

23.1.2 Features

The XEBI includes the following features:

- Output of up to 23-bit address bus and control signals to be used with a non-muxed external bus
- Bidirectional 16-bit external data bus with option to disable upper half
- Visibility of internal bus activity

23.1.3 Modes of Operation

- Single-chip modes

  The external bus interface is not available in these modes.
- Expanded modes

  Address, data, and control signals are activated on the external bus in normal expanded mode and special test mode.
- Emulation modes

  The external bus is activated to interface to an external tool for emulation of normal expanded mode or normal single-chip mode applications.

Refer to the S12X_MMC section for a detailed description of the MCU operating modes.

23.1.4 Block Diagram

Figure 23-1 is a block diagram of the XEBI with all related I/O signals.

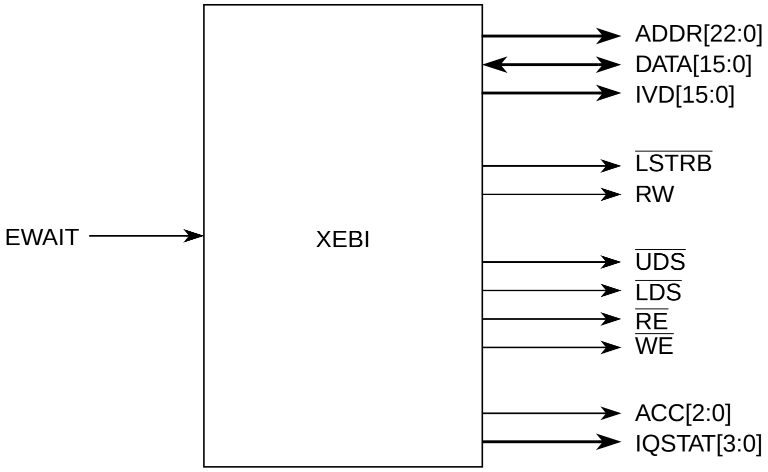


Figure 23-1. XEBI Block Diagram

23.2 External Signal Description

The user is advised to refer to the SoC section for port configuration and location of external bus signals.

NOTE

The following external bus related signals are described in other sections:
$\overline{\text{CS2}}$, $\overline{\text{CS1}}$, $\overline{\text{CS0}}$ (chip selects) — S12X_MMC section
ECLK, ECLKX2 (free-running clocks) — PIM section
$\overline{\text{TAGHI}}$, $\overline{\text{TAGLO}}$ (tag inputs) — PIM section, S12X_DBG section

Table 23-1 outlines the pin names and gives a brief description of their function. Refer to the SoC section and PIM section for reset states of these pins and associated pull-ups or pull-downs.

Table 23-1. External System Signals Associated with XEBI

Signal	I[1]/O	EBI Signal Multiplex (T)ime[2] (F)unction[3]		Description	Available in Modes					
					NS	SS	NX	ES	EX	ST
$\overline{\text{RE}}$	O	—	—	Read Enable, indicates external read access	No	No	Yes	No	No	No
ADDR[22:20]	O	T	—	External address	No	No	Yes	Yes	Yes	Yes
ACC[2:0]	O		—	Access source	No	No	No	Yes	Yes	Yes
ADDR[19:16]	O	T	—	External address	No	No	Yes	Yes	Yes	Yes
IQSTAT[3:0]	O		—	Instruction Queue Status	No	No	No	Yes	Yes	Yes
ADDR[15:1]	O	T	—	External address	No	No	Yes	Yes	Yes	Yes
IVD[15:1]	O		—	Internal visibility read data (IVIS = 1)	No	No	No	Yes	Yes	Yes
ADDR0	O	T	F	External address	No	No	No	Yes	Yes	Yes
IVD0	O			Internal visibility read data (IVIS = 1)	No	No	No	Yes	Yes	Yes
$\overline{\text{UDS}}$	O	—		Upper Data Select, indicates external access to the high byte DATA[15:8]	No	No	Yes	No	No	No
$\overline{\text{LSTRB}}$	O	—	F	Low Strobe, indicates valid data on DATA[7:0]	No	No	No	Yes	Yes	Yes
$\overline{\text{LDS}}$	O	—		Lower Data Select, indicates external access to the low byte DATA[7:0]	No	No	Yes	No	No	No
R$\overline{\text{W}}$	O	—	F	Read/Write, indicates the direction of internal data transfers	No	No	No	Yes	Yes	Yes
$\overline{\text{WE}}$	O	—		Write Enable, indicates external write access	No	No	Yes	No	No	No
DATA[15:8]	I/O	—	—	Bidirectional data (even address)	No	No	Yes	Yes	Yes	Yes
DATA[7:0]	I/O	—	—	Bidirectional data (odd address)	No	No	Yes	Yes	Yes	Yes
$\overline{\text{EWAIT}}$	I	—	—	External control for external bus access stretches (adding wait states)	No	No	Yes	No	Yes	No

[1] All inputs are capable of reducing input threshold level

[2] Time-multiplex means that the respective signals share the same pin on chip level and are active alternating in a dedicated time slot (in modes where applicable).

[3] Function-multiplex means that one of the respective signals sharing the same pin on chip level continuously uses the pin depending on configuration and reset state.

23.3 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in the XEBI.

23.3.1 Module Memory Map

The registers associated with the XEBI block are shown in Figure 23-2.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0E EBICTL0	R	ITHRS	0	HDBE	ASIZ4	ASIZ3	ASIZ2	ASIZ1	ASIZ0
	W		▨						
0x0F EBICTL1	R	EWAITE	0	0	0	0	EXSTR2	EXSTR1	EXSTR0
	W		▨	▨	▨	▨			

▨ = Unimplemented or Reserved

Figure 23-2. XEBI Register Summary

23.3.2 Register Descriptions

The following sub-sections provide a detailed description of each register and the individual register bits.

All control bits can be written anytime, but this may have no effect on the related function in certain operating modes. This allows specific configurations to be set up before changing into the target operating mode.

NOTE

Depending on the operating mode an available function may be enabled, disabled or depend on the control register bit. Reading the register bits will reflect the status of related function only if the current operating mode allows user control. Please refer the individual bit descriptions.

23.3.2.1 External Bus Interface Control Register 0 (EBICTL0)

Module Base +0x000E (PRR)

	7	6	5	4	3	2	1	0
R	ITHRS	0	HDBE	ASIZ4	ASIZ3	ASIZ2	ASIZ1	ASIZ0
W		▨						
Reset	0	0	1	1	1	1	1	1

▨ = Unimplemented or Reserved

Figure 23-3. External Bus Interface Control Register 0 (EBICTL0)

Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes, the data is read from this register.

Write: Anytime. In emulation modes, write operations will also be directed to the external bus.

This register controls input pin threshold level and determines the external address and data bus sizes in normal expanded mode. If not in use with the external bus interface, the related pins can be used for alternative functions.

External bus is available as programmed in normal expanded mode and always full-sized in emulation modes and special test mode; function not available in single-chip modes.

Table 23-2. EBICTL0 Field Descriptions

Field	Description
7 ITHRS	**Reduced Input Threshold** — This bit selects reduced input threshold on external data bus pins and specific control input signals which are in use with the external bus interface in order to adapt to external devices with a 3.3 V, 5 V tolerant I/O. The reduced input threshold level takes effect depending on ITHRS, the operating mode and the related enable signals of the EBI pin function as summarized in Table 23-3. 0 Input threshold is at standard level on all pins 1 Reduced input threshold level enabled on pins in use with the external bus interface
5 HDBE	**High Data Byte Enable** — This bit enables the higher half of the 16-bit data bus. If disabled, only the lower 8-bit data bus can be used with the external bus interface. In this case the unused data pins and the data select signals ($\overline{\text{UDS}}$ and $\overline{\text{LDS}}$) are free to be used for alternative functions. 0 DATA[15:8], $\overline{\text{UDS}}$, and $\overline{\text{LDS}}$ disabled 1 DATA[15:8], $\overline{\text{UDS}}$, and $\overline{\text{LDS}}$ enabled
4–0 ASIZ[4:0]	**External Address Bus Size** — These bits allow scalability of the external address bus. The programmed value corresponds to the number of available low-aligned address lines (refer to Table 23-4). All address lines ADDR[22:0] start up as outputs after reset in expanded modes. This needs to be taken into consideration when using alternative functions on relevant pins in applications which utilize a reduced external address bus.

Table 23-3. Input Threshold Levels on External Signals

ITHRS	External Signal	NS	SS	NX	ES	EX	ST
0	DATA[15:8] $\overline{\text{TAGHI}}$, $\overline{\text{TAGLO}}$	Standard	Standard	Standard	Reduced	Reduced	Standard
	DATA[7:0]						
	EWAIT				Standard	Standard	
1	DATA[15:8] $\overline{\text{TAGHI}}$, $\overline{\text{TAGLO}}$	Standard	Standard	Reduced if HDBE = 1	Reduced	Reduced	Reduced
	DATA[7:0]			Reduced			
	$\overline{\text{EWAIT}}$			Reduced if EWAITE = 1	Standard	Reduced if EWAITE = 1	Standard

Table 23-4. External Address Bus Size

ASIZ[4:0]	Available External Address Lines
00000	None
00001	$\overline{\text{UDS}}$
00010	ADDR1, $\overline{\text{UDS}}$
00011	ADDR[2:1], $\overline{\text{UDS}}$
:	:
10110	ADDR[21:1], $\overline{\text{UDS}}$
10111 : 11111	ADDR[22:1], $\overline{\text{UDS}}$

23.3.2.2 External Bus Interface Control Register 1 (EBICTL1)

Module Base +0x000F (PRR)

	7	6	5	4	3	2	1	0
R	EWAITE	0	0	0	0	EXSTR2	EXSTR1	EXSTR0
W		▨	▨	▨	▨			
Reset	0	0	0	0	0	1	1	1

▨ = Unimplemented or Reserved

Figure 23-4. External Bus Interface Control Register 1 (EBICTL1)

Read: Anytime. In emulation modes, read operations will return the data from the external bus, in all other modes the data is read from this register.

Write: Anytime. In emulation modes, write operations will also be directed to the external bus.

This register is used to configure the external access stretch (wait) function.

Table 23-5. EBICTL1 Field Descriptions

Field	Description
7 EWAITE	**External Wait Enable** — This bit enables the external access stretch function using the external $\overline{\text{EWAIT}}$ input pin. Enabling this feature may have effect on the minimum number of additional stretch cycles (refer to Table 23-6). External wait feature is only active if enabled in normal expanded mode and emulation expanded mode; function not available in all other operating modes. 0 External wait is disabled 1 External wait is enabled
2–0 EXSTR[2:0]	**External Access Stretch Bits 2, 1, 0** — This three bit field determines the amount of additional clock stretch cycles on every access to the external address space as shown in Table 23-6. The minimum number of stretch cycles depends on the EWAITE setting. Stretch cycles are added as programmed in normal expanded mode and emulation expanded mode; function not available in all other operating modes.

Table 23-6. External Access Stretch Bit Definition

EXSTR[2:0]	Number of Stretch Cycles	
	EWAITE = 0	EWAITE = 1
000	1 cycle	>= 2 cycles
001	2 cycles	>= 2 cycles
010	3 cycles	>= 3 cycles
011	4 cycles	>= 4 cycles
100	5 cycles	>= 5 cycles
101	6 cycles	>= 6 cycles
110	7 cycles	>= 7 cycles
111	8 cycles	>= 8 cycles

23.4 Functional Description

This section describes the functions of the external bus interface. The availability of external signals and functions in relation to the operating mode is initially summarized and described in more detail in separate sub-sections.

23.4.1 Operating Modes and External Bus Properties

A summary of the external bus interface functions for each operating mode is shown in Table 23-7.

Table 23-7. Summary of Functions

Properties (if Enabled)	**Single-Chip Modes**		**Expanded Modes**			
	Normal Single-Chip	**Special Single-Chip**	**Normal Expanded**	**Emulation Single-Chip**	**Emulation Expanded**	**Special Test**
Timing Properties						
PRR access[1]	2 cycles read internal write internal	2 cycles read internal write internal	2 cycles read internal write internal	2 cycles read external write int & ext	2 cycles read external write int & ext	2 cycles read internal write internal
Internal access visible externally	—	—	—	1 cycle	1 cycle	1 cycle
External address access and unimplemented area access[2]	—	—	Max. of 2 to 9 programmed cycles or n cycles of ext. wait[3]	1 cycle	Max. of 2 to 9 programmed cycles or n cycles of ext. wait[3]	1 cycle
Flash area address access[4]	—	—	—	1 cycle	1 cycle	1 cycle
Signal Properties						
Bus signals	—	—	ADDR[22:1] DATA[15:0]	ADDR[22:20]/ACC[2:0] ADDR[19:16]/IQSTAT[3:0] ADDR[15:0]/IVD[15:0] DATA[15:0]	ADDR[22:20]/ACC[2:0] ADDR[19:16]/IQSTAT[3:0] ADDR[15:0]/IVD[15:0] DATA[15:0]	ADDR[22:0] DATA[15:0]
Data select signals (if 16-bit data bus)	—	—	$\overline{\text{UDS}}$ $\overline{\text{LDS}}$	ADDR0 $\overline{\text{LSTRB}}$	ADDR0 $\overline{\text{LSTRB}}$	ADDR0 $\overline{\text{LSTRB}}$
Data direction signals	—	—	$\overline{\text{RE}}$ $\overline{\text{WE}}$	R$\overline{\text{W}}$	R$\overline{\text{W}}$	R$\overline{\text{W}}$
Chip Selects	—	—	$\overline{\text{CS0}}$ $\overline{\text{CS1}}$ $\overline{\text{CS2}}$ $\overline{\text{CS3}}$	—	$\overline{\text{CS0}}$ $\overline{\text{CS1}}$ $\overline{\text{CS2}}$ $\overline{\text{CS3}}$	$\overline{\text{CS0}}$ $\overline{\text{CS1}}$ $\overline{\text{CS2}}$ $\overline{\text{CS3}}$
External wait feature	—	—	$\overline{\text{EWAIT}}$	—	$\overline{\text{EWAIT}}$	—
Reduced input threshold enabled on	—	—	Refer to Table 23-3	DATA[15:0] $\overline{\text{EWAIT}}$	DATA[15:0] $\overline{\text{EWAIT}}$	Refer to Table 23-3

[1] Incl. S12X_EBI registers

[2] Refer to S12X_MMC section.

[3] If EWAITE = 1, the minimum number of external bus cycles is 3.

[4] Available only if configured appropriately by ROMON and EROMON (refer to S12X_MMC section).

23.4.2 Internal Visibility

Internal visibility allows the observation of the internal CPU address and data bus as well as the determination of the access source and the CPU pipe (queue) status through the external bus interface.

Internal visibility is always enabled in emulation single chip mode and emulation expanded mode. Internal CPU accesses are made visible on the external bus interface except CPU execution of BDM firmware instructions.

Internal reads are made visible on ADDRx/IVDx (address and read data multiplexed, see Table 23-10 to Table 23-12), internal writes on ADDRx and DATAx (see Table 23-13 to Table 23-15). R$\overline{\text{W}}$ and $\overline{\text{LSTRB}}$ show the type of access. External read data are also visible on IVDx.

During ‘no access’ cycles R$\overline{\text{W}}$ is held in read position while $\overline{\text{LSTRB}}$ is undetermined.

All accesses which make use of the external bus interface are considered external accesses.

23.4.2.1 Access Source Signals (ACC)

The access source can be determined from the external bus control signals ACC[2:0] as shown in Table 23-8.

Table 23-8. Determining Access Source from Control Signals

ACC[2:0]	Access Description
000	Repetition of previous access cycle
001	CPU access
010	BDM external access
011	XGATE PRR access
100	No access[1]
101	CPU access error
110, 111	Reserved

[1] Denotes also CPU accesses to BDM firmware and BDM registers (IQSTATx are ‘XXXX’ and R$\overline{\text{W}}$ = 1 in these cases)

23.4.2.2 Instruction Queue Status Signals (IQSTAT)

The CPU instruction queue status (execution-start and data-movement information) is brought out as IQSTAT[3:0] signals. For decoding of the IQSTAT values, refer to the S12X_CPU section.

23.4.2.3 Internal Visibility Data (IVD)

Depending on the access size and alignment, either a word of read data is made visible on the address lines or only the related data byte will be presented in the ECLK low phase. For details refer to Table 23-9.

Invalid IVD are brought out in case of non-CPU read accesses.

Table 23-9. IVD Read Data Output

Access	IVD[15:8]	IVD[7:0]
Word read of data at an even and even+1 address	ivd(even)	ivd(even+1)
Word read of data at an odd and odd+1 internal RAM address (misaligned)	ivd(odd+1)	ivd(odd)
Byte read of data at an even address	ivd(even)	addr[7:0] (rep.)
Byte read of data at an odd address	addr[15:8] (rep.)	ivd(odd)

23.4.2.4 Emulation Modes Timing

A bus access lasts 1 ECLK cycle. In case of a stretched external access (emulation expanded mode), up to an infinite amount of ECLK cycles may be added. ADDRx values will only be shown in ECLK high phases, while ACCx, IQSTATx, and IVDx values will only be presented in ECLK low phases.

Based on this multiplex timing, ACCx are only shown in the current (first) access cycle. IQSTATx and (for read accesses) IVDx follow in the next cycle. If the access takes more than one bus cycle, ACCx display NULL (0x000) in the second and all following cycles of the access. IQSTATx display NULL (0x0000) from the third until one cycle after the access to indicate continuation.

The resulting timing pattern of the external bus signals is outlined in the following tables for read, write and interleaved read/write accesses. Three examples represent different access lengths of 1, 2, and n–1 bus cycles. Non-shaded bold entries denote all values related to Access #0.

The following terminology is used:

‘addr’ — value(ADDRx); small letters denote the logic values at the respective pins
‘x’ — Undefined output pin values
‘z’ — Tristate pins
‘?’ — Dependent on previous access (read or write); IVDx: ‘ivd’ or ‘x’; DATAx: ‘data’ or ‘z’

23.4.2.4.1 Read Access Timing

Table 23-10. Read Access (1 Cycle)

		Access #0				**Access #1**		
Bus cycle ->	...	**1**		**2**		**3**		...
ECLK phase	...	high	low	high	low	high	low	...
ADDR[22:20] / ACC[2:0]	...	**addr 0**	**acc 0**	addr 1	acc 1	addr 2	acc 2	...
ADDR[19:16] / IQSTAT[3:0]	...		iqstat -1		**iqstat 0**		iqstat 1	...
ADDR[15:0] / IVD[15:0]	...		?		**ivd 0**		ivd 1	...
DATA[15:0] (internal read)	...	?	**z**	**z**	z	z	z	...
DATA[15:0] (external read)	...	?	**z**	**data 0**	z	data 1	z	...
R$\overline{W}$	...	1	1	1	1	1	1	...

Table 23-11. Read Access (2 Cycles)

		Access #0				**Access #1**		
Bus cycle ->	...	**1**		**2**		**3**		...
ECLK phase	...	high	low	high	low	high	low	...
ADDR[22:20] / ACC[2:0]	...	**addr 0**	**acc 0**	**addr 0**	**000**	addr 1	acc 1	...
ADDR[19:16] / IQSTAT[3:0]	...		iqstat-1		**iqstat 0**		**0000**	...
ADDR[15:0] / IVD[15:0]	...		?		**x**		**ivd 0**	...
DATA[15:0] (internal read)	...	?	**z**	**z**	**z**	**z**	z	...
DATA[15:0] (external read)	...	?	**z**	**z**	**z**	**data 0**	z	...
R$\overline{W}$	...	1	1	1	1	1	1	...

Table 23-12. Read Access (n–1 Cycles)

		Access #0							**Access #1**		
Bus cycle ->	...	**1**		**2**		**3**		...	**n**		...
ECLK phase	...	high	low	high	low	high	low	...	high	low	...
ADDR[22:20] / ACC[2:0]	...	**addr 0**	**acc 0**	**addr 0**	**000**	**addr 0**	**000**	...	addr 1	acc 1	...
ADDR[19:16] / IQSTAT[3:0]	...		iqstat-1		**iqstat 0**		**0000**	...		**0000**	...
ADDR[15:0] / IVD[15:0]	...		?		**x**		**x**	...		**ivd 0**	...
DATA[15:0] (internal read)	...	?	**z**	**z**	**z**	**z**	**z**	...	**z**	z	...
DATA[15:0] (external read)	...	?	**z**	**z**	**z**	**z**	**z**	...	**data 0**	z	...
R$\overline{W}$	...	1	1	1	1	1	1	...	1	1	...

23.4.2.4.2 Write Access Timing

Table 23-13. Write Access (1 Cycle)

		Access #0		**Access #1**		**Access #2**		
Bus cycle ->	...	**1**		**2**		**3**		...
ECLK phase	...	high	low	high	low	high	low	...
ADDR[22:20] / ACC[2:0]	...	**addr 0**	**acc 0**	addr 1	acc 1	addr 2	acc 2	...
ADDR[19:16] / IQSTAT[3:0]	...		iqstat -1		**iqstat 0**		iqstat 1	...
ADDR[15:0] / IVD[15:0]	...		?		**x**		x	...
DATA[15:0] (write)	...	?	**data 0**		data 1		data 2	...
R$\overline{W}$	...	0	0	1	1	1	1	...

Table 23-14. Write Access (2 Cycles)

		Access #0				**Access #1**		
Bus cycle ->	...	**1**		**2**		**3**		...
ECLK phase	...	high	low	high	low	high	low	...
ADDR[22:20] / ACC[2:0]	...	**addr 0**	**acc 0**	**addr 0**	**000**	addr 1	acc 1	...
ADDR[19:16] / IQSTAT[3:0]	...		iqstat-1		**iqstat 0**		**0000**	...
ADDR[15:0] / IVD[15:0]	...		?		**x**		**x**	...
DATA[15:0] (write)	...	?	**data 0**				x	...
R$\overline{W}$	...	0	0	0	0	1	1	...

Table 23-15. Write Access (n–1 Cycles)

		Access #0							**Access #1**		
Bus cycle ->	...	**1**		**2**		**3**		...	**n**		...
ECLK phase	...	high	low	high	low	high	low	...	high	low	...
ADDR[22:20] / ACC[2:0]	...	**addr 0**	**acc 0**	**addr 0**	**000**	**addr 0**	**000**	**...**	addr 1	acc 1	...
ADDR[19:16] / IQSTAT[3:0]	...		iqstat-1		**iqstat 0**		**0000**	**...**		**0000**	...
ADDR[15:0] / IVD[15:0]	...		?		**x**		**x**	**...**		**x**	...
DATA[15:0] (write)	...	?	**data 0**							x	...
R$\overline{W}$	...	0	0	0	0	0	0	...	1	1	...

23.4.2.4.3 Read-Write-Read Access Timing

Table 23-16. Interleaved Read-Write-Read Accesses (1 Cycle)

		Access #0		**Access #1**		**Access #2**		
Bus cycle ->	...	**1**		**2**		**3**		...
ECLK phase	...	high	low	high	low	high	low	...
ADDR[22:20] / ACC[2:0]	...	**addr 0**	**acc 0**	addr 1	acc 1	addr 2	acc 2	...
ADDR[19:16] / IQSTAT[3:0]	...		iqstat -1		**iqstat 0**		iqstat 1	...
ADDR[15:0] / IVD[15:0]	...		?		**ivd 0**		x	...
DATA[15:0] (internal read)	...	?	**z**	**z**	(write) data 1		z	...
DATA[15:0] (external read)	...	?	**z**	**data 0**	(write) data 1		z	...
R$\overline{W}$	...	1	1	0	0	1	1	...

23.4.3 Accesses to Port Replacement Registers

All read and write accesses to PRR addresses take two bus clock cycles independent of the operating mode. If writing to these addresses in emulation modes, the access is directed to both, the internal register and the external resource while reads will be treated external.

The XEBI control registers also belong to this category.

23.4.4 Stretched External Bus Accesses

In order to allow fast internal bus cycles to coexist in a system with slower external resources, the XEBI supports stretched external bus accesses (wait states).

This feature is available in normal expanded mode and emulation expanded mode for accesses to all external addresses except emulation memory and PRR. In these cases the fixed access times are 1 or 2 cycles, respectively.

Stretched accesses are controlled by:

1. EXSTR[2:0] bits in the EBICTL1 register configuring fixed amount of stretch cycles
2. Activation of the external wait feature by EWAITE in EBICTL1 register
3. Assertion of the external $\overline{\text{EWAIT}}$ signal when EWAITE = 1

The EXSTR[2:0] control bits can be programmed for generation of a fixed number of 1 to 8 stretch cycles. If the external wait feature is enabled, the minimum number of additional stretch cycles is 2. An arbitrary amount of stretch cycles can be added using the $\overline{\text{EWAIT}}$ input.

$\overline{\text{EWAIT}}$ needs to be asserted at least for a minimal specified time window within an external access cycle for the internal logic to detect it and add a cycle (refer to electrical characteristics). Holding it for additional cycles will cause the external bus access to be stretched accordingly.

Write accesses are stretched by holding the initiator in its current state for additional cycles as programmed and controlled by external wait after the data have been driven out on the external bus. This results in an extension of time the bus signals and the related control signals are valid externally.

Read data are not captured by the system in normal expanded mode until the specified setup time before the $\overline{\text{RE}}$ rising edge.

Read data are not captured in emulation expanded mode until the specified setup time before the falling edge of ECLK.

In emulation expanded mode, accesses to the internal flash or the emulation memory (determined by EROMON and ROMON bits; see S12X_MMC section for details) always take 1 cycle and stretching is not supported. In case the internal flash is taken out of the map in user applications, accesses are stretched as programmed and controlled by external wait.

23.4.5 Data Select and Data Direction Signals

The S12X_EBI supports byte and word accesses at any valid external address. The big endian system of the MCU is extended to the external bus; however, word accesses are restricted to even aligned addresses. The only exception is the visibility of misaligned word accesses to addresses in the internal RAM as this module exclusively supports these kind of accesses in a single cycle.

With the above restriction, a fixed relationship is implied between the address parity and the dedicated bus halves where the data are accessed: DATA[15:8] is related to even addresses and DATA[7:0] is related to odd addresses.

In expanded modes the data access type is externally determined by a set of control signals, i.e., data select and data direction signals, as described below. The data select signals are not available if using the external bus interface with an 8-bit data bus.

23.4.5.1 Normal Expanded Mode

In normal expanded mode, the external signals $\overline{\text{RE}}$, $\overline{\text{WE}}$, $\overline{\text{UDS}}$, $\overline{\text{LDS}}$ indicate the access type (read/write), data size and alignment of an external bus access (Table 23-17).

Table 23-17. Access in Normal Expanded Mode

Access	$\overline{\text{RE}}$	$\overline{\text{WE}}$	$\overline{\text{UDS}}$	$\overline{\text{LDS}}$	DATA[15:8]		DATA[7:0]	
					I/O	data(addr)	I/O	data(addr)
Word write of data on DATA[15:0] at an even and even+1 address	1	0	0	0	Out	data(even)	Out	data(odd)
Byte write of data on DATA[7:0] at an odd address	1	0	1	0	In	x	Out	data(odd)
Byte write of data on DATA[15:8] at an even address	1	0	0	1	Out	data(even)	In	x
Word read of data on DATA[15:0] at an even and even+1 address	0	1	0	0	In	data(even)	In	data(odd)
Byte read of data on DATA[7:0] at an odd address	0	1	1	0	In	x	In	data(odd)
Byte read of data on DATA[15:8] at an even address	0	1	0	1	In	data(even)	In	x
Indicates No Access	1	1	1	1	In	x	In	x
Unimplemented	1	1	1	0	In	x	In	x
	1	1	0	1	In	x	In	x

23.4.5.2 Emulation Modes and Special Test Mode

In emulation modes and special test mode, the external signals $\overline{\text{LSTRB}}$, R$\overline{\text{W}}$, and ADDR0 indicate the access type (read/write), data size and alignment of an external bus access. Misaligned accesses to the internal RAM and misaligned XGATE PRR accesses in emulation modes are the only type of access that are able to produce $\overline{\text{LSTRB}}$ = ADDR0 = 1. This is summarized in Table 23-18.

Table 23-18. Access in Emulation Modes and Special Test Mode

Access	R$\overline{\text{W}}$	$\overline{\text{LSTRB}}$	ADDR0	DATA[15:8]		DATA[7:0]	
				I/O	data(addr)	I/O	data(addr)
Word write of data on DATA[15:0] at an even and even+1 address	0	0	0	Out	data(even)	Out	data(odd)
Byte write of data on DATA[7:0] at an odd address	0	0	1	In	x	Out	data(odd)
Byte write of data on DATA[15:8] at an even address	0	1	0	Out	data(odd)	In	x
Word write at an odd and odd+1 internal RAM address (misaligned — only in emulation modes)	0	1	1	Out	data(odd+1)	Out	data(odd)
Word read of data on DATA[15:0] at an even and even+1 address	1	0	0	In	data(even)	In	data(even+1)
Byte read of data on DATA[7:0] at an odd address	1	0	1	In	x	In	data(odd)
Byte read of data on DATA[15:8] at an even address	1	1	0	In	data(even)	In	x
Word read at an odd and odd+1 internal RAM address (misaligned - only in emulation modes)	1	1	1	In	data(odd+1)	In	data(odd)

23.4.6 Low-Power Options

The XEBI does not support any user-controlled options for reducing power consumption.

23.4.6.1 Run Mode

The XEBI does not support any options for reducing power in run mode.

Power consumption is reduced in single-chip modes due to the absence of the external bus interface. Operation in expanded modes results in a higher power consumption, however any unnecessary toggling of external bus signals is reduced to the lowest indispensable activity by holding the previous states between external accesses.

23.4.6.2 Wait Mode

The XEBI does not support any options for reducing power in wait mode.

23.4.6.3 Stop Mode

The XEBI will cease to function in stop mode.

23.5 Initialization/Application Information

This section describes the external bus interface usage and timing. Typical customer operating modes are normal expanded mode and emulation modes, specifically to be used in emulator applications. Taking the availability of the external wait feature into account the use cases are divided into four scenarios:

- Normal expanded mode
  - External wait feature disabled
  - External wait feature enabled
- Emulation modes
  - Emulation single-chip mode (without wait states)
  - Emulation expanded mode (with optional access stretching)

Normal single-chip mode and special single-chip mode do not have an external bus. Special test mode is used for factory test only. Therefore, these modes are omitted here.

All timing diagrams referred to throughout this section are available in the Electrical Characteristics appendix of the SoC section.

23.5.1 Normal Expanded Mode

This mode allows interfacing to external memories or peripherals which are available in the commercial market. In these applications the normal bus operation requires a minimum of 1 cycle stretch for each external access.

23.5.1.1 Example 1a: External Wait Feature Disabled

The first example of bus timing of an external read and write access with the external wait feature disabled is shown in

- Figure ‘Example 1a: Normal Expanded Mode — Read Followed by Write’

The associated supply voltage dependent timing are numbers given in

- Table ‘Example 1a: Normal Expanded Mode Timing V_{DD5} = 5.0 V (EWAITE = 0)’
- Table ‘Example 1a: Normal Expanded Mode Timing V_{DD5} = 3.0 V (EWAITE = 0)’

Systems designed this way rely on the internal programmable access stretching. These systems have predictable external memory access times. The additional stretch time can be programmed up to 8 cycles to provide longer access times.

23.5.1.2 Example 1b: External Wait Feature Enabled

The external wait operation is shown in this example. It can be used to exceed the amount of stretch cycles over the programmed number in EXSTR[2:0]. The feature must be enabled by writing EWAITE = 1.

If the $\overline{\text{EWAIT}}$ signal is not asserted, the number of stretch cycles is forced to a minimum of 2 cycles. If $\overline{\text{EWAIT}}$ is asserted within the predefined time window during the access it will be strobed active and another stretch cycle is added. If strobed inactive, the next cycle will be the last cycle before the access is finished. $\overline{\text{EWAIT}}$ can be held asserted as long as desired to stretch the access.

An access with 1 cycle stretch by $\overline{\text{EWAIT}}$ assertion is shown in

- Figure ‘Example 1b: Normal Expanded Mode — Stretched Read Access’
- Figure ‘Example 1b: Normal Expanded Mode — Stretched Write Access’

The associated timing numbers for both operations are given in

- Table ‘Example 1b: Normal Expanded Mode Timing V_{DD5} = 5.0 V (EWAITE = 1)’
- Table ‘Example 1b: Normal Expanded Mode Timing V_{DD5} = 3.0 V (EWAITE = 1)’

It is recommended to use the free-running clock (ECLK) at the fastest rate (bus clock rate) to synchronize the $\overline{\text{EWAIT}}$ input signal.

23.5.2 Emulation Modes

In emulation mode applications, the development systems use a custom PRU device to rebuild the single-chip or expanded bus functions which are lost due to the use of the external bus with an emulator.

Accesses to a set of registers controlling the related ports in normal modes (refer to SoC section) are directed to the external bus in emulation modes which are substituted by PRR as part of the PRU. Accesses to these registers take a constant time of 2 cycles.

Depending on the setting of ROMON and EROMON (refer to S12X_MMC section), the program code can be executed from internal memory or an optional external emulation memory (EMULMEM). No wait state operation (stretching) of the external bus access is done in emulation modes when accessing internal memory or emulation memory addresses.

In both modes observation of the internal operation is supported through the external bus (internal visibility).

23.5.2.1 Example 2a: Emulation Single-Chip Mode

This mode is used for emulation systems in which the target application is operating in normal single-chip mode.

Figure 23-5 shows the PRU connection with the available external bus signals in an emulator application.

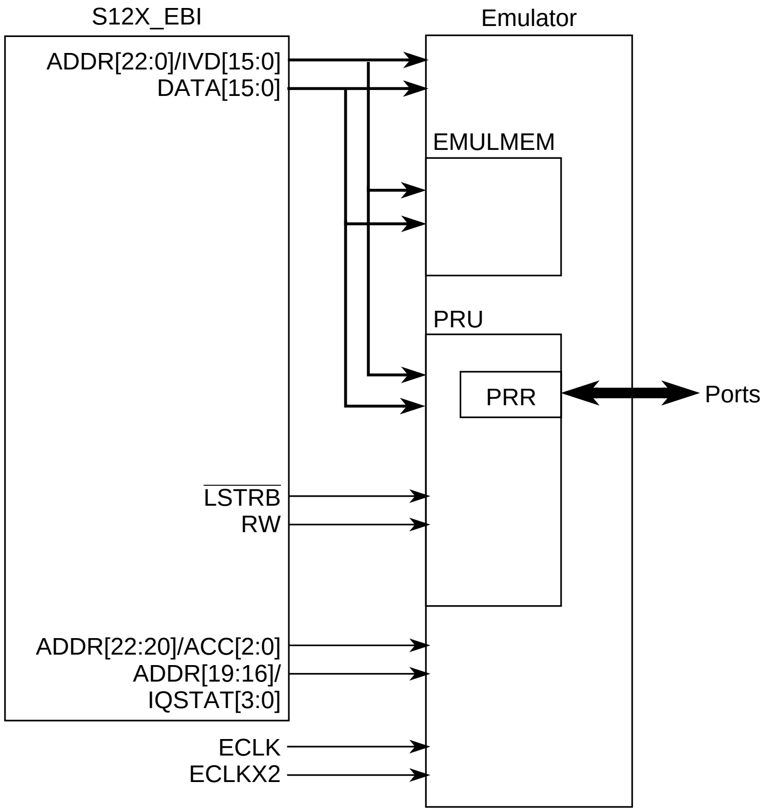


Figure 23-5. Application in Emulation Single-Chip Mode

The timing diagram for this operation is shown in:

- Figure ‘Example 2a: Emulation Single-Chip Mode — Read Followed by Write’

The associated timing numbers are given in:

- Table ‘Example 2a: Emulation Single-Chip Mode Timing (EWAITE = 0)’

Timing considerations:

- Signals muxed with address lines ADDRx, i.e., IVDx, IQSTATx and ACCx, have the same timing.
- $\overline{\text{LSTRB}}$ has the same timing as R$\overline{\text{W}}$.
- ECLKX2 rising edges have the same timing as ECLK edges.
- The timing for accesses to PRU registers, which take 2 cycles to complete, is the same as the timing for an external non-PRR access with 1 cycle of stretch as shown in example 2b.

23.5.2.2 Example 2b: Emulation Expanded Mode

This mode is used for emulation systems in which the target application is operating in normal expanded mode.

If the external bus is used with a PRU, the external device rebuilds the data select and data direction signals $\overline{\text{UDS}}$, $\overline{\text{LDS}}$, $\overline{\text{RE}}$, and $\overline{\text{WE}}$ from the ADDR0, $\overline{\text{LSTRB}}$, and R$\overline{\text{W}}$ signals.

Figure 23-6 shows the PRU connection with the available external bus signals in an emulator application.

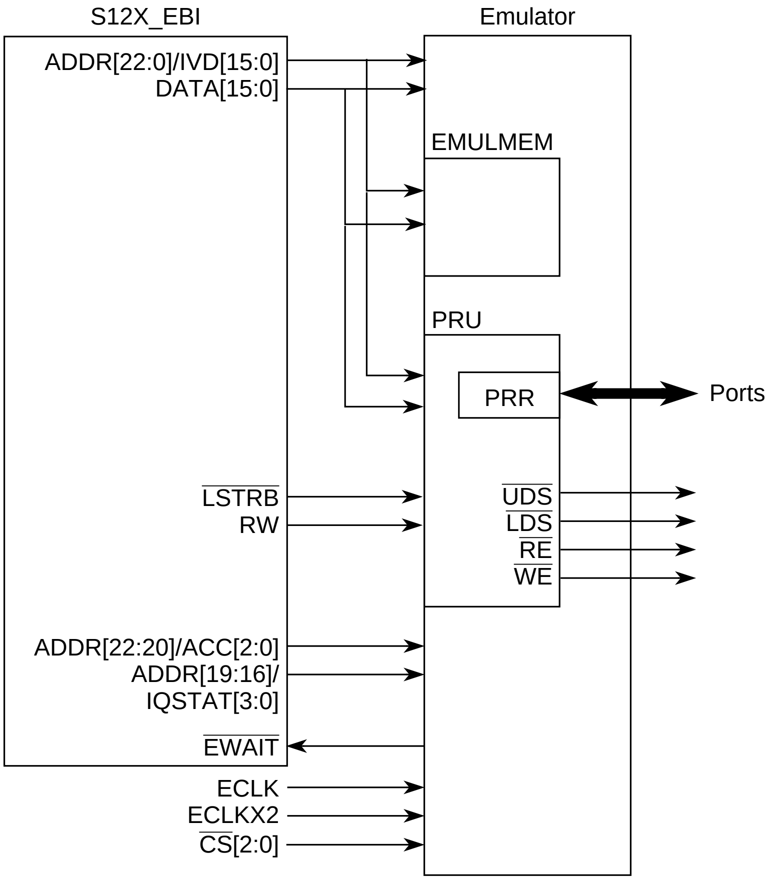


Figure 23-6. Application in Emulation Expanded Mode

The timings of accesses with 1 stretch cycle are shown in

- Figure ‘Example 2b: Emulation Expanded Mode — Read with 1 Stretch Cycle’
- Figure ‘Example 2b: Emulation Expanded Mode — Write with 1 Stretch Cycle’

The associated timing numbers are given in

- Table ‘Example 2b: Emulation Expanded Mode Timing V_{DD5} = 5.0 V (EWAITE = 0)’ (this also includes examples for alternative settings of 2 and 3 additional stretch cycles)

Timing considerations:

- If no stretch cycle is added, the timing is the same as in Emulation Single-Chip Mode.

Chapter 24
Interrupt (S12XINTV1)

24.1 Introduction

The XINT module decodes the priority of all system exception requests and provides the applicable vector for processing the exception to either the CPU or the XGATE module. The XINT module supports:

- I bit and X bit maskable interrupt requests
- A non-maskable unimplemented opcode trap
- A non-maskable software interrupt (SWI) or background debug mode request
- A spurious interrupt vector request
- Three system reset vector requests

Each of the I bit maskable interrupt requests can be assigned to one of seven priority levels supporting a flexible priority scheme. For interrupt requests that are configured to be handled by the CPU, the priority scheme can be used to implement nested interrupt capability where interrupts from a lower level are automatically blocked if a higher level interrupt is being processed. Interrupt requests configured to be handled by the XGATE module cannot be nested because the XGATE module cannot be interrupted while processing.

NOTE

The HPRIO register and functionality of the XINT module is no longer supported, since it is superseded by the 7-level interrupt request priority scheme.

24.1.1 Glossary

The following terms and abbreviations are used in the document.

Table 24-1. Terminology

Term	Meaning
CCR	Condition Code Register (in the S12X CPU)
DMA	Direct Memory Access
INT	Interrupt
IPL	Interrupt Processing Level
ISR	Interrupt Service Routine
MCU	Micro-Controller Unit
XGATE	please refer to the "XGATE Block Guide"
$\overline{\text{IRQ}}$	refers to the interrupt request associated with the $\overline{\text{IRQ}}$ pin
$\overline{\text{XIRQ}}$	refers to the interrupt request associated with the $\overline{\text{XIRQ}}$ pin

24.1.2 Features

- Interrupt vector base register (IVBR)
- One spurious interrupt vector (at address vector base[1] + 0x0010).
- 2–113 I bit maskable interrupt vector requests (at addresses vector base + 0x0012–0x00F2).
- Each I bit maskable interrupt request has a configurable priority level and can be configured to be handled by either the CPU or the XGATE module[2].
- I bit maskable interrupts can be nested, depending on their priority levels.
- One X bit maskable interrupt vector request (at address vector base + 0x00F4).
- One non-maskable software interrupt request (SWI) or background debug mode vector request (at address vector base + 0x00F6).
- One non-maskable unimplemented opcode trap (TRAP) vector (at address vector base + 0x00F8).
- Three system reset vectors (at addresses 0xFFFA–0xFFFE).
- Determines the highest priority DMA and interrupt vector requests, drives the vector to the XGATE module or to the bus on CPU request, respectively.
- Wakes up the system from stop or wait mode when an appropriate interrupt request occurs or whenever $\overline{\text{XIRQ}}$ is asserted, even if X interrupt is masked.
- XGATE can wake up and execute code, even with the CPU remaining in stop or wait mode.

24.1.3 Modes of Operation

- Run mode

  This is the basic mode of operation.

---

[1] The vector base is a 16-bit address which is accumulated from the contents of the interrupt vector base register (IVBR, used as upper byte) and 0x00 (used as lower byte).

[2] The $\overline{\text{IRQ}}$ interrupt can only be handled by the CPU

- Wait mode

  In wait mode, the XINT module is frozen. It is however capable of either waking up the CPU if an interrupt occurs or waking up the XGATE if an XGATE request occurs. Please refer to Section 24.5.3, “Wake Up from Stop or Wait Mode” for details.

- Stop Mode

  In stop mode, the XINT module is frozen. It is however capable of either waking up the CPU if an interrupt occurs or waking up the XGATE if an XGATE request occurs. Please refer to Section 24.5.3, “Wake Up from Stop or Wait Mode” for details.

- Freeze mode (BDM active)

  In freeze mode (BDM active), the interrupt vector base register is overridden internally. Please refer to Section 24.3.1.1, “Interrupt Vector Base Register (IVBR)” for details.

24.1.4 Block Diagram

Figure 24-1 shows a block diagram of the XINT module.

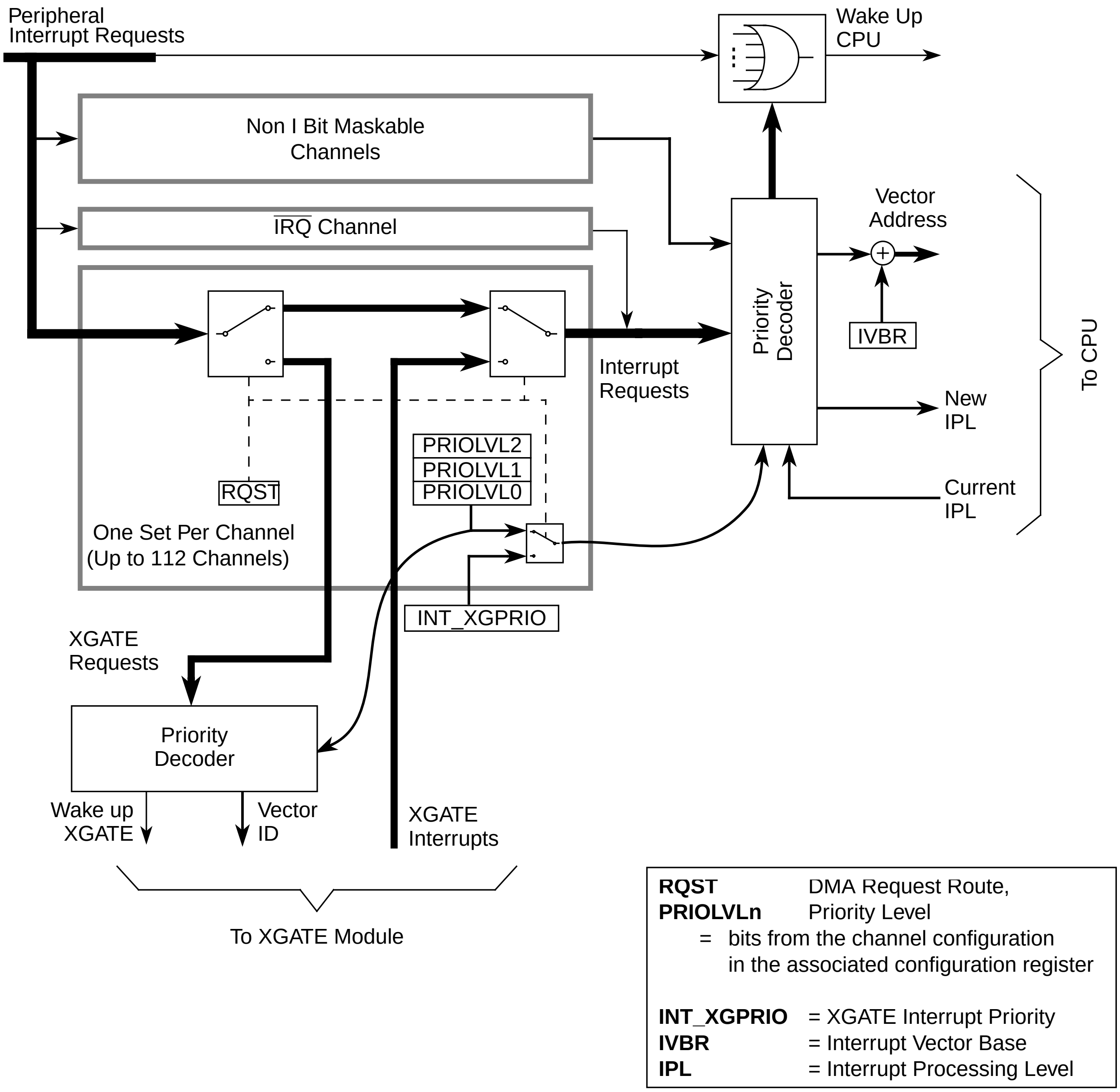


Figure 24-1. XINT Block Diagram

24.2 External Signal Description

The XINT module has no external signals.

24.3 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in the XINT.

24.3.1 Register Descriptions

This section describes in address order all the XINT registers and their individual bits.

Address	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0121	IVBR	R	IVB_ADDR[7:0]							
		W								
0x0126	INT_XGPRIO	R	0	0	0	0	0	XILVL[2:0]		
		W	▨	▨	▨	▨	▨			
0x0127	INT_CFADDR	R	INT_CFADDR[7:4]				0	0	0	0
		W					▨	▨	▨	▨
0x0128	INT_CFDATA0	R	RQST	0	0	0	0	PRIOLVL[2:0]		
		W		▨	▨	▨	▨			
0x0129	INT_CFDATA1	R	RQST	0	0	0	0	PRIOLVL[2:0]		
		W		▨	▨	▨	▨			
0x012A	INT_CFDATA2	R	RQST	0	0	0	0	PRIOLVL[2:0]		
		W		▨	▨	▨	▨			
0x012B	INT_CFDATA3	R	RQST	0	0	0	0	PRIOLVL[2:0]		
		W		▨	▨	▨	▨			
0x012C	INT_CFDATA4	R	RQST	0	0	0	0	PRIOLVL[2:0]		
		W		▨	▨	▨	▨			
0x012D	INT_CFDATA5	R	RQST	0	0	0	0	PRIOLVL[2:0]		
		W		▨	▨	▨	▨			
0x012E	INT_CFDATA6	R	RQST	0	0	0	0	PRIOLVL[2:0]		
		W		▨	▨	▨	▨			
0x012F	INT_CFDATA7	R	RQST	0	0	0	0	PRIOLVL[2:0]		
		W		▨	▨	▨	▨			

▨ = Unimplemented or Reserved

Figure 24-2. XINT Register Summary

24.3.1.1 Interrupt Vector Base Register (IVBR)

Address: 0x0121

	7	6	5	4	3	2	1	0
R W	IVB_ADDR[7:0]							
Reset	1	1	1	1	1	1	1	1

Figure 24-3. Interrupt Vector Base Register (IVBR)

Read: Anytime

Write: Anytime

Table 24-2. IVBR Field Descriptions

Field	Description
7–0 IVB_ADDR[7:0]	**Interrupt Vector Base Address Bits** — These bits represent the upper byte of all vector addresses. Out of reset these bits are set to 0xFF (i.e., vectors are located at 0xFF10–0xFFFE) to ensure compatibility to HCS12. **Note:** A system reset will initialize the interrupt vector base register with “0xFF” before it is used to determine the reset vector address. Therefore, changing the IVBR has no effect on the location of the three reset vectors (0xFFFA–0xFFFE). **Note:** If the BDM is active (i.e., the CPU is in the process of executing BDM firmware code), the contents of IVBR are ignored and the upper byte of the vector address is fixed as “0xFF”.

24.3.1.2 XGATE Interrupt Priority Configuration Register (INT_XGPRIO)

Address: 0x0126

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	XILVL[2:0]		
W	▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	1

▨ = Unimplemented or Reserved

Figure 24-4. XGATE Interrupt Priority Configuration Register (INT_XGPRIO)

Read: Anytime

Write: Anytime

Table 24-3. INT_XGPRIO Field Descriptions

Field	Description
2–0 XILVL[2:0]	**XGATE Interrupt Priority Level** — The XILVL[2:0] bits configure the shared interrupt level of the DMA interrupts coming from the XGATE module. Out of reset the priority is set to the lowest active level (“1”).

Table 24-4. XGATE Interrupt Priority Levels

Priority	XILVL2	XILVL1	XILVL0	Meaning
	0	0	0	Interrupt request is disabled
low	0	0	1	Priority level 1
	0	1	0	Priority level 2
	0	1	1	Priority level 3
	1	0	0	Priority level 4
	1	0	1	Priority level 5
	1	1	0	Priority level 6
high	1	1	1	Priority level 7

24.3.1.3 Interrupt Request Configuration Address Register (INT_CFADDR)

Address: 0x0127

	7	6	5	4	3	2	1	0
R	INT_CFADDR[7:4]				0	0	0	0
W					▨	▨	▨	▨
Reset	0	0	0	1	0	0	0	0

▨ = Unimplemented or Reserved

Figure 24-5. Interrupt Configuration Address Register (INT_CFADDR)

Read: Anytime

Write: Anytime

Table 24-5. INT_CFADDR Field Descriptions

Field	Description
7–4 INT_CFADDR[7:4]	**Interrupt Request Configuration Data Register Select Bits** — These bits determine which of the 128 configuration data registers are accessible in the 8 register window at INT_CFDATA0–7. The hexadecimal value written to this register corresponds to the upper nibble of the lower byte of the interrupt vector, i.e., writing 0xE0 to this register selects the configuration data register block for the 8 interrupt vector requests starting with vector (vector base + 0x00E0) to be accessible as INT_CFDATA0–7. **Note:** Writing all 0s selects non-existing configuration registers. In this case write accesses to INT_CFDATA0–7 will be ignored and read accesses will return all 0.

24.3.1.4 Interrupt Request Configuration Data Registers (INT_CFDATA0–7)

The eight register window visible at addresses INT_CFDATA0–7 contains the configuration data for the block of eight interrupt requests (out of 128) selected by the interrupt configuration address register (INT_CFADDR) in ascending order. INT_CFDATA0 represents the interrupt configuration data register of the vector with the lowest address in this block, while INT_CFDATA7 represents the interrupt configuration data register of the vector with the highest address, respectively.

Address: 0x0128

	7	6	5	4	3	2	1	0
R	RQST	0	0	0	0	PRIOLVL[2:0]		
W		▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	1[1]

▨ = Unimplemented or Reserved

Figure 24-6. Interrupt Request Configuration Data Register 0 (INT_CFDATA0)

[1] Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x0129

	7	6	5	4	3	2	1	0
R	RQST	0	0	0	0	PRIOLVL[2:0]		
W		▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	1[1]

▨ = Unimplemented or Reserved

Figure 24-7. Interrupt Request Configuration Data Register 1 (INT_CFDATA1)

[1] Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x012A

	7	6	5	4	3	2	1	0
R	RQST	0	0	0	0	PRIOLVL[2:0]		
W		▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	1[1]

▨ = Unimplemented or Reserved

Figure 24-8. Interrupt Request Configuration Data Register 2 (INT_CFDATA2)

[1] Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x012B

	7	6	5	4	3	2	1	0
R	RQST	0	0	0	0	PRIOLVL[2:0]		
W		▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	1[1]

▨ = Unimplemented or Reserved

Figure 24-9. Interrupt Request Configuration Data Register 3 (INT_CFDATA3)

[1] Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x012C

	7	6	5	4	3	2	1	0
R	RQST	0	0	0	0	PRIOLVL[2:0]		
W		▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	1[1]

▨ = Unimplemented or Reserved

Figure 24-10. Interrupt Request Configuration Data Register 4 (INT_CFDATA4)

[1] Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x012D

	7	6	5	4	3	2	1	0
R	RQST	0	0	0	0	PRIOLVL[2:0]		
W		▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	1[1]

▨ = Unimplemented or Reserved

Figure 24-11. Interrupt Request Configuration Data Register 5 (INT_CFDATA5)

[1] Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x012E

	7	6	5	4	3	2	1	0
R	RQST	0	0	0	0	PRIOLVL[2:0]		
W		▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	1[1]

▨ = Unimplemented or Reserved

Figure 24-12. Interrupt Request Configuration Data Register 6 (INT_CFDATA6)

[1] Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x012F

	7	6	5	4	3	2	1	0
R	RQST	0	0	0	0	PRIOLVL[2:0]		
W		▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	1[1]

▨ = Unimplemented or Reserved

Figure 24-13. Interrupt Request Configuration Data Register 7 (INT_CFDATA7)

[1] Please refer to the notes following the PRIOLVL[2:0] description below.

Read: Anytime

Write: Anytime

Table 24-6. INT_CFDATA0–7 Field Descriptions

Field	Description
7 RQST	**XGATE Request Enable** — This bit determines if the associated interrupt request is handled by the CPU or by the XGATE module. 0 Interrupt request is handled by the CPU 1 Interrupt request is handled by the XGATE module **Note:** The $\overline{\text{IRQ}}$ interrupt cannot be handled by the XGATE module. For this reason, the configuration register for vector (vector base + 0x00F2) = $\overline{\text{IRQ}}$ vector address) does not contain a RQST bit. Writing a 1 to the location of the RQST bit in this register will be ignored and a read access will return 0.
2–0 PRIOLVL[2:0]	**Interrupt Request Priority Level Bits** — The PRIOLVL[2:0] bits configure the interrupt request priority level of the associated interrupt request. Out of reset all interrupt requests are enabled at the lowest active level (“1”) to provide backwards compatibility with previous HCS12 interrupt controllers. Please also refer to Table 24-7 for available interrupt request priority levels. **Note:** Write accesses to configuration data registers of unused interrupt channels will be ignored and read accesses will return all 0. For information about what interrupt channels are used in a specific MCU, please refer to the Device User Guide of that MCU. **Note:** When vectors (vector base + 0x00F0–0x00FE) are selected by writing 0xF0 to INT_CFADDR, writes to INT_CFDATA2–7 (0x00F4–0x00FE) will be ignored and read accesses will return all 0s. The corresponding vectors do not have configuration data registers associated with them. **Note:** Write accesses to the configuration register for the spurious interrupt vector request (vector base + 0x0010) will be ignored and read accesses will return 0x07 (request is handled by the CPU, PRIOLVL = 7).

Table 24-7. Interrupt Priority Levels

Priority	PRIOLVL2	PRIOLVL1	PRIOLVL0	Meaning
	0	0	0	Interrupt request is disabled
low	0	0	1	Priority level 1
	0	1	0	Priority level 2
	0	1	1	Priority level 3
	1	0	0	Priority level 4
	1	0	1	Priority level 5
	1	1	0	Priority level 6
high	1	1	1	Priority level 7

24.4 Functional Description

The XINT module processes all exception requests to be serviced by the CPU module. These exceptions include interrupt vector requests and reset vector requests. Each of these exception types and their overall priority level is discussed in the subsections below.

24.4.1 S12X Exception Requests

The CPU handles both reset requests and interrupt requests. The XINT contains registers to configure the priority level of each I bit maskable interrupt request which can be used to implement an interrupt priority scheme. This also includes the possibility to nest interrupt requests. A priority decoder is used to evaluate the priority of a pending interrupt request.

24.4.2 Interrupt Prioritization

After system reset all interrupt requests with a vector address lower than or equal to (vector base + 0x00F2) are enabled, are set up to be handled by the CPU and have a pre-configured priority level of 1. The exception to this rule is the spurious interrupt vector request at (vector base + 0x0010) which cannot be disabled, is always handled by the CPU and has a fixed priority level of 7. A priority level of 0 effectively disables the associated interrupt request.

If more than one interrupt request is configured to the same interrupt priority level the interrupt request with the higher vector address wins the prioritization.

The following conditions must be met for an I bit maskable interrupt request to be processed.

1. The local interrupt enabled bit in the peripheral module must be set.
2. The setup in the configuration register associated with the interrupt request channel must meet the following conditions:
   a) The XGATE request enable bit must be 0 to have the CPU handle the interrupt request.
   b) The priority level must be set to non zero.
   c) The priority level must be greater than the current interrupt processing level in the condition code register (CCR) of the CPU (PRIOLVL[2:0] > IPL[2:0]).
3. The I bit in the condition code register (CCR) of the CPU must be cleared.
4. There is no SWI, TRAP, or $\overline{\text{XIRQ}}$ request pending.

NOTE

All non I bit maskable interrupt requests always have higher priority than I bit maskable interrupt requests. If an I bit maskable interrupt request is interrupted by a non I bit maskable interrupt request, the currently active interrupt processing level (IPL) remains unaffected. It is possible to nest non I bit maskable interrupt requests, e.g., by nesting SWI or TRAP calls.

24.4.2.1 Interrupt Priority Stack

The current interrupt processing level (IPL) is stored in the condition code register (CCR) of the CPU. This way the current IPL is automatically pushed to the stack by the standard interrupt stacking procedure. The new IPL is copied to the CCR from the priority level of the highest priority active interrupt request channel which is configured to be handled by the CPU. The copying takes place when the interrupt vector is fetched. The previous IPL is automatically restored by executing the RTI instruction.

24.4.3 XGATE Requests

The XINT module processes all exception requests to be serviced by the XGATE module. The overall priority level of those exceptions is discussed in the subsections below.

24.4.3.1 XGATE Request Prioritization

An interrupt request channel is configured to be handled by the XGATE module, if the RQST bit of the associated configuration register is set to 1 (please refer to Section 24.3.1.4, “Interrupt Request Configuration Data Registers (INT_CFDATA0–7)”). The priority level setting (PRIOLVL) for this channel becomes the DMA priority which will be used to determine the highest priority DMA request to be serviced next by the XGATE module. Additionally, DMA interrupts may be raised by the XGATE module by setting one or more of the XGATE channel interrupt flags (using the SIF instruction). This will result in an CPU interrupt with vector address vector base + (2 * channel ID number), where the channel ID number corresponds to the highest set channel interrupt flag, if the XGIE and channel RQST bits are set.

The shared interrupt priority for the DMA interrupt requests is taken from the XGATE interrupt priority configuration register (please refer to Section 24.3.1.2, “XGATE Interrupt Priority Configuration Register (INT_XGPRIO)”). If more than one DMA interrupt request channel becomes active at the same time, the channel with the highest vector address wins the prioritization.

24.4.4 Priority Decoders

The XINT module contains priority decoders to determine the priority for all interrupt requests pending for the respective target.

There are two priority decoders, one for each interrupt request target (CPU, XGATE module). The function of both priority decoders is basically the same with one exception: the priority decoder for the XGATE module does not take the current interrupt processing level into account because XGATE requests cannot be nested.

Because the vector is not supplied until the CPU requests it, it is possible that a higher priority interrupt request could override the original exception that caused the CPU to request the vector. In this case, the CPU will receive the highest priority vector and the system will process this exception instead of the original request.

If the interrupt source is unknown (for example, in the case where an interrupt request becomes inactive after the interrupt has been recognized, but prior to the vector request), the vector address supplied to the CPU will default to that of the spurious interrupt vector.

NOTE

> Care must be taken to ensure that all exception requests remain active until the system begins execution of the applicable service routine; otherwise, the exception request may not get processed at all or the result may be a spurious interrupt request (vector at address (vector base + 0x0010)).

24.4.5 Reset Exception Requests

The XINT supports three system reset exception request types (please refer to CRG for details):

1. Pin reset, power-on reset, low-voltage reset, or illegal address reset
2. Clock monitor reset request
3. COP watchdog reset request

24.4.6 Exception Priority

The priority (from highest to lowest) and address of all exception vectors issued by the XINT upon request by the CPU is shown in Table 24-8.

Table 24-8. Exception Vector Map and Priority

Vector Address[1]	Source
0xFFFE	Pin reset, power-on reset, low-voltage reset, illegal address reset
0xFFFC	Clock monitor reset
0xFFFA	COP watchdog reset
(Vector base + 0x00F8)	Unimplemented opcode trap
(Vector base + 0x00F6)	Software interrupt instruction (SWI) or BDM vector request
(Vector base + 0x00F4)	$\overline{\text{XIRQ}}$ interrupt request
(Vector base + 0x00F2)	$\overline{\text{IRQ}}$ interrupt request
(Vector base + 0x00F0–0x0012)	Device specific I bit maskable interrupt sources (priority determined by the associated configuration registers, in descending order)
(Vector base + 0x0010)	Spurious interrupt

[1] 16 bits vector address based

24.5 Initialization/Application Information

24.5.1 Initialization

After system reset, software should:

- Initialize the interrupt vector base register if the interrupt vector table is not located at the default location (0xFF10–0xFFF9).
- Initialize the interrupt processing level configuration data registers (INT_CFADDR, INT_CFDATA0–7) for all interrupt vector requests with the desired priority levels and the request target (CPU or XGATE module). It might be a good idea to disable unused interrupt requests.
- If the XGATE module is used, setup the XGATE interrupt priority register (INT_XGPRIO) and configure the XGATE module (please refer the XGATE Block Guide for details).
- Enable I maskable interrupts by clearing the I bit in the CCR.
- Enable the X maskable interrupt by clearing the X bit in the CCR (if required).

24.5.2 Interrupt Nesting

The interrupt request priority level scheme makes it possible to implement priority based interrupt request nesting for the I bit maskable interrupt requests handled by the CPU.

- I bit maskable interrupt requests can be interrupted by an interrupt request with a higher priority, so that there can be up to seven nested I bit maskable interrupt requests at a time (refer to Figure 24-14 for an example using up to three nested interrupt requests).

I bit maskable interrupt requests cannot be interrupted by other I bit maskable interrupt requests per default. In order to make an interrupt service routine (ISR) interruptible, the ISR must explicitly clear the I bit in the CCR (CLI). After clearing the I bit, I bit maskable interrupt requests with higher priority can interrupt the current ISR.

An ISR of an interruptible I bit maskable interrupt request could basically look like this:

- Service interrupt, e.g., clear interrupt flags, copy data, etc.
- Clear I bit in the CCR by executing the instruction CLI (thus allowing interrupt requests with higher priority)
- Process data
- Return from interrupt by executing the instruction RTI

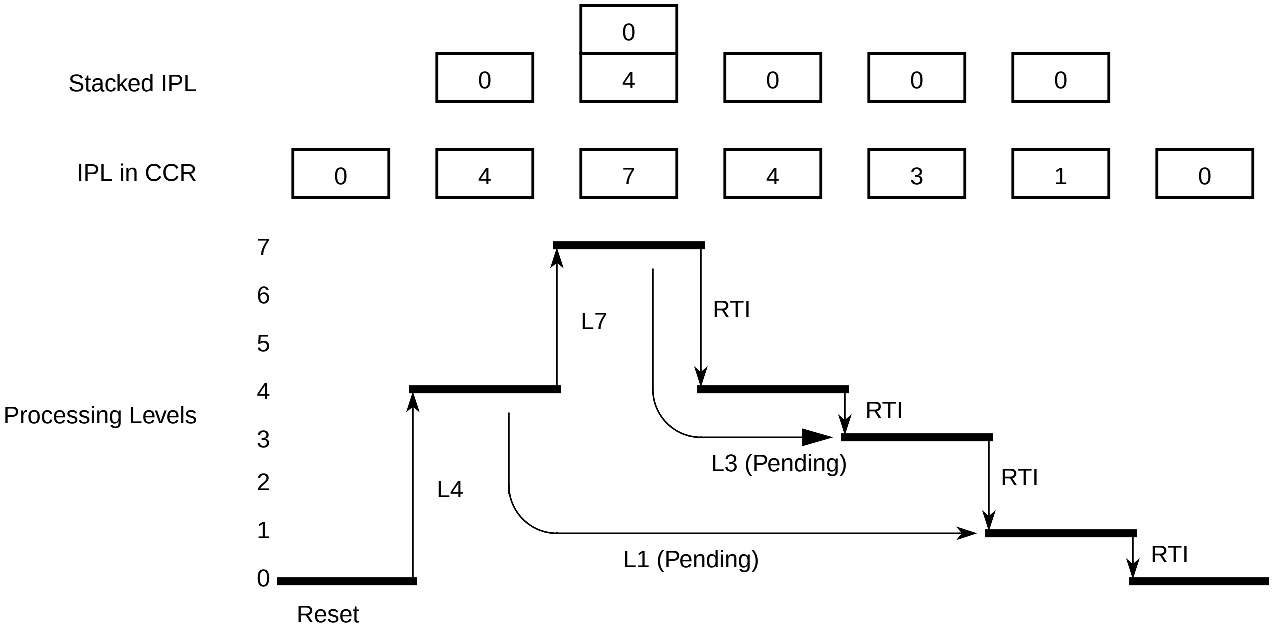


Figure 24-14. Interrupt Processing Example

24.5.3 Wake Up from Stop or Wait Mode

24.5.3.1 CPU Wake Up from Stop or Wait Mode

Every I bit maskable interrupt request which is configured to be handled by the CPU is capable of waking the MCU from stop or wait mode. To determine whether an I bit maskable interrupts is qualified to wake up the CPU or not, the same settings as in normal run mode are applied during stop or wait mode:

- If the I bit in the CCR is set, all I bit maskable interrupts are masked from waking up the MCU.
- An I bit maskable interrupt is ignored if it is configured to a priority level below or equal to the current IPL in CCR.
- I bit maskable interrupt requests which are configured to be handled by the XGATE are not capable of waking up the CPU.

An $\overline{\text{XIRQ}}$ request can wake up the MCU from stop or wait mode at anytime, even if the X bit in CCR is set.

24.5.3.2 XGATE Wake Up from Stop or Wait Mode

Interrupt request channels which are configured to be handled by the XGATE are capable of waking up the XGATE. Interrupt request channels handled by the XGATE do not affect the state of the CPU.

Chapter 25
Memory Mapping Control (S12XMMCV3)

Version Number	Revision Date	Effective Date	Author	Description of Changes
v03.00	25 May 2005	05/25/2005		Generic S12XMMC BlockGuide is meant for S12X derivatives. - Added FLEXRAY IP like a Master Block. - Major Cleanup,. - Added condional texts to different configurations. - Added Internal section.
v03.01	21 July 2005	07/21/2005		Clarify in details External Spaces accesses and firmware in single chip modes Update reviewed wording

25.1 Introduction

This section describes the functionality of the module mapping control (MMC) sub-block of the S12X platform. The block diagram of the MMC is shown in Figure 25-1.

The MMC module controls the multi-master priority accesses, the selection of internal resources and external space. Internal buses, including internal memories and peripherals, are controlled in this module. The local address space for each master is translated to a global memory space.

25.1.1 Terminology

Table 25-1. Acronyms and Abbreviations

Logic level “1”	Voltage that corresponds to Boolean true state
Logic level “0”	Voltage that corresponds to Boolean false state
0x	Represents hexadecimal number
x	Represents logic level ’don’t care’
byte	8-bit data
word	16-bit data
local address	based on the 64 KBytes Memory Space (16-bit address)
global address	based on the 8 MBytes Memory Space (23-bit address)
Aligned address	Address on even boundary
Mis-aligned address	Address on odd boundary
Bus Clock	System Clock. Refer to CRG Block Guide.
expanded modes	Normal Expanded Mode Emulation Single-Chip Mode Emulation Expanded Mode Special Test Mode
single-chip modes	Normal Single-Chip Mode Special Single-Chip Mode
emulation modes	Emulation Single-Chip Mode Emulation Expanded Mode
normal modes	Normal Single-Chip Mode Normal Expanded Mode
special modes	Special Single-Chip Mode Special Test Mode
NS	Normal Single-Chip Mode
SS	Special Single-Chip Mode
NX	Normal Expanded Mode
ES	Emulation Single-Chip Mode
EX	Emulation Expanded Mode
ST	Special Test Mode
Unimplemented areas	Areas which are accessible by the pages (RPAGE,PPAGE,EPAGE) and not implemented
External Space	Area which is accessible in the global address range 14_0000 to 3F_FFFF
external resource	Resources (Emulator, Application) connected to the MCU via the external bus on expanded modes (Unimplemented areas and External Space)
PRR	Port Replacement Registers
PRU	Port Replacement Unit located on the emulator side
MCU	MicroController Unit
NVM	Non-volatile Memory; Flash EEPROM or ROM

25.1.2 Features

The main features of this block are:

- Paging capability to support a global 8 Mbytes memory address space
- Bus arbitration between the masters CPU, BDM and XGATE

- Simultaneous accesses to different resources[1] (internal, external, and peripherals) (see)
- Resolution of target bus access collision
- Access restriction control from masters to some targets (e.g., RAM write access protection for user specified areas)
- MCU operation mode control
- MCU security control
- Separate memory map schemes for each master CPU, BDM and XGATE
- ROM control bits to enable the on-chip FLASH or ROM selection
- Port replacement registers access control
- Generation of system reset when CPU accesses an unimplemented address (i.e., an address which does not belong to any of the on-chip modules) in single-chip modes

25.1.3 S12X Memory Mapping

The S12X architecture implements a number of memory mapping schemes including

- a CPU 8 MByte global map, defined using a global page (GPAGE) register and dedicated 23-bit address load/store instructions.
- a BDM 8 MByte global map, defined using a global page (BDMGPR) register and dedicated 23-bit address load/store instructions.
- a (CPU or BDM) 64 KByte local map, defined using specific resource page (RPAGE, EPAGE and PPAGE) registers and the default instruction set. The 64 KBytes visible at any instant can be considered as the local map accessed by the 16-bit (CPU or BDM) address.
- The XGATE 64 Kbyte local map.

The MMC module performs translation of the different memory mapping schemes to the specific global (physical) memory implementation.

25.1.4 Modes of Operation

This subsection lists and briefly describes all operating modes supported by the MMC.

25.1.4.1 Power Saving Modes

- Run mode

  MMC is functional during normal run mode.
- Wait mode

  MMC is functional during wait mode.
- Stop mode

  MMC is inactive during stop mode.

---

[1] Resources are also called targets.

25.1.4.2 Functional Modes

- Single chip modes

  In normal and special single chip mode the internal memory is used. External bus is not active.
- Expanded modes

  Address, data, and control signals are activated in normal expanded and special test modes when accessing the external bus. Access to internal resources will not cause activity on the external bus.
- Emulation modes

  External bus is active to emulate, via an external tool, the normal expanded or the normal single chip mode.

25.1.5 Block Diagram

Figure 25-1[1] shows a block diagram of the MMC.

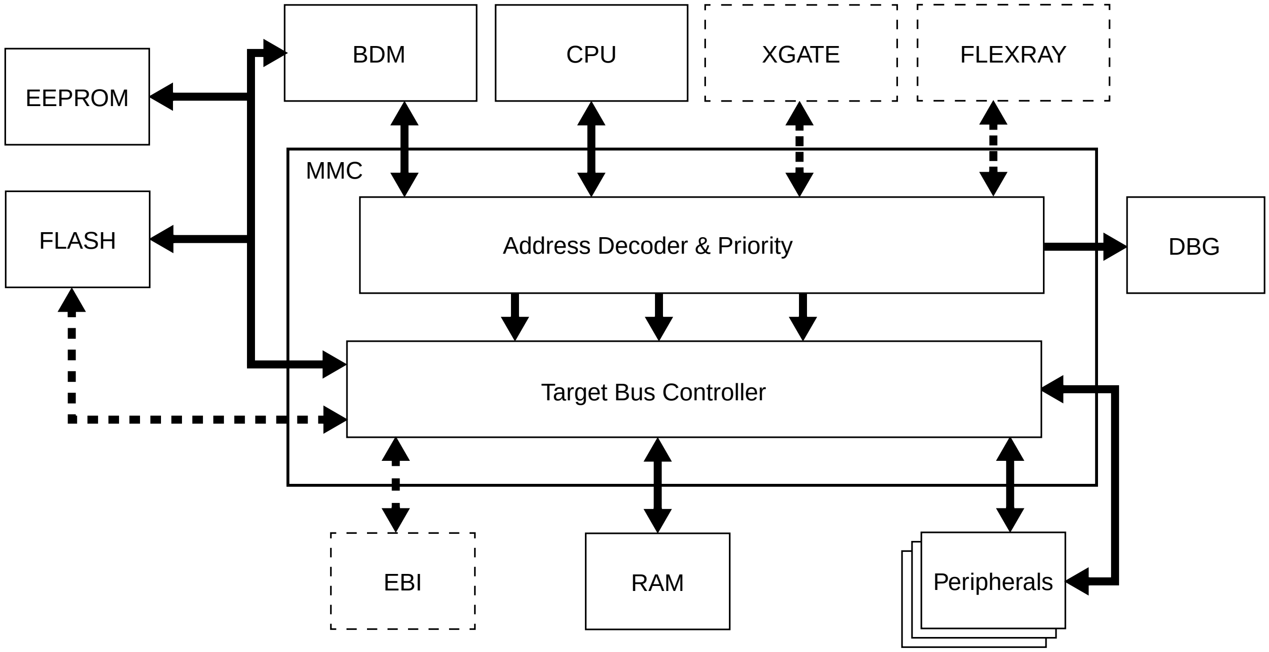


Figure 25-1. MMC Block Diagram

25.2 External Signal Description

The user is advised to refer to the SoC Guide for port configuration and location of external bus signals. Some pins may not be bonded out in all implementations.

Table 25-2 and Table 25-3 outline the pin names and functions. It also provides a brief description of their operation.

---

1. Doted blocks and lines are optional. Please refer to the Device User Guide for their availlibilities.

Table 25-2. External Input Signals Associated with the MMC

Signal	I/O	Description	Availability
MODC	I	Mode input	Latched after $\overline{\text{RESET}}$ (active low)
MODB	I	Mode input	Latched after $\overline{\text{RESET}}$ (active low)
MODA	I	Mode input	Latched after $\overline{\text{RESET}}$ (active low)
EROMCTL	I	EROM control input	Latched after $\overline{\text{RESET}}$ (active low)
ROMCTL	I	ROM control input	Latched after $\overline{\text{RESET}}$ (active low)

Table 25-3. External Output Signals Associated with the MMC

Signal	I/O	Description	Available in Modes					
			NS	SS	NX	ES	EX	ST
CS0	O	Chip select line 0	(see Table 25-4)					
CS1	O	Chip select line 1						
CS2	O	Chip select line 2						
CS3	O	Chip select line 3						

25.3 Memory Map and Registers

25.3.1 Module Memory Map

A summary of the registers associated with the MMC block is shown in Figure 25-2. Detailed descriptions of the registers and bits are given in the subsections that follow.

Address	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x000A	MMCCTL0	R	0	0	0	0	CS3E	CS2E	CS1E	CS0E
		W	▨	▨	▨	▨				
0x000B	MODE	R	MODC	MODB	MODA	0	0	0	0	0
		W				▨	▨	▨	▨	▨
0x0010	GPAGE	R	0	GP6	GP5	GP4	GP3	GP2	GP1	GP0
		W	▨							
0x0011	DIRECT	R	DP15	DP14	DP13	DP12	DP11	DP10	DP9	DP8
		W								
0x0012	Reserved	R	0	0	0	0	0	0	0	0
		W	▨	▨	▨	▨	▨	▨	▨	▨
0x0013	MMCCTL1	R	0	0	0	0	0	EROMON	ROMHM	ROMON
		W	▨	▨	▨	▨	▨			
0x0014	Reserved	R	0	0	0	0	0	0	0	0
		W	▨	▨	▨	▨	▨	▨	▨	▨
0x0015	Reserved	R	0	0	0	0	0	0	0	0
		W	▨	▨	▨	▨	▨	▨	▨	▨
0x0016	RPAGE	R	RP7	RP6	RP5	RP4	RP3	RP2	RP1	RP0
		W								
0x0017	EPAGE	R	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
		W								
0x0030	PPAGE	R	PIX7	PIX6	PIX5	PIX4	PIX3	PIX2	PIX1	PIX0
		W								
0x0031	Reserved	R	0	0	0	0	0	0	0	0
		W	▨	▨	▨	▨	▨	▨	▨	▨

▨ = Unimplemented or Reserved

Figure 25-2. MMC Register Summary

Address	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x011D	RAMXGU	R	1	XGU6	XGU5	XGU4	XGU3	XGU2	XGU1	XGU0
		W	▨							
0x011E	RAMSHL	R	1	SHL6	SHL5	SHL4	SHL3	SHL2	SHL1	SHL0
		W	▨							
0x011F	RAMSHU	R	1	SHU6	SHU5	SHU4	SHU3	SHU2	SHU1	SHU0
		W	▨							

▨ = Unimplemented or Reserved

Figure 25-2. MMC Register Summary

25.3.2 Register Descriptions

25.3.2.1 MMC Control Register (MMCCTL0)

Address: 0x000A PRR

	7	6	5	4	3	2	1	0
R	0	0	0	0	CS3E	CS2E	CS1E	CS0E
W	▨	▨	▨	▨				
Reset	0	0	0	0	0	0	0	$\overline{\text{ROMON}}$[1]

1. ROMON is bit[0] of the register MMCTL1 (see Figure 25-10)

▨ = Unimplemented or Reserved

Figure 25-3. MMC Control Register (MMCCTL0)

Read: Anytime. In emulation modes read operations will return the data from the external bus. In all other modes the data is read from this register.

Write: Anytime. In emulation modes write operations will also be directed to the external bus.

Table 25-4. Chip Selects Function Activity

Register Bit	Chip Modes					
	NS	SS	NX	ES	EX	ST
CS3E, CS2E, CS1E, CS0E	Disabled[1]	Disabled	Enabled[2]	Disabled	Enabled	Enabled

[1] Disabled: feature always inactive.

[2] Enabled: activity is controlled by the appropriate register bit value.

The MMCCTL0 register is used to control external bus functions, i.e., availability of chip selects.

CAUTION

XGATE write access to this register during an CPU access which makes use of this register could lead to unexpected results.

Table 25-5. MMCCTL0 Field Descriptions

Field	Description
3–0 CS[3:0]E	**Chip Select Enables** — Each of these bits enables one of the external chip selects $\overline{\text{CS3}}$, $\overline{\text{CS2}}$, $\overline{\text{CS1}}$, and $\overline{\text{CS0}}$ outputs which are asserted during accesses to specific external addresses. The associated global address ranges are shown in Table 25-6 and Table 25-21 and Figure 25-21. Chip selects are only active if enabled in normal expanded mode, Emulation expanded mode and special test mode. The function disabled in all other operating modes. 0 Chip select is disabled 1 Chip select is enabled

Table 25-6. Chip Select Signals

Global Address Range	Asserted Signal
0x00_0800–0x0F_FFFF	$\overline{\text{CS3}}$
0x10_0000–0x1F_FFFF	$\overline{\text{CS2}}$
0x20_0000–0x3F_FFFF	$\overline{\text{CS1}}$
0x40_0000–0x7F_FFFF	$\overline{\text{CS0}}$[1]

[1] When the internal NVM is enabled (see ROMON in Section 25.3.2.5, “MMC Control Register (MMCCTL1)) the $\overline{\text{CS0}}$ is not asserted in the space occupied by this on-chip memory block.

25.3.2.2 Mode Register (MODE)

Address: 0x000B PRR

	7	6	5	4	3	2	1	0
R	MODC	MODB	MODA	0	0	0	0	0
W				▨	▨	▨	▨	▨
Reset	MODC[1]	MODB[1]	MODA[1]	0	0	0	0	0

1. External signal (see Table 25-2).

▨ = Unimplemented or Reserved

Figure 25-4. Mode Register (MODE)

Read: Anytime. In emulation modes read operations will return the data read from the external bus. In all other modes the data are read from this register.

Write: Only if a transition is allowed (see Figure 25-5). In emulation modes write operations will be also directed to the external bus.

The MODE bits of the MODE register are used to establish the MCU operating mode.

CAUTION

XGATE write access to this register during an CPU access which makes use of this register could lead to unexpected results.

Table 25-7. MODE Field Descriptions

Field	Description
7–5 MODC, MODB, MODA	**Mode Select Bits** — These bits control the current operating mode during $\overline{\text{RESET}}$ high (inactive). The external mode pins MODC, MODB, and MODA determine the operating mode during $\overline{\text{RESET}}$ low (active). The state of the pins is latched into the respective register bits after the $\overline{\text{RESET}}$ signal goes inactive (see Figure 25-5). Write restrictions exist to disallow transitions between certain modes. Figure 25-5 illustrates all allowed mode changes. Attempting non authorized transitions will not change the MODE bits, but it will block further writes to these register bits except in special modes. Both transitions from normal single-chip mode to normal expanded mode and from emulation single-chip to emulation expanded mode are only executed by writing a value of 3'b101 (write once). Writing any other value will not change the MODE bits, but will block further writes to these register bits. Changes of operating modes are not allowed when the device is secured, but it will block further writes to these register bits except in special modes. In emulation modes reading this address returns data from the external bus which has to be driven by the emulator. It is therefore responsibility of the emulator hardware to provide the expected value (i.e. a value corresponding to normal single chip mode while the device is in emulation single-chip mode or a value corresponding to normal expanded mode while the device is in emulation expanded mode).

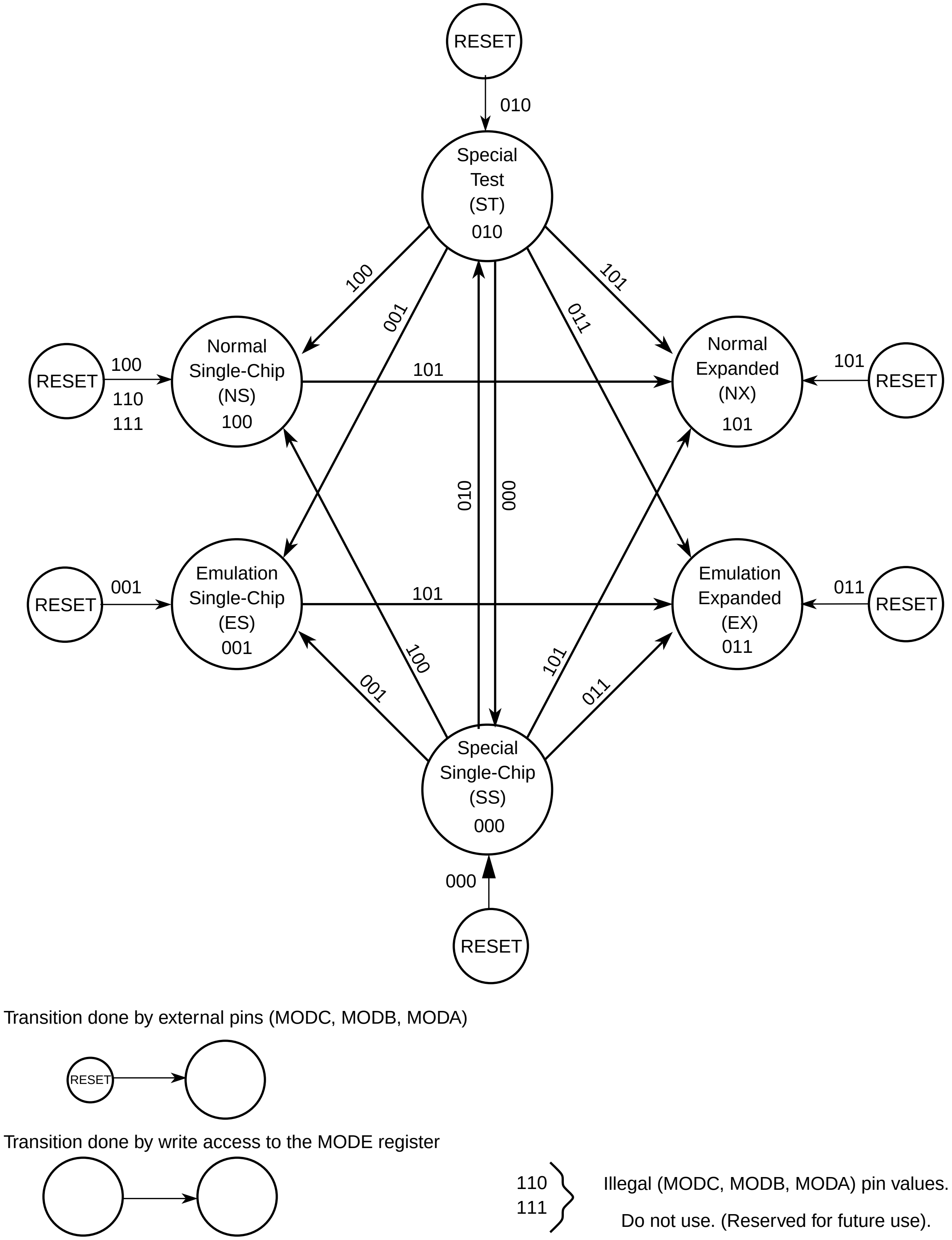


Figure 25-5. Mode Transition Diagram when MCU is Unsecured

25.3.2.3 Global Page Index Register (GPAGE)

Address: 0x0010

	7	6	5	4	3	2	1	0
R	0	GP6	GP5	GP4	GP3	GP2	GP1	GP0
W	▨							
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 25-6. Global Page Index Register (GPAGE)

Read: Anytime

Write: Anytime

The global page index register is used to construct a 23 bit address in the global map format. It is only used when the CPU is executing a global instruction (GLDAA, GLDAB, GLDD, GLDS, GLDX, GLDY,GSTAA, GSTAB, GSTD, GSTS, GSTX, GSTY) (see CPU Block Guide). The generated global address is the result of concatenation of the CPU local address [15:0] with the GPAGE register [22:16] (see Figure 25-7).

CAUTION

XGATE write access to this register during an CPU access which makes use of this register could lead to unexpected results.

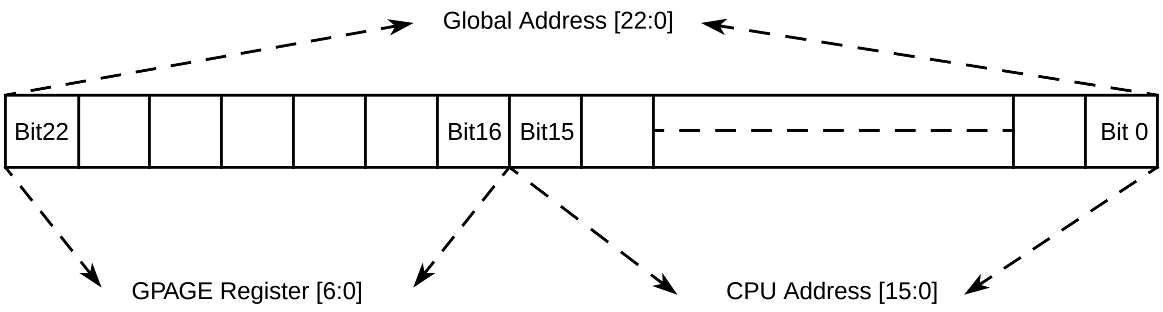


Figure 25-7. GPAGE Address Mapping

Table 25-8. GPAGE Field Descriptions

Field	Description
6–0 GP[6:0]	**Global Page Index Bits 6–0** — These page index bits are used to select which of the 128 64-kilobyte pages is to be accessed.

Example 25-1. This example demonstrates usage of the GPAGE register

```
LDX      #0x5000            ;Set GPAGE offset to the value of 0x5000
MOVB     #0x14, GPAGE       ;Initialize GPAGE register with the value of 0x14
GLDAA    X                  ;Load Accu A from the global address 0x14_5000
```

25.3.2.4 Direct Page Register (DIRECT)

Address: 0x0011

	7	6	5	4	3	2	1	0
R W	DP15	DP14	DP13	DP12	DP11	DP10	DP9	DP8
Reset	0	0	0	0	0	0	0	0

Figure 25-8. Direct Register (DIRECT)

Read: Anytime

Write: anytime in special modes, one time only in other modes.

This register determines the position of the 256 Byte direct page within the memory map.It is valid for both global and local mapping scheme.

Table 25-9. DIRECT Field Descriptions

Field	Description
7–0 DP[15:8]	**Direct Page Index Bits 15–8** — These bits are used by the CPU when performing accesses using the direct addressing mode. The bits from this register form bits [15:8] of the address (see Figure 25-9).

CAUTION

XGATE write access to this register during an CPU access which makes use of this register could lead to unexpected results.

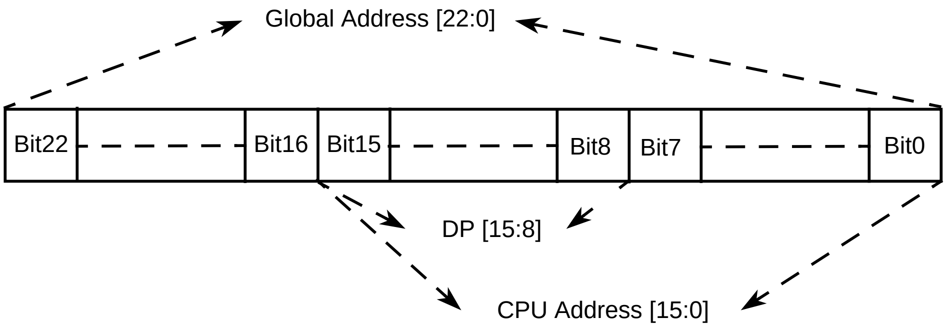


Figure 25-9. DIRECT Address Mapping

Bits [22:16] of the global address will be formed by the GPAGE[6:0] bits in case the CPU executes a global instruction in direct addressing mode or by the appropriate local address to the global address expansion (refer to Section 25.4.2.1.1, “Expansion of the Local Address Map).

Example 25-2. This example demonstrates usage of the Direct Addressing Mode

```
MOVB    #0x80,DIRECT        ;Set DIRECT register to 0x80. Write once only.
                            ;Global data accesses to the range 0xXX_80XX can be direct.
                            ;Logical data accesses to the range 0x80XX are direct.

LDY     <00                 ;Load the Y index register from 0x8000 (direct access).
                            ;< operator forces direct access on some assemblers but in
                            ;many cases assemblers are “direct page aware” and can
```

```
                    ;automatically select direct mode.
```

25.3.2.5 MMC Control Register (MMCCTL1)

Address: 0x0013 PRR

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	EROMON	ROMHM	ROMON
W	▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	EROMCTL	0	ROMCTL

▨ = Unimplemented or Reserved

Figure 25-10. MMC Control Register (MMCCTL1)

Read: Anytime. In emulation modes read operations will return the data from the external bus. In all other modes the data are read from this register.Write: Refer to each bit description. In emulation modes write operations will also be directed to the external bus.

CAUTION

XGATE write access to this register during an CPU access which makes use of this register could lead to unexpected results.

Table 25-10. MMCCTL1 Field Descriptions

Field	Description
2 EROMON	**Enables emulated Flash or ROM memory in the memory map** Write: Never This bit is used in some modes to define the placement of the Emulated Flash or ROM (Refer to Table 25-11) 0 Disables the emulated Flash or ROM in the memory map. 1 Enables the emulated Flash or ROM in the memory map.
1 ROMHM	**FLASH or ROM only in higher Half of Memory Map** Write: Once in normal and emulation modes and anytime in special modes 0 The fixed page of Flash or ROM can be accessed in the lower half of the memory map. Accesses to 0x4000–0x7FFF will be mapped to 0x7F_4000-0x7F_7FFF in the global memory space. 1 Disables access to the Flash or ROM in the lower half of the memory map.These physical locations of the Flash or ROM can still be accessed through the program page window. Accesses to 0x4000–0x7FFF will be mapped to 0x14_4000-0x14_7FFF in the global memory space (external access).
0 ROMON	**Enable FLASH or ROM in the memory map** Write: Once in normal and emulation modes and anytime in special modes. This bit is used in some modes to define the placement of the ROM (Refer to Table 25-11) 0 Disables the Flash or ROM from the memory map. 1 Enables the Flash or ROM in the memory map.

EROMON and ROMON control the visibility of the Flash in the memory map for CPU or BDM (not for XGATE). Both local and global memory maps are affected.

Table 25-11. Data Sources when CPU or BDM is Accessing Flash Area

Chip Modes	ROMON	EROMON	DATA SOURCE[1]	Stretch[2]
Normal Single Chip	X	X	Internal Flash	N
Special Single Chip				
Emulation Single Chip	X	0	Emulation Memory	N
	X	1	Internal Flash	
Normal Expanded	0	X	External Application	Y
	1	X	Internal Flash	N
Emulation Expanded	0	X	External Application	Y
	1	0	Emulation Memory	N
	1	1	Internal Flash	
Special Test	0	X	External Application	N
	1	X	Internal Flash	

[1] Internal Flash means Flash resources inside the MCU are read/written.
Emulation memory means resources inside the emulator are read/written (PRU registers, flash replacement, RAM, EEPROM and register space are always considered internal).
External application means resources residing outside the MCU are read/written.

[2] The external access stretch mechanism is part of the EBI module (refer to EBI Block Guide for details).

25.3.2.6 RAM Page Index Register (RPAGE)

Address: 0x0016

	7	6	5	4	3	2	1	0
R W	RP7	RP6	RP5	RP4	RP3	RP2	RP1	RP0
Reset	1	1	1	1	1	1	0	1

Figure 25-11. RAM Page Index Register (RPAGE)

Read: Anytime

Write: Anytime

These eight index bits are used to page 4 KByte blocks into the RAM page window located in the local (CPU or BDM) memory map from address 0x1000 to address 0x1FFF (see Figure 25-12). This supports accessing up to 1022 Kbytes of RAM (in the Global map) within the 64 KByte Local map. The RAM page index register is effectively used to construct paged RAM addresses in the Local map format.

CAUTION

> XGATE write access to this register during an CPU access which makes use of this register could lead to unexpected results.

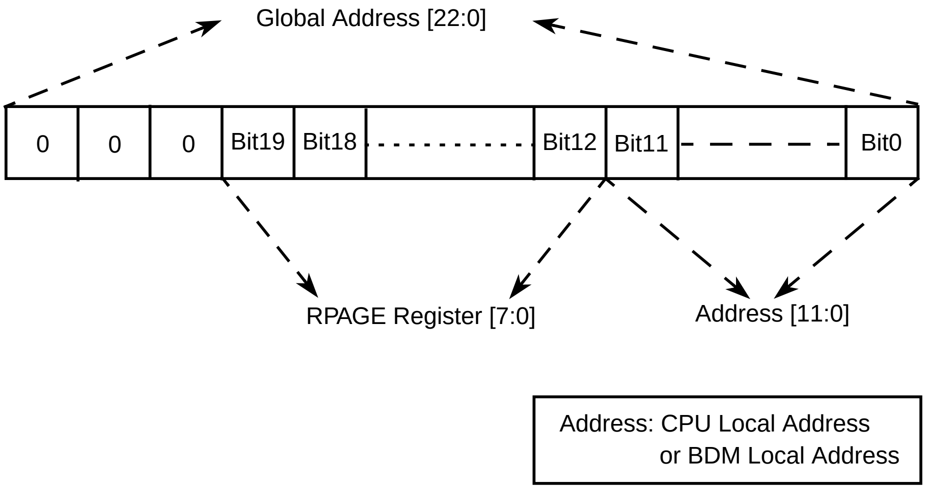


Figure 25-12. RPAGE Address Mapping

NOTE

Because RAM page 0 has the same global address as the register space, it is possible to write to registers through the RAM space when RPAGE = 0x00.

Table 25-12. RPAGE Field Descriptions

Field	Description
7–0 RP[7:0]	**RAM Page Index Bits 7–0** — These page index bits are used to select which of the 256 RAM array pages is to be accessed in the RAM Page Window.

The reset value of 0xFD ensures that there is a linear RAM space available between addresses 0x1000 and 0x3FFF out of reset.

The fixed 4K page from 0x2000–0x2FFF of RAM is equivalent to page 254 (page number 0xFE).

The fixed 4K page from 0x3000–0x3FFF of RAM is equivalent to page 255 (page number 0xFF).

25.3.2.7 EEPROM Page Index Register (EPAGE)

Address: 0x0017

	7	6	5	4	3	2	1	0
R W	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
Reset	1	1	1	1	1	1	1	0

Figure 25-13. EEPROM Page Index Register (EPAGE)

Read: Anytime

Write: Anytime

These eight index bits are used to page 1 KByte blocks into the EEPROM page window located in the local (CPU or BDM) memory map from address 0x0800 to address 0x0BFF (see Figure 25-14). This supports accessing up to 256 Kbytes of EEPROM (in the Global map) within the 64 KByte Local map. The EEPROM page index register is effectively used to construct paged EEPROM addresses in the Local map format.

CAUTION

XGATE write access to this register during an CPU access which makes use of this register could lead to unexpected results.

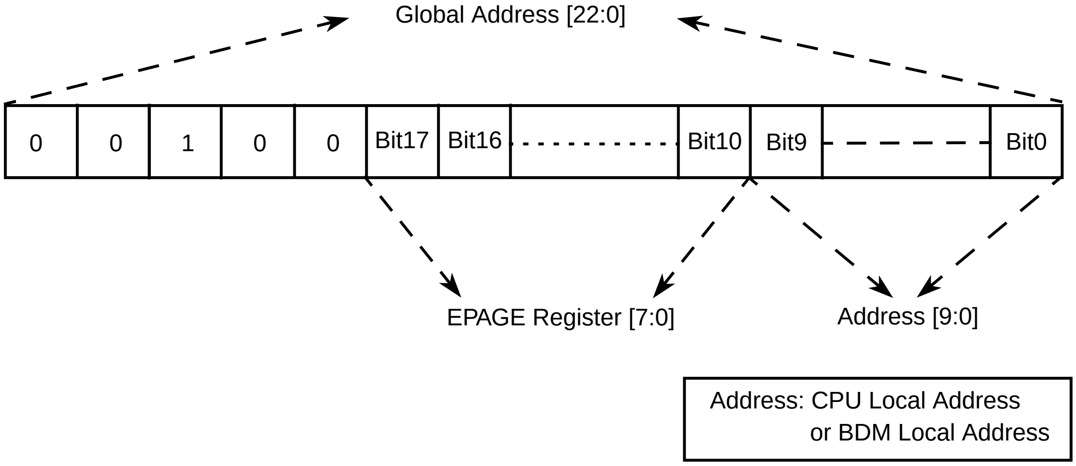


Figure 25-14. EPAGE Address Mapping

Table 25-13. EPAGE Field Descriptions

Field	Description
7–0 EP[7:0]	**EEPROM Page Index Bits 7–0** — These page index bits are used to select which of the 256 EEPROM array pages is to be accessed in the EEPROM Page Window.

The reset value of 0xFE ensures that there is a linear EEPROM space available between addresses 0x0800 and 0x0FFF out of reset.

The fixed 1K page 0x0C00–0x0FFF of EEPROM is equivalent to page 255 (page number 0xFF).

25.3.2.8 Program Page Index Register (PPAGE)

Address: 0x0030

	7	6	5	4	3	2	1	0
R W	PIX7	PIX6	PIX5	PIX4	PIX3	PIX2	PIX1	PIX0
Reset	1	1	1	1	1	1	1	0

Figure 25-15. Program Page Index Register (PPAGE)

Read: Anytime

Write: Anytime

These eight index bits are used to page 16 KByte blocks into the Flash page window located in the local (CPU or BDM) memory map from address 0x8000 to address 0xBFFF (see Figure 25-16). This supports accessing up to 4 Mbytes of Flash (in the Global map) within the 64 KByte Local map. The PPAGE age index register is effectively used to construct paged Flash addresses in the Local map format. The CPU has special access to read and write this register directly during execution of CALL and RTC instructions. .

CAUTION

XGATE write access to this register during an CPU access which makes use of this register could lead to unexpected results.

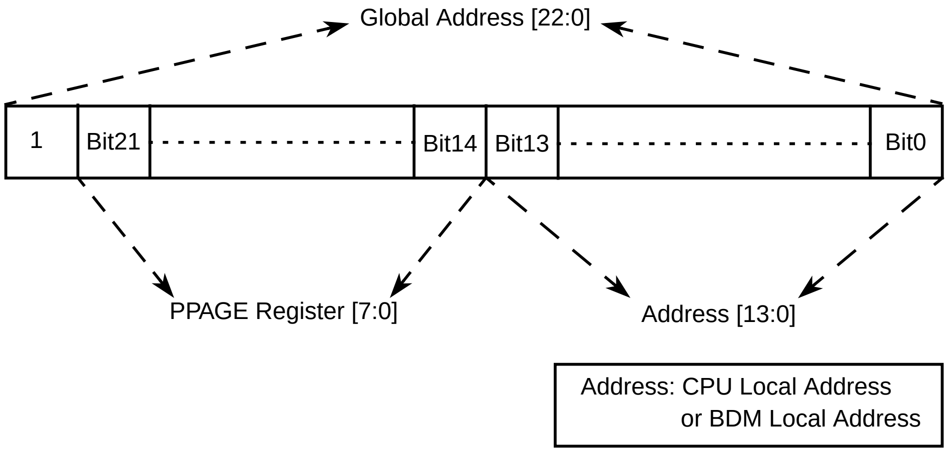


Figure 25-16. PPAGE Address Mapping

NOTE

Writes to this register using the special access of the CALL and RTC instructions will be complete before the end of the instruction execution.

Table 25-14. PPAGE Field Descriptions

Field	Description
7–0 PIX[7:0]	**Program Page Index Bits 7–0** — These page index bits are used to select which of the 256 FLASH or ROM array pages is to be accessed in the Program Page Window.

The fixed 16K page from 0x4000–0x7FFF (when ROMHM = 0) is the page number 0xFD.

The reset value of 0xFE ensures that there is linear Flash space available between addresses 0x4000 and 0xFFFF out of reset.

The fixed 16K page from 0xC000-0xFFFF is the page number 0xFF.

25.3.2.9 RAM Write Protection Control Register (RAMWPC)

Address: 0x011C

	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 25-17. RAM Write Protection Control Register (RAMWPC)

Read: Anytime

Write: Anytime

Table 25-15. RAMWPC Field Descriptions

Field	Description
0 RWPE	**RAM Write Protection Enable** — This bit enables the RAM write protection mechanism. When the RWPE bit is cleared, there is no write protection and any memory location is writable by the CPU module and the XGATE module. When the RWPE bit is set the write protection mechanism is enabled and write access of the CPU or to the XGATE RAM region. Write access performed by the XGATE module to outside of the XGATE RAM region or the shared region is suppressed as well in this case. 0 RAM write protection check is disabled, region boundary registers can be written. 1 RAM write protection check is enabled, region boundary registers cannot be written.
1 AVIE	**CPU Access Violation Interrupt Enable** — This bit enables the Access Violation Interrupt. If AVIE is set and AVIF is set, an interrupt is generated. 0 CPU Access Violation Interrupt Disabled. 1 CPU Access Violation Interrupt Enabled.
0 AVIF	**CPU Access Violation Interrupt Flag** — When set, this bit indicates that the CPU has tried to write a memory location inside the XGATE RAM region. This flag can be reset by writing ’1’ to the AVIF bit location. 0 No access violation by the CPU was detected. 1 Access violation by the CPU was detected.

25.3.2.10 RAM XGATE Upper Boundary Register (RAMXGU)

Address: 0x011D

	7	6	5	4	3	2	1	0
R	1	XGU6	XGU5	XGU4	XGU3	XGU2	XGU1	XGU0
W	▨							
Reset	1	1	1	1	1	1	1	1

▨ = Unimplemented or Reserved

Figure 25-18. RAM XGATE Upper Boundary Register (RAMXGU)

Read: Anytime

Write: Anytime when RWPE = 0

Table 25-16. RAMXGU Field Descriptions

Field	Description
6–0 XGU[6:0]	**XGATE Region Upper Boundary Bits 6-0** — These bits define the upper boundary of the RAM region allocated to the XGATE module in multiples of 256 bytes. The 256 byte block selected by this register is included in the region. See Figure 25-25 for details.

25.3.2.11 RAM Shared Region Lower Boundary Register (RAMSHL)

Address: 0x011E

	7	6	5	4	3	2	1	0
R	1	SHL6	SHL5	SHL4	SHL3	SHL2	SHL1	SHL0
W	▨							
Reset	1	1	1	1	1	1	1	1

▨ = Unimplemented or Reserved

Figure 25-19. RAM Shared Region Lower Boundary Register (RAMSHL)

Read: Anytime

Write: Anytime when RWPE = 0

Table 25-17. RAMSHL Field Descriptions

Field	Description
6–0 SHL[6:0]	**RAM Shared Region Lower Boundary Bits 6–0** — These bits define the lower boundary of the shared memory region in multiples of 256 bytes. The block selected by this register is included in the region. See Figure 25-25 for details.

25.3.2.12 RAM Shared Region Upper Boundary Register (RAMSHU)

Address: 0x011F

	7	6	5	4	3	2	1	0
R	1	SHU6	SHU5	SHU4	SHU3	SHU2	SHU1	SHU0
W	▨							
Reset	1	1	1	1	1	1	1	1

▨ = Unimplemented or Reserved

Figure 25-20. RAM Shared Region Upper Boundary Register (RAMSHU)

Read: Anytime

Write: Anytime when RWPE = 0

Table 25-18. RAMSHU Field Descriptions

Field	Description
6–0 SHU[6:0]	**RAM Shared Region Upper Boundary Bits 6–0** — These bits define the upper boundary of the shared memory in multiples of 256 bytes. The block selected by this register is included in the region. See Figure 25-25 for details.

25.4 Functional Description

The MMC block performs several basic functions of the S12X sub-system operation: MCU operation modes, priority control, address mapping, select signal generation and access limitations for the system. Each aspect is described in the following subsections.

25.4.1 MCU Operating Mode

- Normal single-chip mode

  There is no external bus in this mode. The MCU program is executed from the internal memory and no external accesses are allowed.
- Special single-chip mode

  This mode is generally used for debugging single-chip operation, boot-strapping or security related operations. The active background debug mode is in control of the CPU code execution and the BDM firmware is waiting for serial commands sent through the BKGD pin. There is no external bus in this mode.
- Emulation single-chip mode

  Tool vendors use this mode for emulation systems in which the user's target application is normal single-chip mode. Code is executed from external or internal memory depending on the set-up of the EROMON bit (see Section 25.3.2.5, "MMC Control Register (MMCCTL1)). The external bus is active in both cases to allow observation of internal operations (internal visibility).

- Normal expanded mode

  The external bus interface is configured as an up to 23-bit address bus, 8 or 16-bit data bus with dedicated bus control and status signals. This mode allows 8 or 16-bit external memory and peripheral devices to be interfaced to the system. The fastest external bus rate is half of the internal bus rate. An external signal can be used in this mode to cause the external bus to wait as desired by the external logic.
- Emulation expanded mode

  Tool vendors use this mode for emulation systems in which the user's target application is normal expanded mode.
- Special test mode

  This mode is an expanded mode for factory test.

25.4.2 Memory Map Scheme

25.4.2.1 CPU and BDM Memory Map Scheme

The BDM firmware lookup tables and BDM register memory locations share addresses with other modules; however they are not visible in the memory map during user's code execution. The BDM memory resources are enabled only during the READ_BD and WRITE_BD access cycles to distinguish between accesses to the BDM memory area and accesses to the other modules. (Refer to BDM Block Guide for further details).

When the MCU enters active BDM mode, the BDM firmware lookup tables and the BDM registers become visible in the local memory map in the range 0xFF00-0xFFFF (global address 0x7F_FF00 - 0x7F_FFFF) and the CPU begins execution of firmware commands or the BDM begins execution of hardware commands. The resources which share memory space with the BDM module will not be visible in the memory map during active BDM mode.

Please note that after the MCU enters active BDM mode the BDM firmware lookup tables and the BDM registers will also be visible between addresses 0xBF00 and 0xBFFF if the PPAGE register contains value of 0xFF.

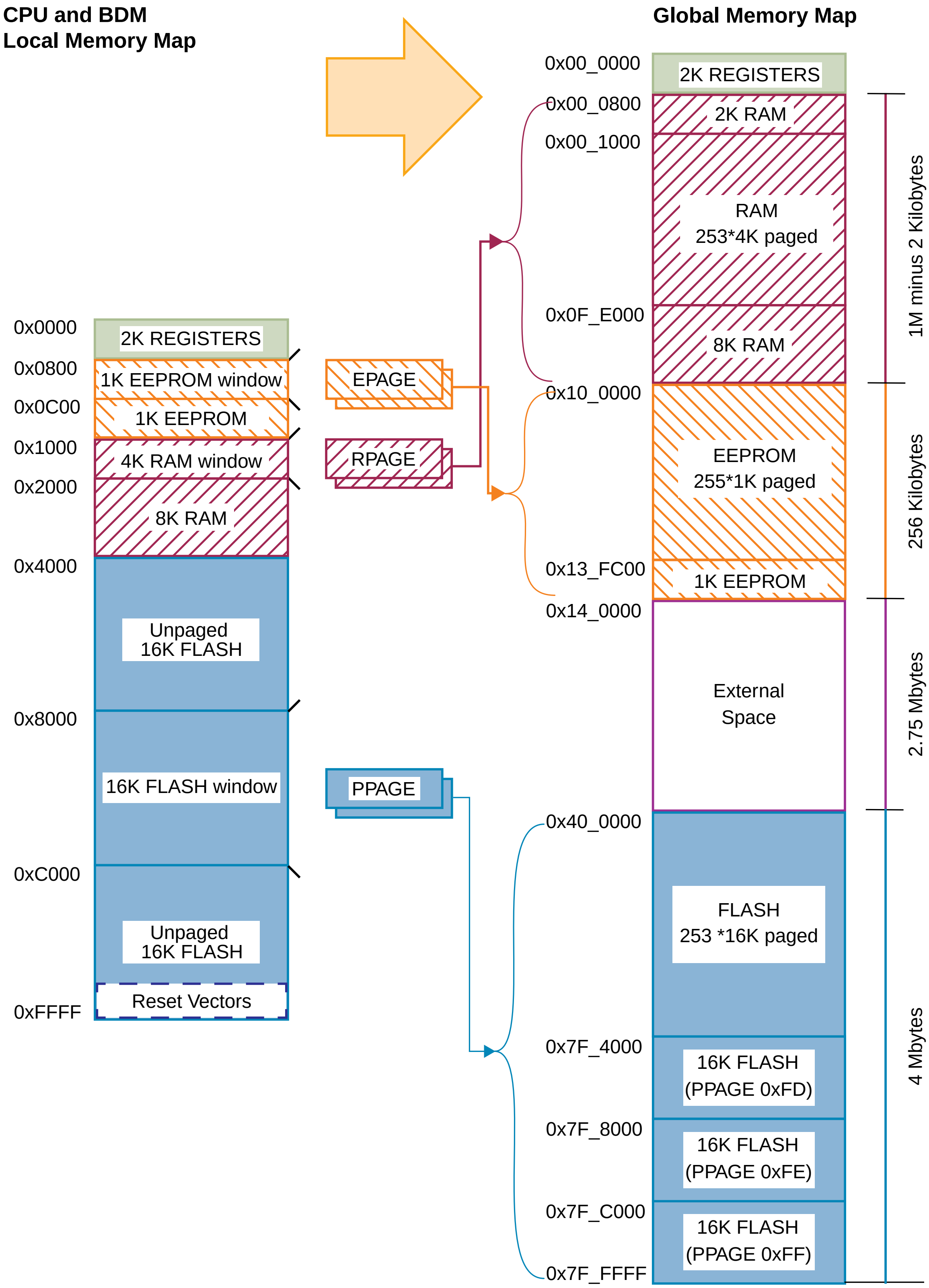


Figure 25-21. Expansion of the Local Address Map

25.4.2.1.1 Expansion of the Local Address Map

Expansion of the CPU Local Address Map

The program page index register in MMC allows accessing up to 4 Mbyte of FLASH or ROM in the global memory map by using the eight page index bits to page 256 16 Kbyte blocks into the program page window located from address 0x8000 to address 0xBFFF in the local CPU memory map.

The page value for the program page window is stored in the PPAGE register. The value of the PPAGE register can be read or written by normal memory accesses as well as by the CALL and RTC instructions (see Section 25.5.1, “CALL and RTC Instructions).

Control registers, vector space and parts of the on-chip memories are located in unpaged portions of the 64-kilobyte local CPU address space.

The starting address of an interrupt service routine must be located in unpaged memory unless the user is certain that the PPAGE register will be set to the appropriate value when the service routine is called. However an interrupt service routine can call other routines that are in paged memory. The upper 16-kilobyte block of the local CPU memory space (0xC000–0xFFFF) is unpaged. It is recommended that all reset and interrupt vectors point to locations in this area or to the other unpaged sections of the local CPU memory map.

Table 25-19 Table 25-12 summarizes mapping of the address bus in Flash/External space based on the address, the PPAGE register value and value of the ROMHM bit in the MMCCTL1 register.

Table 25-19. Global FLASH/ROM Allocated

Local CPU Address	ROMHM	External Access	Global Address
0x4000–0x7FFF	0	No	0x7F_4000 –0x7F_7FFF
	1	Yes	0x14_4000–0x14_7FFF
0x8000–0xBFFF	N/A	No[1]	0x40_0000–0x7F_FFFF
	N/A	Yes[1]	
0xC000–0xFFFF	N/A	No	0x7F_C000–0x7F_FFFF

[1] The internal or the external bus is accessed based on the size of the memory resources implemented on-chip. Please refer to Figure 1-23 for further details.

The RAM page index register allows accessing up to 1 Mbyte –2 Kbytes of RAM in the global memory map by using the eight RPAGE index bits to page 4 Kbyte blocks into the RAM page window located in the local CPU memory space from address 0x1000 to address 0x1FFF. The EEPROM page index register EPAGE allows accessing up to 256 Kbytes of EEPROM in the system by using the eight EPAGE index bits to page 1 Kbyte blocks into the EEPROM page window located in the local CPU memory space from address 0x0800 to address 0x0BFF.

Expansion of the BDM Local Address Map

PPAGE, RPAGE, and EPAGE registers are also used for the expansion of the BDM local address to the global address. These registers can be read and written by the BDM.

The BDM expansion scheme is the same as the CPU expansion scheme.

25.4.2.2 Global Addresses Based on the Global Page

CPU Global Addresses Based on the Global Page

The seven global page index bits allow access to the full 8 Mbyte address map that can be accessed with 23 address bits. This provides an alternative way to access all of the various pages of FLASH, RAM and EEPROM as well as additional external memory.

The GPAGE Register is used only when the CPU is executing a global instruction (see Section 25.3.2.3, “Global Page Index Register (GPAGE)). The generated global address is the result of concatenation of the CPU local address [15:0] with the GPAGE register [22:16] (see Figure 25-7).

BDM Global Addresses Based on the Global Page

The seven BDMGPR Global Page index bits allow access to the full 8 Mbyte address map that can be accessed with 23 address bits. This provides an alternative way to access all of the various pages of FLASH, RAM and EEPROM as well as additional external memory.

The BDM global page index register (BDMGPR) is used only in the case the CPU is executing a firmware command which uses a global instruction (like GLDD, GSTD) or by a BDM hardware command (like WRITE_W, WRITE_BYTE, READ_W, READ_BYTE). See the BDM Block Guide for further details.

The generated global address is a result of concatenation of the BDM local address with the BDMGPR register [22:16] in the case of a hardware command or concatenation of the CPU local address and the BDMGPR register [22:16] in the case of a firmware command (see Figure 25-22).

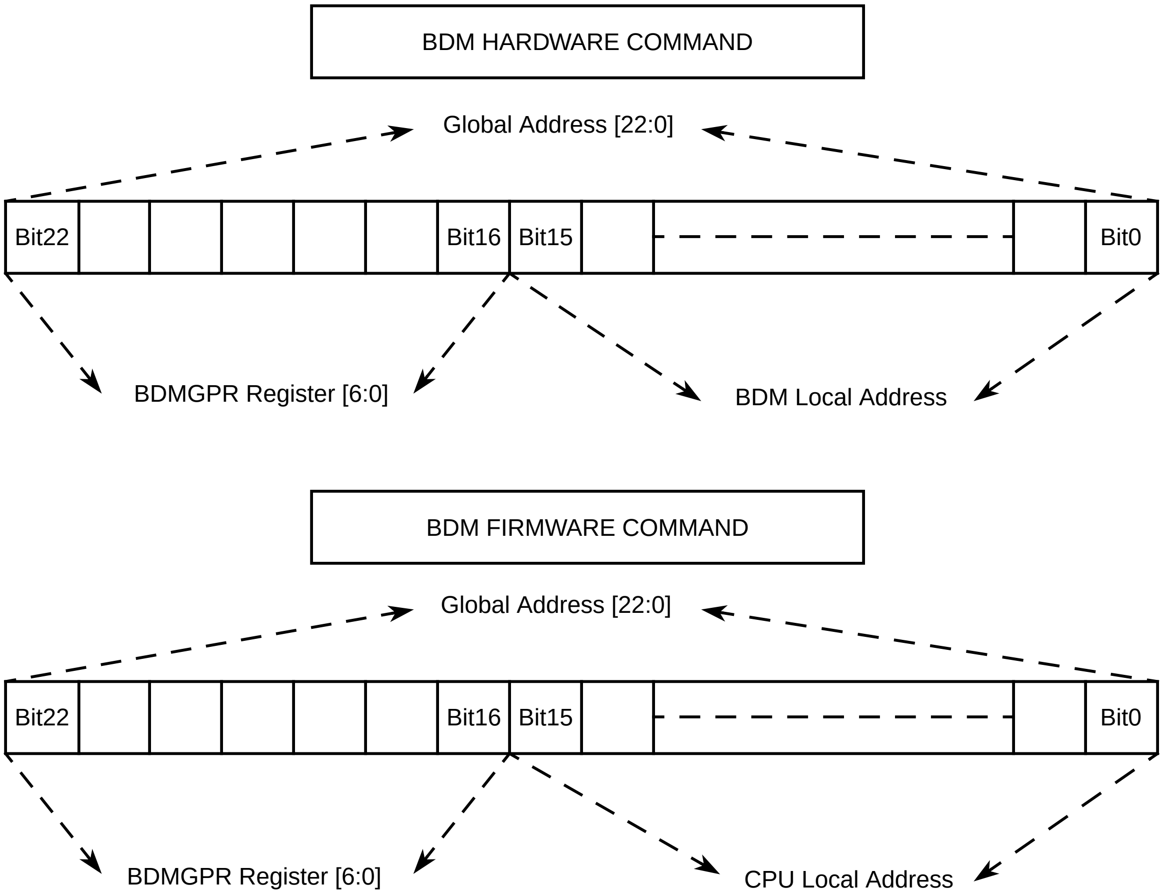


Figure 25-22. BDMGPR Address Mapping

25.4.2.3 Implemented Memory Map

The global memory spaces reserved for the internal resources (RAM, EEPROM, and FLASH) are not determined by the MMC module. Size of the individual internal resources are however fixed in the design of the device cannot be changed by the user. Please refer to the Device User Guide for further details. Figure 25-23 and Table 25-20 show the memory spaces occupied by the on-chip resources. Please note that the memory spaces have fixed top addresses.

Table 25-20. Global Implemented Memory Space

Internal Resource	$Address
RAM	RAM_LOW = 0x10_0000 minus RAMSIZE[1]
EEPROM	EEPROM_LOW = 0x14_0000 minus EEPROMSIZE[2]
FLASH0[3]	FLASH0_LOW = 0x77_FFFF plus FLASHSIZE0
FLASH1	FLASH1_HIGH = 0x80_0000 minus FLASHSIZE1

[1] RAMSIZE is the hexadecimal value of RAM SIZE in bytes

[2] EEPROMSIZE is the hexadecimal value of EEPROM SIZE in bytes

[3] Internal FLASH SIZE (FLASHSIZE) is the sum of FLASHSIZE0 and FLASHSIZE1.

When the device is operating in expanded modes except emulation single-chip mode, accesses to global addresses which are not occupied by the on-chip resources (unimplemented areas or external memory space) result in accesses to the external bus (see Figure 25-23).

In emulation single-chip mode, accesses to global addresses which are not occupied by the on-chip resources (unimplemented areas) result in accesses to the external bus. CPU accesses to global addresses which are occupied by external memory space result in an illegal access reset (system reset). BDM accesses to the external space are performed but the data will be undefined.

In single-chip modes accesses by the CPU (except for firmware commands) to any of the unimplemented areas (see Figure 25-23) will result in an illegal access reset (system reset). BDM accesses to the unimplemented areas are allowed but the data will be undefined.

No misaligned word access from the BDM module will occur; these accesses are blocked in the BDM module (Refer to BDM Block Guide).

Misaligned word access to the last location of RAM is performed but the data will be undefined.

Misaligned word access to the last location of any global page (64 Kbyte) by any global instruction, is performed by accessing the last byte of the page and the first byte of the same page, considering the above mentioned misaligned access cases.

The non-internal resources (unimplemented areas or external space) are used to generate the chip selects (CS0,CS1,CS2 and CS3) (see Figure 25-23), which are only active in normal expanded, emulation expanded and special test modes (see Section 25.3.2.1, “MMC Control Register (MMCCTL0)).

Table 25-21 shows the address boundaries of each chip select and the relationship with the implemented resources (internal) parameters.

Table 25-21. Global Chip Selects Memory Space

Chip Selects	Bottom Address	Top Address
$\overline{\text{CS3}}$	0x00_0800	0x0F_FFFF minus RAMSIZE[1]
$\overline{\text{CS2}}$	0x10_0000	0x13_FFFF minus EEPROMSIZE[2]
$\overline{\text{CS2}}$[3]	0x14_0000	0x1F_FFFF
$\overline{\text{CS1}}$	0x20_0000	0x3F_FFFF
$\overline{\text{CS0}}$[4]	0x40_0000	0x7F_FFFF minus FLASHSIZE[5]

[1] External RPAGE accesses in (NX, EX and ST)

[2] External EPAGE accesses in (NX, EX and ST)

[3] When ROMHM is set (see ROMHM in Table 25-19) the $\overline{\text{CS2}}$ is asserted in the space occupied by this on-chip memory block.

[4] When the internal NVM is enabled (see ROMON in Section 25.3.2.5, "MMC Control Register (MMCCTL1)) the $\overline{\text{CS0}}$ is not asserted in the space occupied by this on-chip memory block.

[5] External PPAGE accesses in (NX, EX and ST)

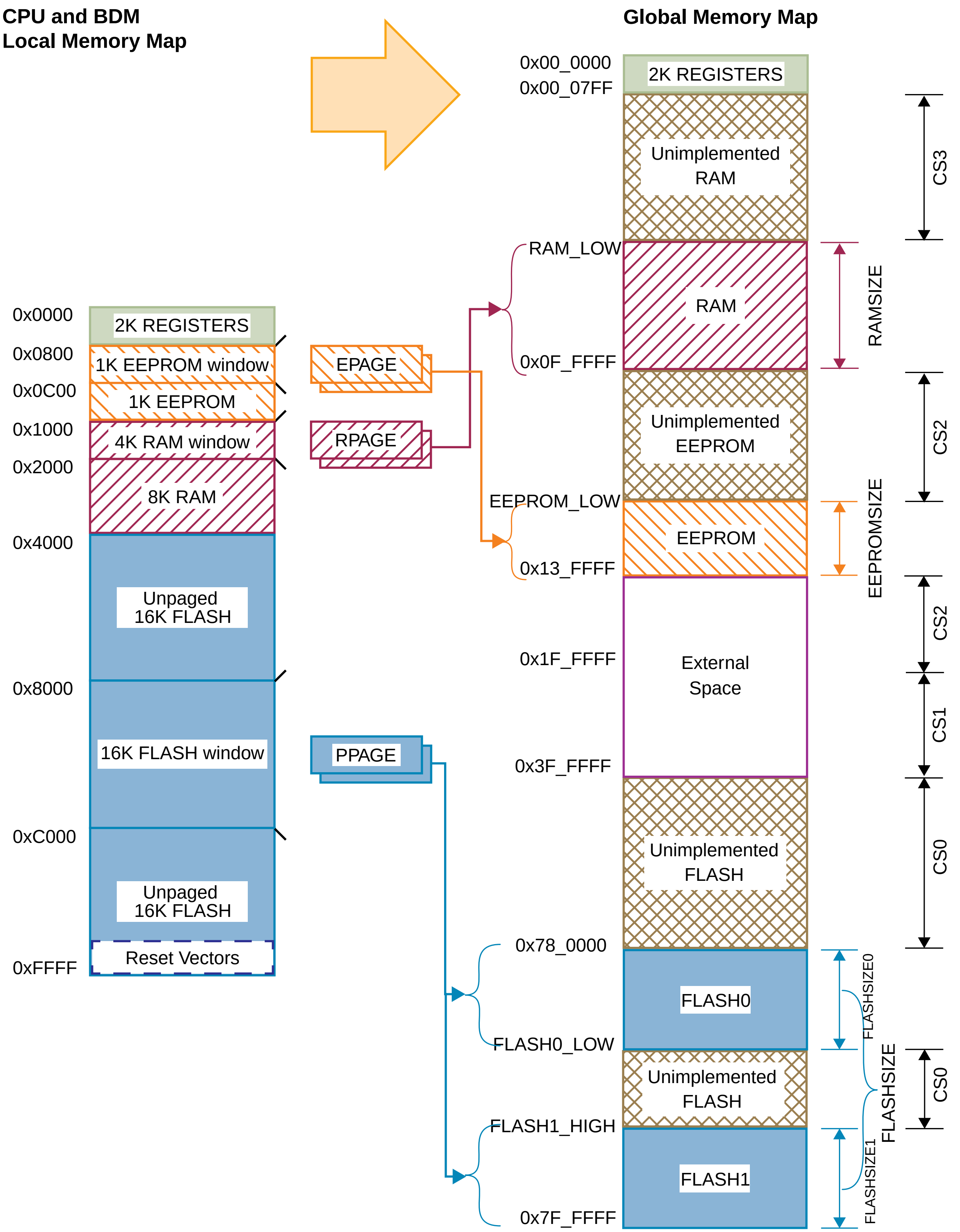


Figure 25-23. Local to Implemented Global Address Mapping (Without GPAGE)

25.4.2.4 XGATE Memory Map Scheme

25.4.2.4.1 Expansion of the XGATE Local Address Map

The XGATE 64 Kbyte memory space allows access to internal resources only (Registers, RAM, and FLASH). The 2 Kilobyte register address range is the same register address range as for the CPU and the BDM module (see Table 25-22).

XGATE can access the FLASH in single chip modes, even when the MCU is secured. In expanded modes, XGATE can not access the FLASH when MCU is secured.

The local address of the XGATE RAM access is translated to the global RAM address range. The XGATE shares the RAM resource with the CPU and the BDM module (see Table 25-22).

XGATE RAM size (XGRAMSIZE) may be lower or equal to the MCU RAM size (RAMSIZE).

The local address of the XGATE FLASH access is translated to the global address as defined by Table 25-22.

Table 25-22. XGATE Implemented Memory Space

Internal Resource	$Address
XGATE RAM	XGRAM_LOW = 0x0F_0000 plus (0x1_0000 minus XGRAMSIZE)[1]
XGATE FLASH	XGFLASH_HIGH = 0x78_0000 plus (0xFFFF minus XGRAMSIZE)

[1] XGRAMSIZE is the hexadecimal value of XGATE RAM SIZE in bytes.

Example 25-3. is a general example of the XGATE memory map implementation.

Example 25-3.

The MCU FLASHSIZE is 64 Kbytes (0x10000) and MCU RAMSIZE is 32 Kbytes (0x8000).

The XGATE RAMSIZE is 16 Kbytes (0x4000).

The space occupied by the XGATE RAM in the global address space will be:

Bottom address: (0x10_0000 minus 0x4000) = 0x0F_C000

Top address: 0x0F_FFFF

XGATE accesses to local address range 0x0800–0xBFFF will result in accesses to the following FLASH block in the global address space:

Bottom address: 0x78_0800

Top address: (0x78_FFFF minus 0x4000) = 0x78_BFFF

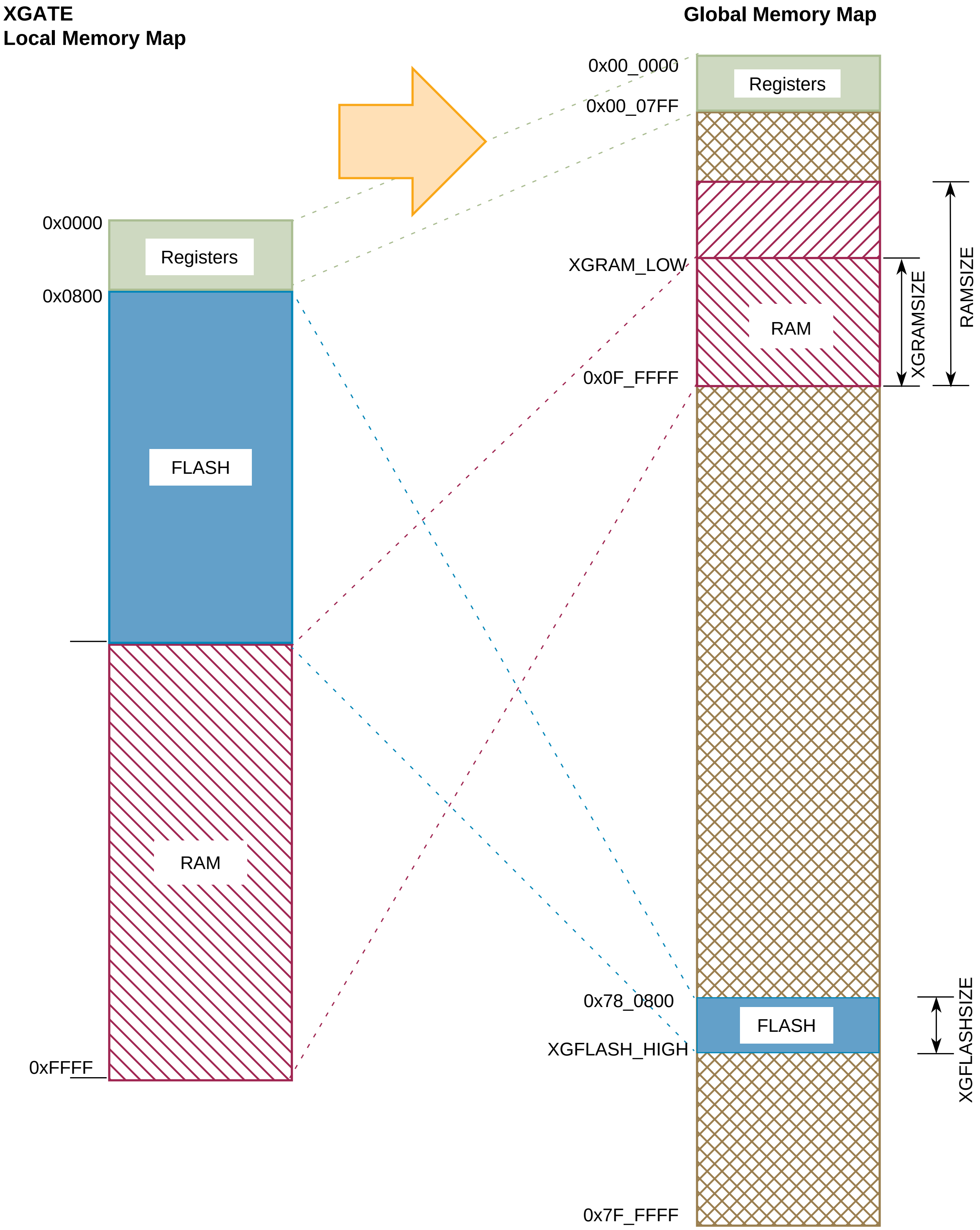


Figure 25-24. XGATE Global Address Mapping

25.4.3 Chip Access Restrictions

25.4.3.1 Illegal XGATE Accesses

A possible access error is flagged by the MMC and signalled to XGATE under the following conditions:

- XGATE performs misaligned word (in case of load-store or opcode or vector fetch accesses).
- XGATE accesses the register space (in case of opcode or vector fetch).
- XGATE performs a write to Flash in any modes (in case of load-store access).
- XGATE performs an access to a secured Flash in expanded modes (in case of load-store or opcode or vector fetch accesses).
- XGATE performs an access to an unimplemented area (in case of load-store or opcode or vector fetch accesses).
- XGATE performs a write to non-XGATE region in RAM (RAM protection mechanism) (in case of load-store access).

For further details refer to the XGATE Block Guide.

25.4.3.2 Illegal CPU Accesses

After programming the protection mechanism registers (see Figure 25-17, Figure 25-18, Figure 25-19, and Figure 25-20) and setting the RWPE bit (see Figure 25-17) there are 3 regions recognized by the MMC module:

1. XGATE RAM region
2. CPU RAM region
3. Shared Region (XGATE AND CPU)

If the RWPE bit is set the CPU write accesses into the XGATE RAM region are blocked. If the CPU tries to write the XGATE RAM region the AVIF bit is set and an interrupt is generated if enabled. Furthermore if the XGATE tries to write to outside of the XGATE RAM or shared regions and the RWPE bit is set, the write access is suppressed and the access error will be flagged to the XGATE module (see Section 25.4.3.1, “Illegal XGATE Accesses and the XGATE Block Guide).

The bottom address of the XGATE RAM region always starts at the lowest implemented RAM address.

The values stored in the boundary registers define the boundary addresses in 256 byte steps. The 256 byte block selected by any of the registers is always included in the respective region. For example setting the shared region lower boundary register (RAMSHL) to 0xC1 and the shared region upper boundary register (RAMSHU) to 0xE0 defines the shared region from address 0x0F_C100 to address 0x0F_E0FF in the global memory space (see Figure 25-20).

The interrupt requests generated by the MMC are listed in Table 25-23. Refer to the Device User Guide for the related interrupt vector address and interrupt priority.

The following conditions must be satisfied to ensure correct operation of the RAM protection mechanism:

- Value stored in RAMXGU must be lower than the value stored in RAMSHL.
- Value stored RAMSHL must be lower or equal than the value stored in RAMSHU.

Table 25-23. RAM Write Protection Interrupt Vectors

Interrupt Source	CCR Mask	Local Enable
CPU access violation	I Bit	AVIE in RAMWPC

Figure 25-25. RAM write protection scheme

25.4.4 Chip Bus Control

The MMC controls the address buses and the data buses that interface the S12X masters (CPU, BDMand XGATE) with the rest of the system (master buses). In addition the MMC handles all CPU read data bus swapping operations. All internal and external resources are connected to specific target buses (see Figure 25-26[1]).

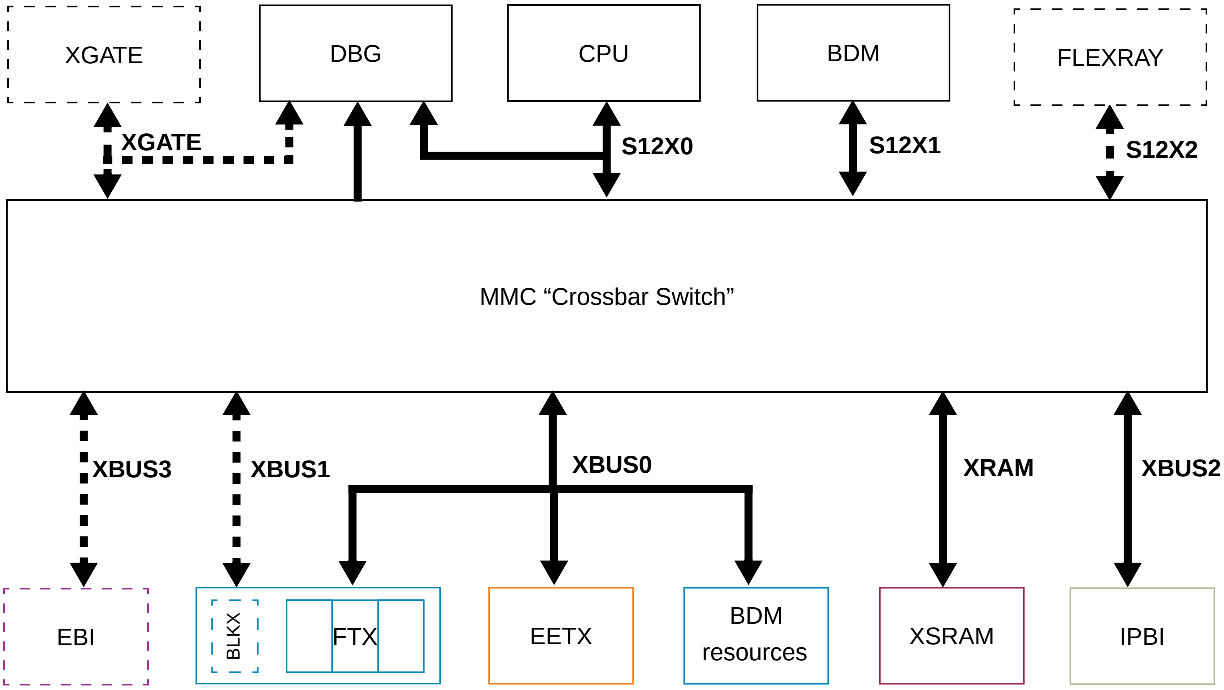


Figure 25-26. MMC Block Diagram

---

[1] Doted blocks and lines are optional. Please refer to the Device User Guide for their availlibilities.

25.4.4.1 Master Bus Prioritization regarding Access Conflicts on Target Buses

The arbitration scheme allows only one master to be connected to a target at any given time. The following rules apply when prioritizing accesses from different masters to the same target bus:

- CPU always has priority over BDM and XGATE.
- XGATE access to PRU registers constitutes a special case. It is always granted and stalls the CPU for its duration.
- XGATE has priority over BDM.
- BDM has priority over CPU and XGATE when its access is stalled for more than 128 cycles. In the later case the suspect master will be stalled after finishing the current operation and the BDM will gain access to the bus.
- In emulation modes all internal accesses are visible on the external bus as well and the external bus is used during access to the PRU registers.

25.4.5 Interrupts

25.4.5.1 Outgoing Interrupt Requests

The following interrupt requests can be triggered by the MMC module:

CPU access violation: The CPU access violation signals to the CPU detection of an error condition in the CPU application code which is resulted in write access to the protected XGATE RAM area (see Section 25.4.3.2, “Illegal CPU Accesses).

25.5 Initialization/Application Information

25.5.1 CALL and RTC Instructions

CALL and RTC instructions are uninterruptable CPU instructions that automate page switching in the program page window. The CALL instruction is similar to the JSR instruction, but the subroutine that is called can be located anywhere in the local address space or in any Flash or ROM page visible through the program page window. The CALL instruction calculates and stacks a return address, stacks the current PPAGE value and writes a new instruction-supplied value to the PPAGE register. The PPAGE value controls which of the 256 possible pages is visible through the 16 Kbyte program page window in the 64 Kbyte local CPU memory map. Execution then begins at the address of the called subroutine.

During the execution of the CALL instruction, the CPU performs the following steps:

1. Writes the current PPAGE value into an internal temporary register and writes the new instruction-supplied PPAGE value into the PPAGE register
2. Calculates the address of the next instruction after the CALL instruction (the return address) and pushes this 16-bit value onto the stack
3. Pushes the temporarily stored PPAGE value onto the stack
4. Calculates the effective address of the subroutine, refills the queue and begins execution at the new address

This sequence is uninterruptable. There is no need to inhibit interrupts during the CALL instruction execution. A CALL instruction can be performed from any address to any other address in the local CPU memory space.

The PPAGE value supplied by the instruction is part of the effective address of the CPU. For all addressing mode variations (except indexed-indirect modes) the new page value is provided by an immediate operand in the instruction. In indexed-indirect variations of the CALL instruction a pointer specifies memory locations where the new page value and the address of the called subroutine are stored. Using indirect addressing for both the new page value and the address within the page allows usage of values calculated at run time rather than immediate values that must be known at the time of assembly.

The RTC instruction terminates subroutines invoked by a CALL instruction. The RTC instruction unstacks the PPAGE value and the return address and refills the queue. Execution resumes with the next instruction after the CALL instruction.

During the execution of an RTC instruction the CPU performs the following steps:

1. Pulls the previously stored PPAGE value from the stack
2. Pulls the 16-bit return address from the stack and loads it into the PC
3. Writes the PPAGE value into the PPAGE register
4. Refills the queue and resumes execution at the return address

This sequence is uninterruptable. The RTC can be executed from anywhere in the local CPU memory space.

The CALL and RTC instructions behave like JSR and RTS instruction, they however require more execution cycles. Usage of JSR/RTS instructions is therefore recommended when possible and CALL/RTC instructions should only be used when needed. The JSR and RTS instructions can be used to access subroutines that are already present in the local CPU memory map (i.e. in the same page in the program memory page window for example). However calling a function located in a different page requires usage of the CALL instruction. The function must be terminated by the RTC instruction. Because the RTC instruction restores contents of the PPAGE register from the stack, functions terminated with the RTC instruction must be called using the CALL instruction even when the correct page is already present in the memory map. This is to make sure that the correct PPAGE value will be present on stack at the time of the RTC instruction execution.

25.5.2 Port Replacement Registers (PRRs)

Registers used for emulation purposes must be rebuilt by the in-circuit emulator hardware to achieve full emulation of single chip mode operation. These registers are called port replacement registers (PRRs) (see Table 1-25). PRRs are accessible from CPU, BDM and XGATE using different access types (word aligned, word-misaligned and byte).

Each access to PRRs will be extended to 2 bus cycles for write or read accesses independent of the operating mode. In emulation modes all write operations result in simultaneous writing to the internal registers (peripheral access) and to the emulated registers (external access) located in the PRU in the emulator. All read operations are performed from external registers (external access) in emulation modes. In all other modes the read operations are performed from the internal registers (peripheral access).

Due to internal visibility of CPU accesses the CPU will be halted during XGATE or BDM access to any PRR. This rule applies also in normal modes to ensure that operation of the device is the same as in emulation modes.

A summary of PRR accesses:

- An aligned word access to a PRR will take 2 bus cycles.
- A misaligned word access to a PRRs will take 4 cycles. If one of the two bytes accessed by the misaligned word access is not a PRR, the access will take only 3 cycles.
- A byte access to a PRR will take 2 cycles.

Table 25-24. PRR Listing

PRR Name	PRR Local Address	PRR Location
PORTA	0x0000	PIM
PORTB	0x0001	PIM
DDRA	0x0002	PIM
DDRB	0x0003	PIM
PORTC	0x0004	PIM
PORTD	0x0005	PIM
DDRC	0x0006	PIM
DDRD	0x0007	PIM
PORTE	0x0008	PIM
DDRE	0x0009	PIM
MMCCTL0	0x000A	MMC
MODE	0x000B	MMC
PUCR	0x000C	PIM
RDRIV	0x000D	PIM
EBICTL0	0x000E	EBI
EBICTL1	0x000F	EBI
Reserved	0x0012	MMC
MMCCTL1	0x0013	MMC
ECLKCTL	0x001C	PIM
Reserved	0x001D	PIM
PORTK	0x0032	PIM
DDRK	0x0033	PIM

25.5.3 On-Chip ROM Control

The MCU offers two modes to support emulation. In the first mode (called generator) the emulator provides the data instead of the internal FLASH and traces the CPU actions. In the other mode (called observer) the internal FLASH provides the data and all internal actions are made visible to the emulator.

25.5.3.1 ROM Control in Single-Chip Modes

In single-chip modes the MCU has no external bus. All memory accesses and program fetches are internal (see Figure 25-27).

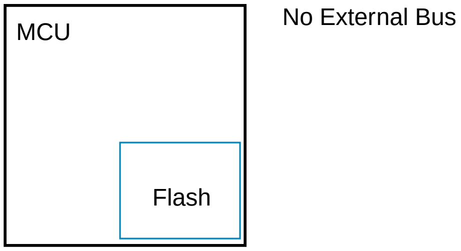


Figure 25-27. ROM in Single Chip Modes

25.5.3.2 ROM Control in Emulation Single-Chip Mode

In emulation single-chip mode the external bus is connected to the emulator. If the EROMON bit is set, the internal FLASH provides the data and the emulator can observe all internal CPU actions on the external bus. If the EROMON bit is cleared, the emulator provides the data (generator) and traces the all CPU actions (see Figure 25-28).

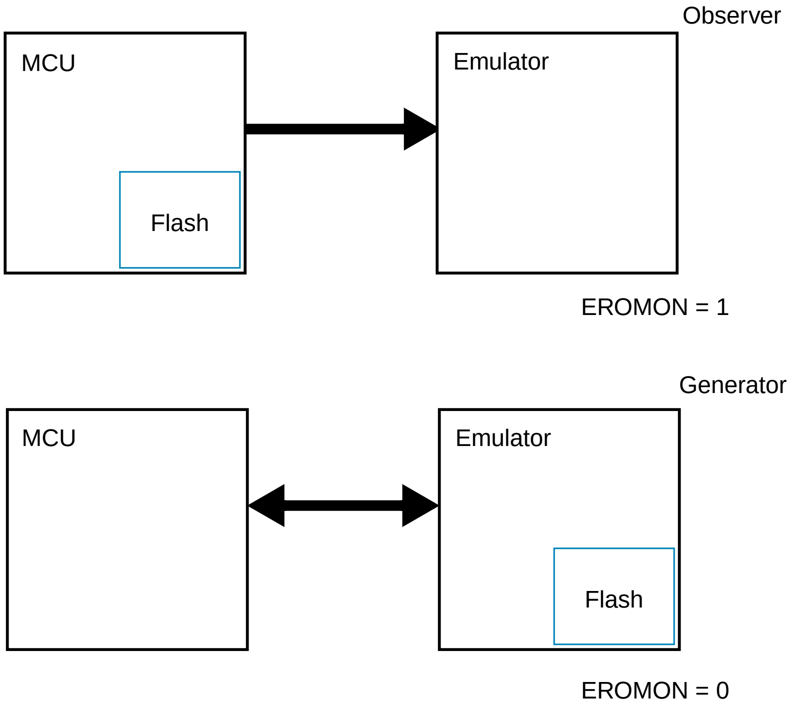


Figure 25-28. ROM in Emulation Single-Chip Mode

25.5.3.3 ROM Control in Normal Expanded Mode

In normal expanded mode the external bus will be connected to the application. If the ROMON bit is set, the internal FLASH provides the data. If the ROMON bit is cleared, the application memory provides the data (see Figure 25-29).

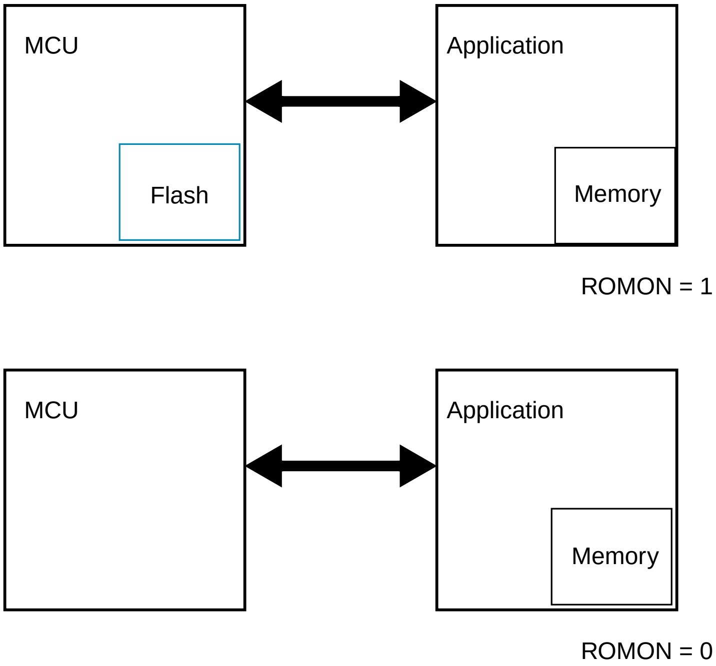


Figure 25-29. ROM in Normal Expanded Mode

25.5.3.4 ROM Control in Emulation Expanded Mode

In emulation expanded mode the external bus will be connected to the emulator and to the application. If the ROMON bit is set, the internal FLASH provides the data. If the EROMON bit is set as well the emulator observes all CPU internal actions, otherwise the emulator provides the data and traces all CPU actions (see Figure 25-30). When the ROMON bit is cleared, the application memory provides the data and the emulator will observe the CPU internal actions (see Figure 25-31).

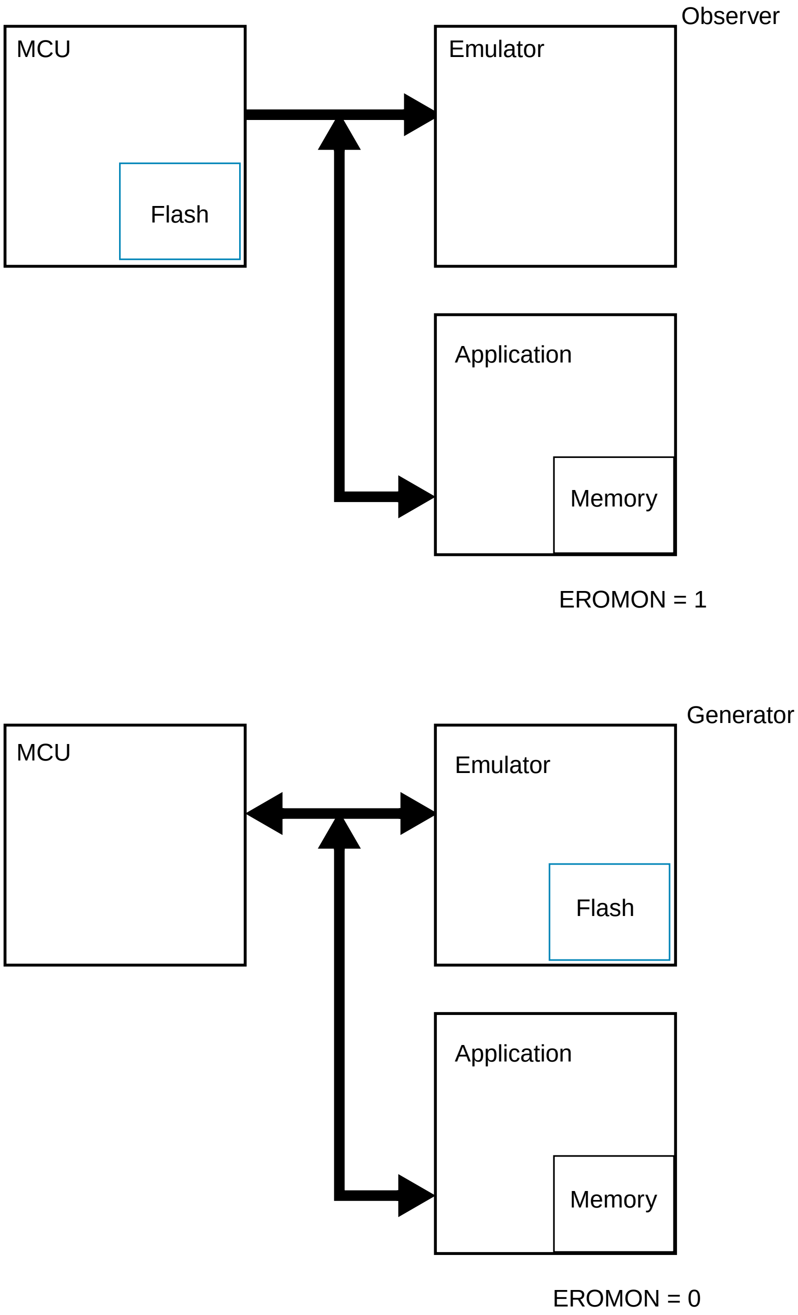


Figure 25-30. ROMON = 1 in Emulation Expanded Mode

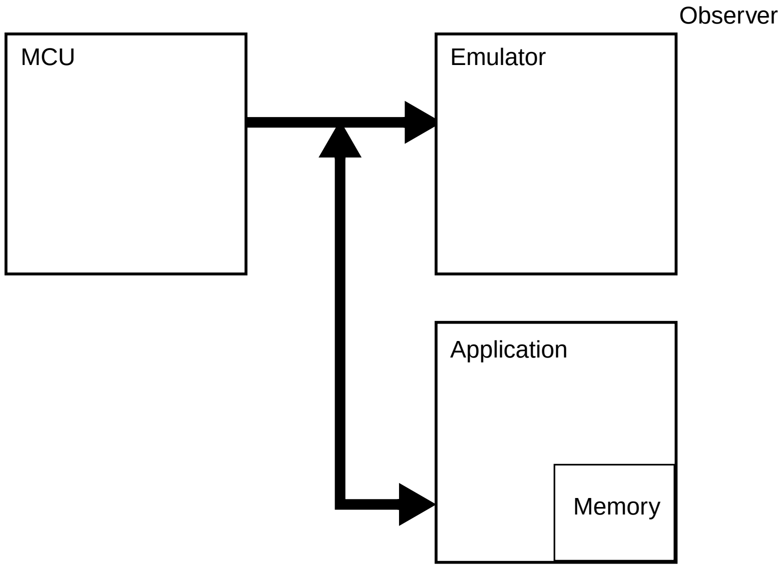


Figure 25-31. ROMON = 0 in Emulation Expanded Mode

25.5.3.5 ROM Control in Special Test Mode

In special test mode the external bus is connected to the application. If the ROMON bit is set, the internal FLASH provides the data, otherwise the application memory provides the data (see Figure 25-32).

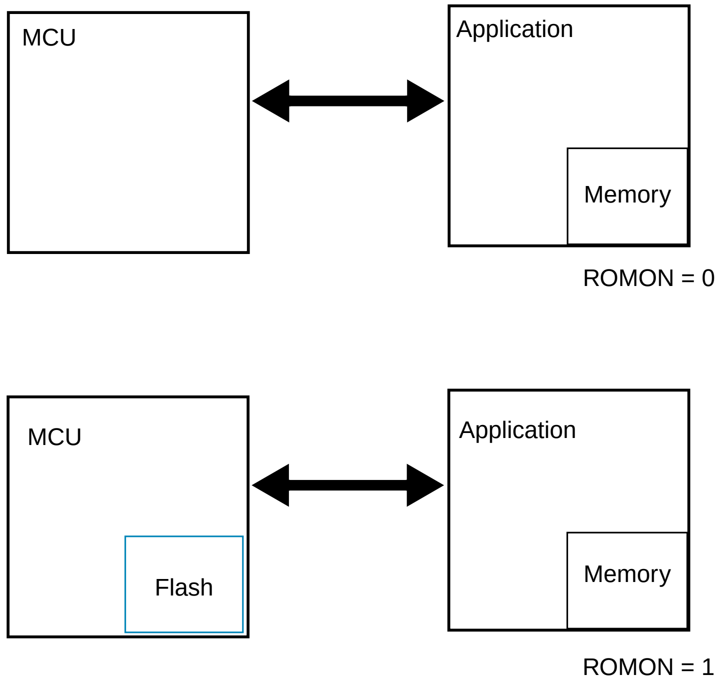


Figure 25-32. ROM in Special Test Mode

Appendix A
Electrical Characteristics

A.1 General

NOTE

> The electrical characteristics given in this section should be used as a guide only. Values cannot be guaranteed by Freescale and are subject to change without notice.

This supplement contains the most accurate electrical information for the MC9S12XHZ512 of microcontrollers available at the time of publication.

This introduction is intended to give an overview on several common topics like power supply, current injection etc.

A.1.1 Parameter Classification

The electrical parameters shown in this supplement are guaranteed by various methods. To give the customer a better understanding the following classification is used and the parameters are tagged accordingly in the tables where appropriate.

NOTE

> This classification is shown in the column labeled “C” in the parameter tables where appropriate.

P: Those parameters are guaranteed during production testing on each individual device.

C: Those parameters are achieved by the design characterization by measuring a statistically relevant sample size across process variations.

T: Those parameters are achieved by design characterization on a small sample size from typical devices under typical conditions unless otherwise noted. All values shown in the typical column are within this category.

D: Those parameters are derived mainly from simulations.

A.1.2 Power Supply

The MC9S12XHZ512 utilizes several pins to supply power to the I/O ports, A/D converter, oscillator, and PLL as well as the digital core.

The V_{DDA}, V_{SSA} pair supplies the A/D converter and parts of the internal voltage regulator.

The V_{DDX1}/V_{SSX1} and V_{DDX2}/V_{SSX2} pairs supply the I/O pins except PU, PV and PW. V_{DDR} supplies the internal voltage regulator.

V_{DDM1}/V_{SSM1}, V_{DDM2}/V_{SSM2} and V_{DDM3}/V_{SSM3} pairs supply the ports PU, PV and PW.

V_{DD1}, V_{SS1} and V_{SS2} are the supply pins for the digital logic, V_{DDPLL}, V_{SSPLL} supply the oscillator and the PLL.

V_{SS1} and V_{SS2} are internally connected by metal.

V_{DDA}, V_{DDX1}, V_{DDX2}, V_{DDM} as well as V_{SSA}, V_{SSX1}, V_{SSX2} and V_{SSM} are connected by anti-parallel diodes for ESD protection.

NOTE

> In the following context V_{DD5} is used for either V_{DDA}, V_{DDM}, V_{DDR} and $V_{DDX1/2}$; V_{SS5} is used for either V_{SSA}, V_{SSR} and V_{SSX} unless otherwise noted.
> I_{DD5} denotes the sum of the currents flowing into the V_{DDA}, $V_{DDX1/2}$, V_{DDM} and V_{DDR} pins.
> V_{DD} is used for V_{DD1} and V_{DDPLL}, V_{SS} is used for V_{SS1}, V_{SS2} and V_{SSPLL}.
> I_{DD} is used for the sum of the currents flowing into V_{DD1} and V_{DDPLL}.

A.1.3 Pins

There are four groups of functional pins.

A.1.3.1 I/O Pins

Those I/O pins have a nominal level of 5 V. This class of pins is comprised of all port I/O pins, the analog inputs, BKGD and the $\overline{\text{RESET}}$ pins.The internal structure of all those pins is identical; however, some of the functionality may be disabled. For example, for the analog inputs the output drivers, pull-up and pull-down resistors are disabled permanently.

A.1.3.2 Analog Reference

This group is made up by the V_{RH} and V_{RL} pins.

A.1.3.3 Oscillator

The pins XFC, EXTAL, XTAL dedicated to the oscillator have a nominal 2.5 V level. They are supplied by V_{DDPLL}.

A.1.3.4 TEST

This pin is used for production testing only.

A.1.4 Current Injection

Power supply must maintain regulation within operating V_{DD5} or V_{DD} range during instantaneous and operating maximum current conditions. If positive injection current ($V_{in} > V_{DD5}$) is greater than I_{DD5}, the injection current may flow out of V_{DD5} and could result in external power supply going out of regulation. Ensure external V_{DD5} load will shunt current greater than maximum injection current. This will be the greatest risk when the MCU is not consuming power; e.g., if no system clock is present, or if clock rate is very low which would reduce overall power consumption.

A.1.5 Absolute Maximum Ratings

Absolute maximum ratings are stress ratings only. A functional operation under or outside those maxima is not guaranteed. Stress beyond those limits may affect the reliability or cause permanent damage of the device.

This device contains circuitry protecting against damage due to high static voltage or electrical fields; however, it is advised that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either V_{SS5} or V_{DD5}).

Table A-1. Absolute Maximum Ratings[1]

Num	Rating	Symbol	Min	Max	Unit
1	I/O, regulator and analog supply voltage	V_{DD5}	–0.3	6.0	V
2	Digital logic supply voltage[2]	V_{DD}	–0.3	3.0	V
3	PLL supply voltage[2]	V_{DDPLL}	–0.3	3.0	V
4	Voltage difference V_{DDX} to V_{DDR} to V_{DDM} and V_{DDA}	Δ_{VDDX}	–0.3	0.3	V
5	Voltage difference V_{SSX} to V_{SSR} to V_{SSM} and V_{SSA}	Δ_{VSSX}	–0.3	0.3	V
6	Digital I/O input voltage	V_{IN}	–0.3	6.0	V
7	Analog reference	$V_{RH,}$ V_{RL}	–0.3	6.0	V
8	XFC, EXTAL, XTAL inputs	V_{ILV}	–0.3	3.0	V
9	TEST input	V_{TEST}	–0.3	10.0	V
10	Instantaneous maximum current Single pin limit for all digital I/O pins except Port U, V and W[3]	I_D	–25	+25	mA
10	Instantaneous maximum current Single pin limit for Port U, V and W[4]	I_D	–55	+55	mA
11	Instantaneous maximum current Single pin limit for XFC, EXTAL, XTAL[5]	I_{DL}	–25	+25	mA
12	Instantaneous maximum current Single pin limit for TEST [6]	I_{DT}	–0.25	0	mA
13	Storage temperature range	T_{stg}	–65	155	°C

[1] Beyond absolute maximum ratings device might be damaged.

[2] The device contains an internal voltage regulator to generate the logic and PLL supply out of the I/O supply. The absolute maximum ratings apply when the device is powered from an external source.

[3] All digital I/O pins are internally clamped to $V_{SSX1/2}$ and $V_{DDX1/2}$ or V_{SSA} and V_{DDA}.

[4] Ports U, V and W are internally clamped to $V_{SSM1/2/3}$ and $V_{DDM1/2/3}$.

[5] Those pins are internally clamped to V_{SSPLL} and V_{DDPLL}.

[6] This pin is clamped low to V_{SSPLL}, but not clamped high. This pin must be tied low in applications.

A.1.6 ESD Protection and Latch-up Immunity

All ESD testing is in conformity with CDF-AEC-Q100 stress test qualification for automotive grade integrated circuits. During the device qualification ESD stresses were performed for the Human Body Model (HBM) and the Charge Device Model.

A device will be defined as a failure if after exposure to ESD pulses the device no longer meets the device specification. Complete DC parametric and functional testing is performed per the applicable device specification at room temperature followed by hot temperature, unless specified otherwise in the device specification.

Table A-2. ESD and Latch-up Test Conditions

Model	Description	Symbol	Value	Unit
Human Body	Series resistance	R1	1500	Ohm
	Storage capacitance	C	100	pF
	Number of pulse per pin Positive Negative	 — —	 3 3	
Latch-up	Minimum input voltage limit		–2.5	V
	Maximum input voltage limit		7.5	V

Table A-3. ESD and Latch-Up Protection Characteristics

Num	C	Rating	Symbol	Min	Max	Unit
1	C	Human Body Model (HBM)	V_{HBM}	2000	—	V
2	C	Charge Device Model (CDM)	V_{CDM}	500	—	V
3	C	Latch-up current at T_A = 125°C Positive Negative	I_{LAT}	 +100 –100	 — —	mA
4	C	Latch-up current at T_A = 27°C Positive Negative	I_{LAT}	 +200 –200	 — —	mA

A.1.7 Operating Conditions

This section describes the operating conditions of the device. Unless otherwise noted those conditions apply to all the following data.

NOTE

Please refer to the temperature rating of the device (C, V, M) with regards to the ambient temperature T_A and the junction temperature T_J. For power dissipation calculations refer to Section A.1.8, "Power Dissipation and Thermal Characteristics".

Table A-4. Operating Conditions

Rating	Symbol	Min	Typ	Max	Unit
I/O, regulator and analog supply voltage	V_{DD5}	4.5	5	5.5	V
Digital logic supply voltage[1]	V_{DD}	2.35	2.5	2.75	V
PLL supply voltage[2]	V_{DDPLL}	2.35	2.5	2.75	V
Voltage difference V_{DDX} to V_{DDR} to V_{DDM} and V_{DDA}	Δ_{VDDX}	–0.1	0	0.1	V
Voltage difference V_{SSX} to V_{SSR} to V_{SSM} and V_{SSA}	Δ_{VSSX}	–0.1	0	0.1	V
Oscillator	f_{osc}	0.5	—	16	MHz
Bus frequency	f_{bus}	0.5	—	40	MHz
MC9S12XHZ512**C** Operating junction temperature range Operating ambient temperature range[2]	 T_J T_A	 –40 –40	 — 27	 100 85	°C
MC9S12XHZ512**V** Operating junction temperature range Operating ambient temperature range[2]	 T_J T_A	 –40 –40	 — 27	 120 105	°C
MC9S12XHZ512**M** Operating junction temperature range Operating ambient temperature range[2]	 T_J T_A	 –40 –40	 — 27	 140 125	°C

[1] The device contains an internal voltage regulator to generate the logic and PLL supply out of the I/O supply. The absolute maximum ratings apply when this regulator is disabled and the device is powered from an external source.

[2] Please refer to Section A.1.8, "Power Dissipation and Thermal Characteristics" for more details about the relation between ambient temperature T_A and device junction temperature T_J.

A.1.8 Power Dissipation and Thermal Characteristics

Power dissipation and thermal characteristics are closely related. The user must assure that the maximum operating junction temperature is not exceeded. The average chip-junction temperature (T_J) in °C can be obtained from:

$$T_J = T_A + (P_D \bullet \Theta_{JA})$$

T_J = Junction Temperature, [°C]

T_A = Ambient Temperature, [°C]

P_D = Total Chip Power Dissipation, [W]

Θ_{JA} = Package Thermal Resistance, [°C/W]

The total power dissipation can be calculated from:

$$P_D = P_{INT} + P_{IO}$$

P_{INT} = Chip Internal Power Dissipation, [W]

Two cases with internal voltage regulator enabled and disabled must be considered:

1. Internal voltage regulator disabled

$$P_{INT} = I_{DD} \cdot V_{DD} + I_{DDPLL} \cdot V_{DDPLL} + I_{DDA} \cdot V_{DDA}$$

$$P_{IO} = \sum_i R_{DSON} \cdot I_{IO_i}{}^2$$

P_{IO} is the sum of all output currents on I/O ports associated with V_{DDX} and V_{DDR}.

For R_{DSON} is valid:

$$R_{DSON} = \frac{V_{OL}}{I_{OL}}\text{;for outputs driven low}$$

respectively

$$R_{DSON} = \frac{V_{DD5} - V_{OH}}{I_{OH}}\text{;for outputs driven high}$$

2. Internal voltage regulator enabled

$$P_{INT} = I_{DDR} \cdot V_{DDR} + I_{DDA} \cdot V_{DDA}$$

I_{DDR} is the current shown in Table A-9 and not the overall current flowing into V_{DDR}, which additionally contains the current flowing into the external loads with output high.

$$P_{IO} = \sum_i R_{DSON} \cdot I_{IO_i}{}^2$$

P_{IO} is the sum of all output currents on I/O ports associated with V_{DDX} and V_{DDR}.

Table A-5. Thermal Package Characteristics[1]

Num	C	Rating	Symbol	Min	Typ	Max	Unit
LQFP144							
1	T	Thermal resistance LQFP144, single sided PCB[2]	θ_{JA}	—	—	41	°C/W
2	T	Thermal resistance LQFP144, double sided PCB with 2 internal planes[3]	θ_{JA}	—	—	32	°C/W
3		Junction to Board LQFP 144	θ_{JB}	—	—	22	°C/W
4		Junction to Case LQFP 144[4]	θ_{JC}	—	—	7.4	°C/W
5		Junction to Package Top LQFP144[5]	Ψ_{JT}	—	—	3	°C/W
LQFP112							
6	T	Thermal resistance LQFP112, single sided PCB[2]	θ_{JA}	—	—	43	°C/W
7	T	Thermal resistance LQFP112, double sided PCB with 2 internal planes[3]	θ_{JA}	—	—	32	°C/W
8		Junction to Board LQFP112	θ_{JB}	—	—	22	°C/W
9		Junction to Case LQFP112[4]	θ_{JC}	—	—	7	°C/W
10		Junction to Package Top LQFP112[5]	Ψ_{JT}	—	—	3	°C/W

[1] The values for thermal resistance are achieved by package simulations

[2] Junction to ambient thermal resistance, θ_{JA} was simulated to be equivalent to the JEDEC specification JESD51-2 in a horizontal configuration in natural convection.

[3] Junction to ambient thermal resistance, θ_{JA} was simulated to be equivalent to the JEDEC specification JESD51-7 in a horizontal configuration in natural convection.

A.1.9 I/O Characteristics

This section describes the characteristics of all I/O pins except EXTAL, XTAL,XFC,TEST and supply pins.

Table A-6. I/O Characteristics

Conditions are 4.5 V < V_{DD5} < 5.5 V temperature from –40°C to +140°C, unless otherwise noted							
Num	**C**	**Rating**	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
1	P	Input high voltage	V_{IH}	0.65*V_{DD5}	—	—	V
	T			—	—	V_{DD5} + 0.3	V
2	P	Input low voltage	V_{IL}	—	—	0.35*V_{DD5}	V
	T			V_{SS5} – 0.3	—	—	V
3	C	Input hysteresis	V_{HYS}		250	—	mV
4	P	Input leakage current (input mode)[1] V_{in} = V_{DD5} or V_{SS5} All I/O pins except Port U, V, W Port U , V and W	I_{in}	 –1 –2.5	 — —	 1 2.5	 µA µA
5	 C P P	Output high voltage (output mode) Partial drive: I_{OH} = –2 mA Full drive: I_{OH} = –10 mA[2] Port U, V, W: I_{OH} = –20 mA [3]	V_{OH}	 V_{DD5} – 0.8 V_{DD5} – 0.8 V_{DD5} – 0.32	 — — V_{DD5} – 0.2	 — — —	 V V V
6	 C P	Output low voltage (output mode) Partial drive: I_{OL} = +2 mA Full drive: I_{OL} = +10 mA [2] Port U, V, W: I_{OH} = +20 mA [3]	V_{OL}	 — — —	 — — 0.2	 0.8 0.8 0.32	 V V V
7	C	Output Rise Time (slew enabled) V_{DD5}=5V, 10% to 90% of V_{OH} Partial drive (Cload 50pF) Full drive (Cload 50pF) Port U, V, W (Rload=1KΩ)	t_r	 — — 50	 25 9 100	 — — 150	 ns ns ns
8	C	Output Fall Time (slew enabled) V_{DD5}=5V, 10% to 90% of V_{OH} Partial drive (Cload 50pF) Full drive (Cload 50pF) Port U, V, W (Rload=1KΩ)	t_f	 — — 50	 25 9 100	 — — 150	 ns ns ns
9	P	Internal pull up device current, tested at V_{IL} max	I_{PUL}	—	—	–130	µA
10	C	Internal pull up device current, tested at V_{IH} min	I_{PUH}	–10	—	—	µA
11	P	Internal pull down device current, tested at V_{IH} min	I_{PDH}	—	—	130	µA
12	C	Internal pull down device current, tested at V_{IL} max	I_{PDL}	10	—	—	µA
13	D	Input capacitance	C_{in}	—	6	—	pF
14	T	Injection current[4] Single pin limit Total device Limit, sum of all injected currents	 I_{ICS} I_{ICP}	 –2.5 –25	—	 2.5 25	mA
15	P	Port AD interrupt input pulse filtered[5] passed[3]	t_{PULSE}	 — 10	 — —	 3 —	 µs µs

[1] Maximum leakage current occurs at maximum operating temperature. Current decreases by approximately one-half for each 8 C to 12 C in the temperature range from 50°C to 125°C.

[2] At a maximum load of 100mA per VDDX1 and VDDX2 power supplies.

[3] Typical Value at 25°C, with V_{DD5} > 4.75V.

[4] Refer to Section A.1.4, “Current Injection” for more details

[5] Parameter only applies in stop or pseudo stop mode.

Table A-7. I/O Characteristics for Port C, D, PE5, PE6, and PE7 for Reduced Input Voltage Thresholds

Conditions are 4.5 V < V_{DD5} < 5.5 V Temperature from –40°C to +140°C, unless otherwise noted							
Num	**C**	**Rating**	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
1	P	Input high voltage	V_{IH}	1.75	—	—	V
2	P	Input low voltage	V_{IL}	—	—	0.75	V
3	C	Input hysteresis	V_{HYS}	—	100	—	mV

A.1.10 Supply Currents

This section describes the current consumption characteristics of the device as well as the conditions for the measurements.

A.1.10.1 Measurement Conditions

All measurements are without output loads. Unless otherwise noted the currents are measured in single chip mode and the CPU and XGATE code is executed from RAM, V_{DD5}=5.5V, internal voltage regulator is enabled and the bus frequency is 40MHz using a 4-MHz oscillator in loop controlled Pierce mode. Production testing is performed using a square wave signal at the EXTAL input.

Table A-8. shows the configuration of the peripherals for run current measurement.

Table A-8. Peripheral Configurations for Run Supply Current Measurements

Peripheral	Configuration
MSCAN	configured to loop-back mode using a bit rate of 1Mbit/s
SPI	configured to master mode, continously transmit data (0x55 or 0xAA) at 1Mbit/s
SCI	configured into loop mode, continously transmit data (0x55) at speed of 57600 baud
IIC	operate in master mode and continously transmit data (0x55 or 0xAA) at the bit rate of 100Kbit/s
PWM	configured to toggle its pins at the rate of 40kHz
ECT	the peripheral shall be configured to output compare mode, Pulse accumulator and modulus counter enabled.
ATD	the peripheral is configured to operate at its maximum specified frequency and to continuously convert voltages on all input channels in sequence.
XGATE	XGATE fetches code from RAM, XGATE runs in an infinite loop , it reads the Status and Flag registers of CAN's, SPI's, SCI's in sequence and does some bit manipulation on the data
COP	COP Warchdog Rate 2^{24}
RTI	enabled, RTI Control Register (RTICTL) set to $FF
API	the module is configured to run from the RC oscillator clock source.
PIT	PIT is enabled, Micro-timer register 0 and 1 loaded with $0F and timer registers 0 to 3 are loaded with $03/07/0F/1F.
DBG	the module is enabled and the comparators are configured to trigger in outside range. The range covers all the code executed by the core.

A.1.10.2 Additional Remarks

In expanded modes the currents flowing in the system are highly dependent on the load at the address, data, and control signals as well as on the duty cycle of those signals. No generally applicable numbers can be given. A very good estimate is to take single chip currents and add the currents due to the external loads.

Table A-9. Run and Wait Current Characteristics

Conditions are shown in Table A-4 unless otherwise noted

Num	C	Rating	Symbol	Min	Typ	Max	Unit
Run supply current (Peripheral Configuration see Table A-8.)							
1	P	Peripheral Set[1] f_{osc}=4MHz, f_{bus}=40MHz	I_{DD5}			110	mA
2	 C T T	Peripheral Set[1] f_{osc}=4MHz, f_{bus}=40MHz f_{osc}=4MHz, f_{bus}=20MHz f_{osc}=4MHz, f_{bus}=8MHz			 90 45 18		
3	 T T T	Peripheral Set[2] f_{osc}=4MHz, f_{bus}=40MHz f_{osc}=4MHz, f_{bus}=20MHz f_{osc}=4MHz, f_{bus}=8MHz			 70 35 15		
4	 T T T	Peripheral Set[3] f_{osc}=4MHz, f_{bus}=40MHz f_{osc}=4MHz, f_{bus}=20MHz f_{osc}=4MHz, f_{bus}=8MHz			 60 30 13		
5	 T T T	Peripheral Set[4] f_{osc}=4MHz, f_{bus}=40MHz f_{osc}=4MHz, f_{bus}=20MHz f_{osc}=4MHz, f_{bus}=8MHz			 56 28 12		
6	 T T T	Peripheral Set[5] f_{osc}=4MHz, f_{bus}=40MHz f_{osc}=4MHz, f_{bus}=20MHz f_{osc}=4MHz, f_{bus}=8MHz			 53 26 11		
7	 T T T	Peripheral Set[6] f_{osc}=4MHz, f_{bus}=40MHz f_{osc}=4MHz, f_{bus}=20MHz f_{osc}=4MHz, f_{bus}=8MHz			 50 25 10		
Wait supply current							
8	P	Peripheral Set[1] ,PLL on XGATE executing code from RAM	I_{DDW}			95	mA
9	 T T	Peripheral Set[2] f_{osc}=4MHz, f_{bus}=40MHz f_{osc}=4MHz, f_{bus}=8MHz			 50 10		
10	P	All modules disabled, RTI enabled, PLL off				10	

[1] The following peripherals are on: ATD/ECT/IIC0/PWM/SPI/SCI0-SCI1/CAN0-CAN1/XGATE

[2] The following peripherals are on: ATD/ECT/IIC0/PWM/SPI/SCI0-SCI1/CAN0-CAN1

[3] The following peripherals are on: ATD/ECT/IIC0/PWM/SPI/SCI0-SCI1

[4] The following peripherals are on: ATD/ECT/IIC0/PWM/SPI

[5] The following peripherals are on: ATD/ECT/IIC0/PWM

[6] The following peripherals are on: ATD/ECT/IIC0

Table A-10. Pseudo Stop and Full Stop Current

Conditions are shown in Table A-4 unless otherwise noted							
Num	**C**	**Rating**	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
Pseudo stop current (API, RTI, and COP disabled) PLL off							
10	C	–40°C	I_{DDPS}	—	200	—	µA
	P	27°C		—	300	500	
	C	70°C		—	400	—	
	C	85°C		—	500	—	
	P	"C" Temp Option 100°C		—	600	2500	
	C	105°C		—	800	—	
	P	"V" Temp Option 120°C		—	1000	3500	
	C	125°C		—	1200	—	
	P	"M" Temp Option 140°C		—	1500	7000	
Pseudo stop current (API, RTI, and COP enabled) PLL off							
11	C	–40°C	I_{DDPS}	—	500	—	µA
	C	27°C		—	750	—	
	C	70°C		—	850	—	
	C	85°C		—	1000	—	
	C	105°C		—	1200	—	
	C	125°C		—	1500	—	
	C	140°C		—	2000	—	
Stop Current							
12	C	–40°C	I_{DDS}	—	20	—	µA
	P	27°C		—	30	100	
	C	70°C		—	100	—	
	C	85°C		—	200	—	
	P	"C" Temp Option 100°C		—	250	2000	
	C	105°C		—	400	—	
	P	"V" Temp Option 120°C		—	500	3000	
	C	125°C		—	600	—	
	P	"M" Temp Option 140°C		—	1000	7000	

A.2 ATD

This section describes the characteristics of the analog-to-digital converter.

A.2.1 ATD Operating Characteristics

The Table A-11 and Table A-13 show conditions under which the ATD operates.

The following constraints exist to obtain full-scale, full range results:

$$V_{SSA} \le V_{RL} \le V_{IN} \le V_{RH} \le V_{DDA}.$$

This constraint exists since the sample buffer amplifier can not drive beyond the power supply levels that it ties to. If the input level goes outside of this range it will effectively be clipped.

Table A-11. ATD Operating Characteristics

Conditions are shown in Table A-4 unless otherwise noted, supply voltage 4.5 V < V_{DDA} < 5.5 V							
Num	**C**	**Rating**	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
1	D	Reference potential Low High	 V_{RL} V_{RH}	 V_{SSA} $V_{DDA}/2$	 — —	 $V_{DDA}/2$ V_{DDA}	 V V
2	C	Differential reference voltage[1]	V_{RH}-V_{RL}	4.50	5.00	5.5	V
3	D	ATD clock frequency	f_{ATDCLK}	0.5		2.0	MHz
4	D	ATD 10-bit conversion period Clock cycles[2] Conv, time at 2.0 MHz ATD clock f_{ATDCLK}	 N_{CONV10} T_{CONV10}	 14 7	 — —	 28 14	 Cycles µs
5	D	ATD 8-Bit conversion period Clock cycles[2] Conv, time at 2.0 MHz ATD clock f_{ATDCLK}	 N_{CONV8} T_{CONV8}	 12 6	 — —	 26 13	 Cycles µs
6	D	Recovery time (V_{DDA} = 5.0 Volts)	t_{REC}	—	—	20	µs
7	P	Reference supply current	I_{REF}	—	—	0.375	mA

[1] Full accuracy is not guaranteed when differential voltage is less than 4.50 V

[2] The minimum time assumes a final sample period of 2 ATD clocks cycles while the maximum time assumes a final sample period of 16 ATD clocks.

A.2.2 Factors Influencing Accuracy

Three factors — source resistance, source capacitance and current injection — have an influence on the accuracy of the ATD.

A.2.2.1 Source Resistance

Due to the input pin leakage current as specified in Table A-6 in conjunction with the source resistance there will be a voltage drop from the signal source to the ATD input. The maximum source resistance R_S specifies results in an error of less than 1/2 LSB (2.5 mV) at the maximum leakage current. If device or operating conditions are less than worst case or leakage-induced error is acceptable, larger values of source resistance is allowed.

A.2.2.2 Source Capacitance

When sampling an additional internal capacitor is switched to the input. This can cause a voltage drop due to charge sharing with the external and the pin capacitance. For a maximum sampling error of the input voltage ≤ 1LSB, then the external filter capacitor, $C_f \geq 1024 * (C_{INS}–C_{INN})$.

A.2.2.3 Current Injection

There are two cases to consider.

1. A current is injected into the channel being converted. The channel being stressed has conversion values of \$3FF (\$FF in 8-bit mode) for analog inputs greater than V_{RH} and \$000 for values less than V_{RL} unless the current is higher than specified as disruptive condition.
2. Current is injected into pins in the neighborhood of the channel being converted. A portion of this current is picked up by the channel (coupling ratio K), This additional current impacts the accuracy of the conversion depending on the source resistance.

   The additional input voltage error on the converted channel can be calculated as:

   $$V_{ERR} = K * R_S * I_{INJ}$$

   with I_{INJ} being the sum of the currents injected into the two pins adjacent to the converted channel.

Table A-12. ATD Electrical Characteristics

Conditions are shown in Table A-4 unless otherwise noted							
Num	**C**	**Rating**	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
1	C	Max input source resistance	R_S	—	—	1	KΩ
2	T	Total input capacitance Non sampling Sampling	 C_{INN} C_{INS}	 — —	 — —	 10 22	pF
3	C	Disruptive analog input current	I_{NA}	–2.5	—	2.5	mA
4	C	Coupling ratio positive current injection	K_p	—	—	10^{-4}	A/A
5	C	Coupling ratio negative current injection	K_n	—	—	10^{-2}	A/A

A.2.3 ATD Accuracy

Table A-13 specifies the ATD conversion performance excluding any errors due to current injection, input capacitance, and source resistance.

Table A-13. ATD Conversion Performance

Conditions are shown in Table A-4 unless otherwise noted
$V_{REF} = V_{RH} – V_{RL}$ = 5.12 V. Resulting to one 8-bit count = 20 mV and one 10-bit count = 5 mV
f_{ATDCLK} = 2.0 MHz

Num	C	Rating	Symbol	Min	Typ	Max	Unit
1	P	10-bit resolution	LSB	—	5	—	mV
2	P	10-bit differential nonlinearity	DNL	–1	—	1	Counts
3	P	10-bit integral nonlinearity	INL	–2.5	±1.5	2.5	Counts
4	P	10-bit absolute error (Port AD)[1]	AE	–3	±2.0	3	Counts
5	P	10-bit absolute error (Port L)[1]	AE	–4	±3.0	4	Counts
6	P	8-bit resolution	LSB	—	20	—	mV
7	P	8-bit differential nonlinearity	DNL	–0.5	—	0.5	Counts
8	P	8-bit integral nonlinearity	INL	–1.0	±0.5	1.0	Counts
9	P	8-bit absolute error (Port AD)[1]	AE	–1.5	±1.0	1.5	Counts
9	P	8-bit absolute error (Port L)[1]	AE	–2.0	±1.5	2.0	Counts

[1] These values include the quantization error which is inherently 1/2 count for any A/D converter.

A.2.3.1 ATD Accuracy Definitions

For the following definitions see also Figure A-1.

Differential non-linearity (DNL) is defined as the difference between two adjacent switching steps.

$$\mathrm{DNL}(i) = \frac{V_i - V_{i-1}}{1\mathrm{LSB}} - 1$$

The integral non-linearity (INL) is defined as the sum of all DNLs:

$$\mathrm{INL}(n) = \sum_{i=1}^{n} \mathrm{DNL}(i) = \frac{V_n - V_0}{1\mathrm{LSB}} - n$$

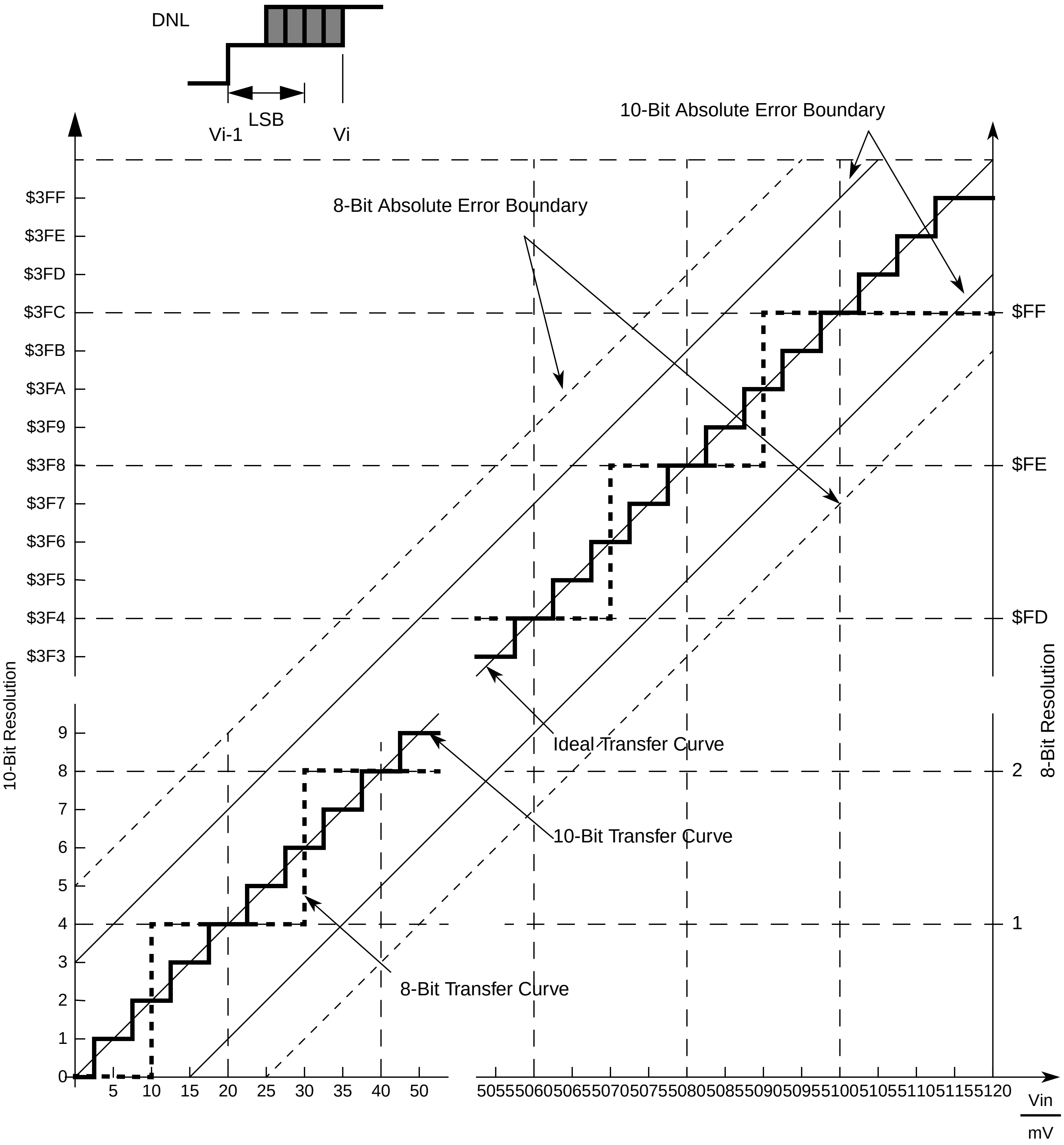


Figure A-1. ATD Accuracy Definitions

NOTE

Figure A-1 shows only definitions, for specification values refer to Table A-13.

A.3 NVM, Flash, and EEPROM

NOTE

> Unless otherwise noted the abbreviation NVM (nonvolatile memory) is used for both Flash and EEPROM.

A.3.1 NVM Timing

The time base for all NVM program or erase operations is derived from the oscillator. A minimum oscillator frequency f_{NVMOSC} is required for performing program or erase operations. The NVM modules do not have any means to monitor the frequency and will not prevent program or erase operation at frequencies above or below the specified minimum. Attempting to program or erase the NVM modules at a lower frequency a full program or erase transition is not assured.

The Flash and EEPROM program and erase operations are timed using a clock derived from the oscillator using the FCLKDIV and ECLKDIV registers respectively. The frequency of this clock must be set within the limits specified as f_{NVMOP}.

The minimum program and erase times shown in Table A-14 are calculated for maximum f_{NVMOP} and maximum f_{bus}. The maximum times are calculated for minimum f_{NVMOP} and a f_{bus} of 2 MHz.

A.3.1.1 Single Word Programming

The programming time for single word programming is dependant on the bus frequency as a well as on the frequency f_{NVMOP} and can be calculated according to the following formula.

$$t_{swpgm} = 9 \cdot \frac{1}{f_{NVMOP}} + 25 \cdot \frac{1}{f_{bus}}$$

A.3.1.2 Burst Programming

This applies only to the Flash where up to 64 words in a row can be programmed consecutively using burst programming by keeping the command pipeline filled. The time to program a consecutive word can be calculated as:

$$t_{bwpgm} = 4 \cdot \frac{1}{f_{NVMOP}} + 9 \cdot \frac{1}{f_{bus}}$$

The time to program a whole row is:

$$t_{brpgm} = t_{swpgm} + 63 \cdot t_{bwpgm}$$

Burst programming is more than 2 times faster than single word programming.

A.3.1.3 Sector Erase

Erasing a 1024-byte Flash sector or a 4-byte EEPROM sector takes:

$$t_{era} \approx 4000 \cdot \frac{1}{f_{NVMOP}}$$

The setup time can be ignored for this operation.

A.3.1.4 Mass Erase

Erasing a NVM block takes:

$$t_{mass} \approx 20000 \cdot \frac{1}{f_{NVMOP}}$$

The setup time can be ignored for this operation.

A.3.1.5 Blank Check

The time it takes to perform a blank check on the Flash or EEPROM is dependant on the location of the first non-blank word starting at relative address zero. It takes one bus cycle per word to verify plus a setup of the command.

$$t_{check} \approx location \cdot t_{cyc} + 10 \cdot t_{cyc}$$

Table A-14. NVM Timing Characteristics

Conditions are shown in Table A-4 unless otherwise noted

Num	C	Rating	Symbol	Min	Typ	Max	Unit
1	D	External oscillator clock	f_{NVMOSC}	0.5	—	80[1]	MHz
2	D	Bus frequency for programming or erase operations	f_{NVMBUS}	1	—	—	MHz
3	D	Operating frequency	f_{NVMOP}	150	—	200	kHz
4	P	Single word programming time	t_{swpgm}	46[2]	—	74.5[3]	µs
5	D	Flash burst programming consecutive word [4]	t_{bwpgm}	20.4[2]	—	31[3]	µs
6	D	Flash burst programming time for 64 words[4]	t_{brpgm}	1331.2[2]	—	2027.5[3]	µs
7	P	Sector erase time	t_{era}	20[5]	—	26.7[3]	ms
8	P	Mass erase time	t_{mass}	100[5]	—	133[3]	ms
9	D	Blank check time Flash per block	t_{check}	11[6]	—	65546[7]	t_{cyc}
10	D	Blank check time EEPROM per block	t_{check}	11[6]	—	2058[7]	t_{cyc}

[1] Restrictions for oscillator in crystal mode apply.

[2] Minimum programming times are achieved under maximum NVM operating frequency f_{NVMOP} and maximum bus frequency f_{bus}.

[3] Maximum erase and programming times are achieved under particular combinations of f_{NVMOP} and bus frequency f_{bus}. Refer to formulae in Sections Section A.3.1.1, "Single Word Programming" – Section A.3.1.4, "Mass Erase" for guidance.

[4] Burst programming operations are not applicable to EEPROM

[5] Minimum erase times are achieved under maximum NVM operating frequency, f_{NVMOP}.

[6] Minimum time, if first word in the array is not blank

[7] Maximum time to complete check on an erased block

A.3.2 NVM Reliability

The reliability of the NVM blocks is guaranteed by stress test during qualification, constant process monitors and burn-in to screen early life failures. The program/erase cycle count on the sector is incremented every time a sector or mass erase event is executed

Table A-15. NVM Reliability Characteristics[1]

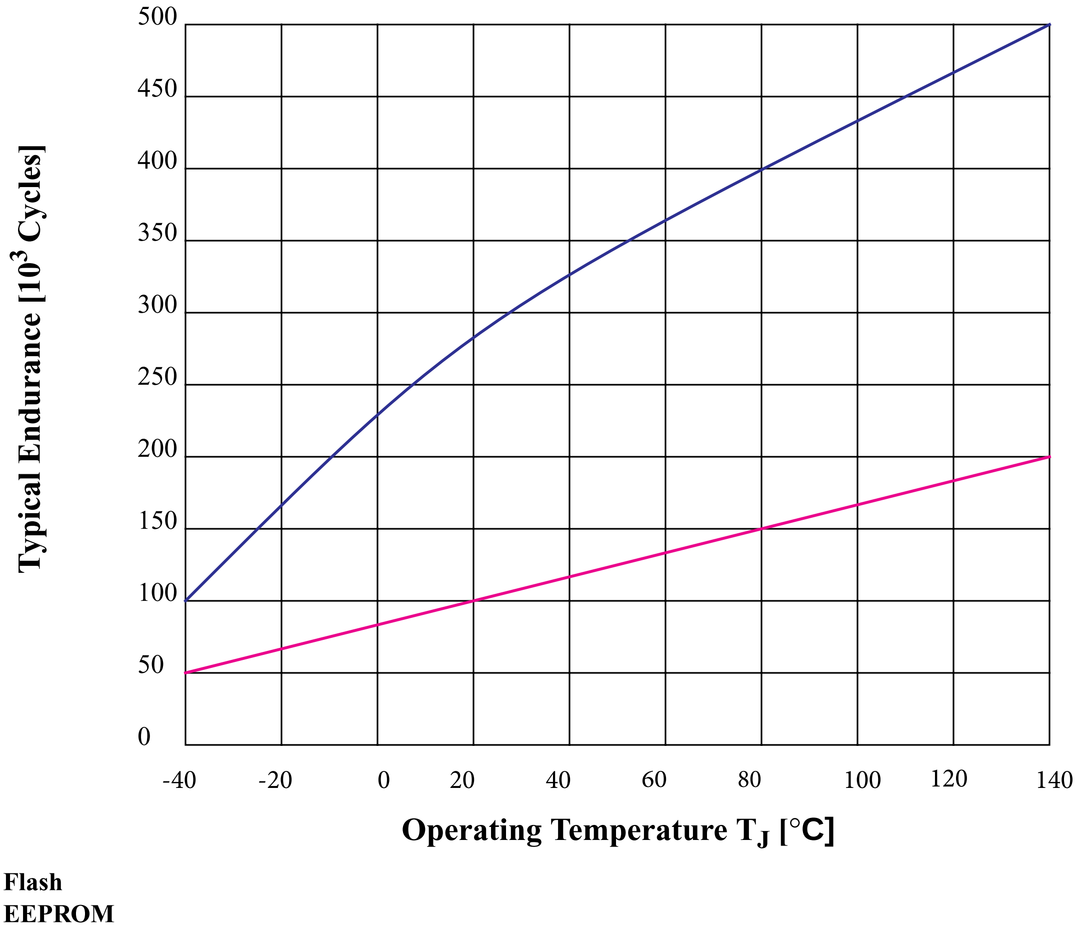
Conditions are shown in Table A-4 unless otherwise noted							
Num	**C**	**Rating**	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
Flash Reliability Characteristics							
1	C	Data retention after 10,000 program/erase cycles at an average junction temperature of $T_{Javg} \leq 85°C$	t_{FLRET}	15	100[2]	—	Years
2	C	Data retention with <100 program/erase cycles at an average junction temperature $T_{Javg} \leq 85°C$		20	100[2]	—	
3	C	Number of program/erase cycles $(–40°C \leq T_J \leq 0°C)$	n_{FL}	10,000	—	—	Cycles
4	C	Number of program/erase cycles $(0°C \leq T_J \leq 140°C)$		10,000	100,000[3]	—	
EEPROM Reliability Characteristics							
5	C	Data retention after up to 100,000 program/erase cycles at an average junction temperature of $T_{Javg} \leq 85°C$	t_{EEPRET}	15	100[2]	—	Years
6	C	Data retention with <100 program/erase cycles at an average junction temperature $T_{Javg} \leq 85°C$		20	100[2]	—	
7	C	Number of program/erase cycles $(–40°C \leq T_J \leq 0°C)$	n_{EEP}	10,000	—	—	Cycles
8	C	Number of program/erase cycles $(0°C < T_J \leq 140°C)$		100,000	300,000[3]	—	

[1] T_{Javg} will not exeed 85°C considering a typical temperature profile over the lifetime of a consumer, industrial or automotive application.

[2] Typical data retention values are based on intrinsic capability of the technology measured at high temperature and de-rated to 25°C using the Arrhenius equation. For additional information on how Freescale defines Typical Data Retention, please refer to Engineering Bulletin EB618.

[3] Spec table quotes typical endurance evaluated at 25°C for this product family, typical endurance at various temperature can be estimated using the graph below. For additional information on how Freescale defines Typical Endurance, please refer to Engineering Bulletin EB619.

Figure A-2. Typical Endurance vs Temperature



A.4 Voltage Regulator

Table A-16. Voltage Regulator Electrical Characteristics

Num	C	Characteristic	Symbol	Min	Typ	Max	Unit
1	P	Input voltages	$V_{VDDR,A}$	4.5	—	5.5	V
3	P	Output voltage core	V_{DD}				
		Full performance mode		2.35	2.54	2.75	V
		Reduced power mode		1.4	2.25	2.75	V
		Shutdown mode		—	—[1]	—	V
4	P	Output Voltage PLL	V_{DDPLL}				
		Full Performance Mode		2.35	2.54	2.75	V
		Reduced power mode		1.25	2.25	2.75	V
		Shutdown mode		—	—[2]	—	V
7	P	Low-voltage interrupt[3]					
		Assert level	V_{LVIA}	4.0	4.37	4.66	V
		Deassert level	V_{LVID}	4.15	4.52	4.77	V
8	P	Low-voltage reset[4]	V_{LVRA}				
		Assert level		2.25	—	—	V
9	C	Power-on reset[5]					
		Assert level	V_{PORA}	0.97	—	—	V
		Deassert level	V_{PORD}	—	—	2.05	V
12	C	Trimmed API internal clock[6] $\Delta f / f_{nominal}$	df_{API}	– 10%	—	+ 10%	—

[1] High impedance output

[2] High impedance output

[3] Monitors V_{DDA}, active only in full performance mode. Indicates I/O and ADC performance degradation due to low supply voltage.

[4] Monitors V_{DD}, active only in full performance mode. MCU is monitored by the POR in RPM (see Figure A-1)

[5] Monitors V_{DD}. Active in all modes.

[6] The API Trimming bits must be set that the minimum periode equals to 0.2 ms. $f_{nominal}$ = 1/0.2ms

A.5 Reset, Oscillator, and PLL

This section summarizes the electrical characteristics of the various startup scenarios for oscillator and phase-locked loop (PLL).

A.5.1 Startup

Table A-17 summarizes several startup characteristics explained in this section. Detailed description of the startup behavior can be found in the Clock and Reset Generator (CRG) Block Guide.

Table A-17. Startup Characteristics

Conditions are shown in Table A-4unless otherwise noted							
Num	**C**	**Rating**	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
1	D	Reset input pulse width, minimum input time	PW_{RSTL}	2	—	—	t_{osc}
2	D	Startup from reset	n_{RST}	192	—	196	n_{osc}
3	D	Interrupt pulse width, $\overline{\text{IRQ}}$ edge-sensitive mode	PW_{IRQ}	20	—	—	ns
4	D	Wait recovery startup time	t_{WRS}	—	—	14	t_{cyc}
5	D	Fast wakeup from STOP[1]	t_{fws}	—	50	—	µs

[1] V_{DD1} filter capacitor 220 nF, V_{DD5} = 5 V, T= 25°C

A.5.1.1 POR

The release level V_{PORR} and the assert level V_{PORA} are derived from the V_{DD} supply. They are also valid if the device is powered externally. After releasing the POR reset the oscillator and the clock quality check are started. If after a time t_{CQOUT} no valid oscillation is detected, the MCU will start using the internal self clock. The fastest startup time possible is given by n_{uposc}.

A.5.1.2 SRAM Data Retention

Provided an appropriate external reset signal is applied to the MCU, preventing the CPU from executing code when V_{DD5} is out of specification limits, the SRAM contents integrity is guaranteed if after the reset the PORF bit in the CRG flags register has not been set.

A.5.1.3 External Reset

When external reset is asserted for a time greater than PW_{RSTL} the CRG module generates an internal reset, and the CPU starts fetching the reset vector without doing a clock quality check, if there was an oscillation before reset.

A.5.1.4 Stop Recovery

Out of stop the controller can be woken up by an external interrupt. A clock quality check as after POR is performed before releasing the clocks to the system.

If the MCU is woken-up by an interrupt and the fast wake-up feature is enabled (FSTWKP = 1 and SCME = 1), the system will resume operation in self-clock mode after t_{fws}.

A.5.1.5 Pseudo Stop and Wait Recovery

The recovery from pseudo stop and wait are essentially the same since the oscillator was not stopped in both modes. The controller can be woken up by internal or external interrupts. After t_{wrs} the CPU starts fetching the interrupt vector.

A.5.2 Oscillator

The device features an internal low-power loop controlled Pierce oscillator and a full swing Pierce oscillator/external clock mode. The selection of loop controlled Pierce oscillator or full swing Pierce oscillator/external clock depends on the $\overline{\text{XCLKS}}$ signal which is sampled during reset. Before asserting the oscillator to the internal system clocks the quality of the oscillation is checked for each start from either power-on, STOP or oscillator fail. t_{CQOUT} specifies the maximum time before switching to the internal self clock mode after POR or STOP if a proper oscillation is not detected. The quality check also determines the minimum oscillator start-up time t_{UPOSC}. The device also features a clock monitor. A clock monitor failure is asserted if the frequency of the incoming clock signal is below the assert frequency $f_{CMFA.}$

Table A-18. Oscillator Characteristics

Conditions are shown in Table A-4 unless otherwise noted

Num	C	Rating	Symbol	Min	Typ	Max	Unit
1a	C	Crystal oscillator range (loop controlled Pierce)	f_{OSC}	4.0	—	16	MHz
1b	C	Crystal oscillator range (full swing Pierce)[1, 2]	f_{OSC}	0.5	—	40	MHz
2	P	Startup current	i_{OSC}	100	—	—	μA
3	C	Oscillator start-up time (loop controlled Pierce)	t_{UPOSC}	—	—[3]	50[4]	ms
4	D	Clock quality check time-out	t_{CQOUT}	0.45		2.5	s
5	P	Clock monitor failure assert frequency	f_{CMFA}	50	100	200	KHz
6	P	External square wave input frequency	f_{EXT}	0.5	—	80	MHz
7	D	External square wave pulse width low	t_{EXTL}	5	—	—	ns
8	D	External square wave pulse width high	t_{EXTH}	5	—	—	ns
9	D	External square wave rise time	t_{EXTR}	—	—	1	ns
10	D	External square wave fall time	t_{EXTF}	—	—	1	ns
11	D	Input capacitance (EXTAL, XTAL inputs)	C_{IN}	—	7	—	pF
12	P	EXTAL pin input high voltage[5]	$V_{IH,EXTAL}$	$0.75 * V_{DDPLL}$	—	—	V
	T	EXTAL pin input high voltage[5]	$V_{IH,EXTAL}$	—	—	$V_{DDPLL} + 0.3$	V
13	P	EXTAL pin input low voltage[5]	$V_{IL,EXTAL}$	—	—	$0.25 * V_{DDPLL}$	V
	T	EXTAL pin input low voltage[5]	$V_{IL,EXTAL}$	$V_{SSPLL} - 0.3$	—	—	V
14	C	EXTAL pin input hysteresis[5]	$V_{HYS,EXTAL}$	—	250	—	mV

[1] Depending on the crystal a damping series resistor might be necessary

[2] $\overline{XCLKS}$ = 0

[3] f_{osc} = 4 MHz, C = 22 pF.

[4] Maximum value is for extreme cases using high Q, low frequency crystals

[5] If full swing Pierce oscillator/external clock circuitry is used. ($\overline{XCLKS}$ = 0)

A.5.3 Phase Locked Loop

The oscillator provides the reference clock for the PLL. The PLL´s voltage controlled oscillator (VCO) is also the system clock source in self clock mode.

A.5.3.1 XFC Component Selection

This section describes the selection of the XFC components to achieve a good filter characteristics.

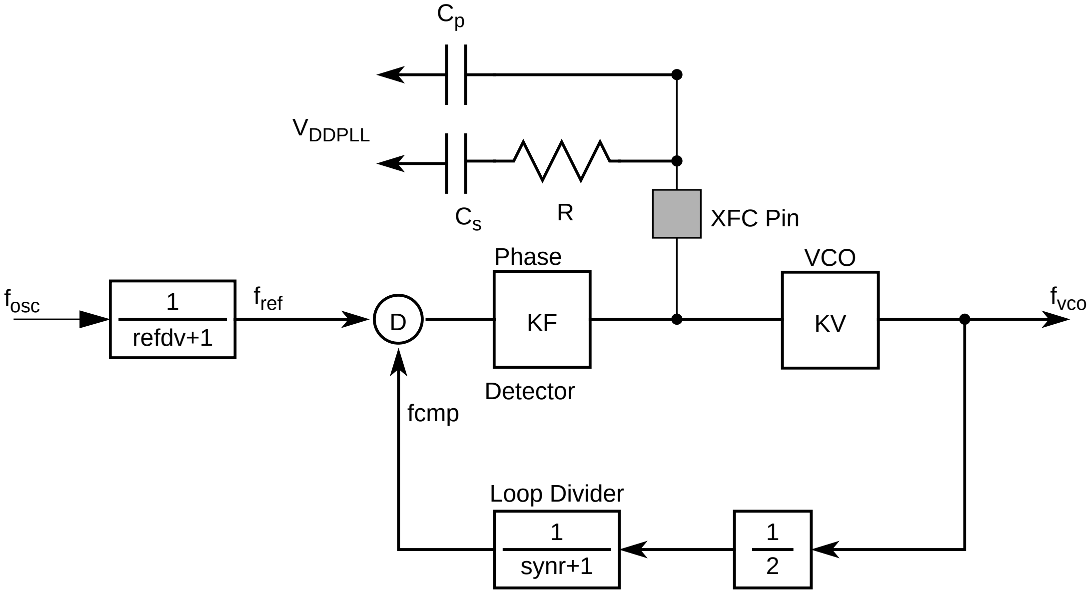


Figure A-3. Basic PLL Functional Diagram

The following procedure can be used to calculate the resistance and capacitance values using typical values for K_1, f_1 and i_{ch} from Table A-19.

The grey boxes show the calculation for f_{VCO} = 80 MHz and f_{ref} = 4 MHz. For example, these frequencies are used for f_{OSC} = 4-MHz and a 40-MHz bus clock.

The VCO gain at the desired VCO frequency is approximated by:

$$K_V = K_1 \cdot e^{\frac{(f_1 - f_{vco})}{K_1 \cdot 1V}} \quad = -195\text{MHz/V} \cdot e^{\frac{126 - 80}{-195}} \quad = -154.0\text{MHz/V}$$

The phase detector relationship is given by:

$$K_\Phi = -|i_{ch}| \cdot K_V = -3.5\mu A \cdot (-154\text{MHz/V}) = 539.1\text{Hz}/\Omega$$

i_{ch} is the current in tracking mode.

The loop bandwidth f_C should be chosen to fulfill the Gardner’s stability criteria by at least a factor of 10, typical values are 50. $\zeta = 0.9$ ensures a good transient response.

$$f_C < \frac{2 \cdot \zeta \cdot f_{ref}}{\pi \cdot \left(\zeta + \sqrt{1+\zeta^2}\right)} \cdot \frac{1}{10} \rightarrow f_C < \frac{f_{ref}}{4 \cdot 10}; (\zeta = 0.9) \quad \boxed{f_C < 100\text{kHz}}$$

And finally the frequency relationship is defined as

$$n = \frac{f_{VCO}}{f_{ref}} = 2 \cdot (synr + 1) \quad \boxed{= 20}$$

With the above values the resistance can be calculated. The example is shown for a loop bandwidth $f_C = 20$ kHz:

$$R = \frac{2 \cdot \pi \cdot n \cdot f_C}{K_\Phi} = \boxed{\frac{2 \cdot \pi \cdot 20 \cdot 20\text{kHz}}{(539.1\text{Hz})/\Omega} = 4.7\text{k}\Omega}$$

The capacitance C_s can now be calculated as:

$$C_s = \frac{2 \cdot \zeta^2}{\pi \cdot f_C \cdot R} = \frac{0.516}{f_C \cdot R}; (\zeta = 0.9) \quad \boxed{= 5.5\text{nF} = \sim 4.7\text{nF}}$$

The capacitance C_p should be chosen in the range of:

$$\frac{C_s}{20} \le C_p \le \frac{C_s}{10} \quad \boxed{C_P = 470\text{pF}}$$

A.5.3.2 Jitter Information

The basic functionality of the PLL is shown in Figure A-3. With each transition of the clock f_{cmp}, the deviation from the reference clock f_{ref} is measured and input voltage to the VCO is adjusted accordingly.The adjustment is done continuously with no abrupt changes in the clock output frequency. Noise, voltage, temperature and other factors cause slight variations in the control loop resulting in a clock jitter. This jitter affects the real minimum and maximum clock periods as illustrated in Figure A-4.

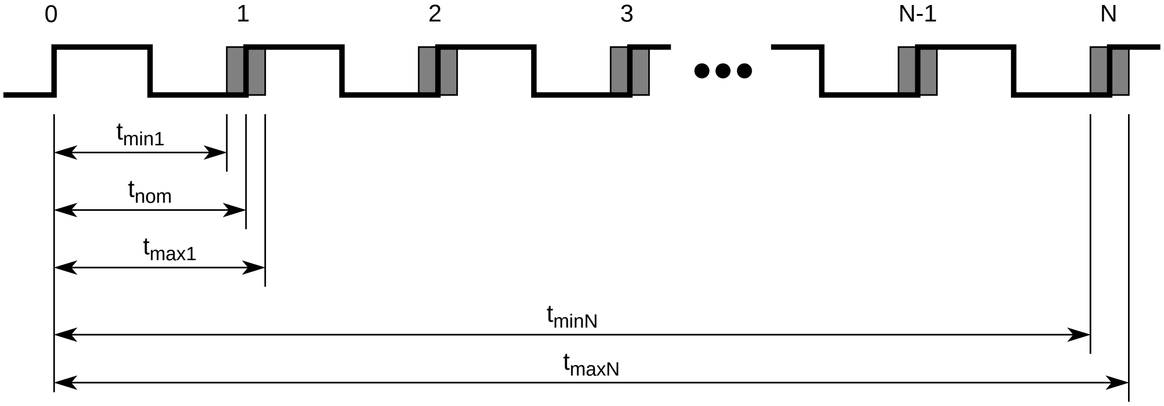


Figure A-4. Jitter Definitions

The relative deviation of t_{nom} is at its maximum for one clock period, and decreases towards zero for larger number of clock periods (N).

Defining the jitter as:

$$J(N) = \max\left(\left|1 - \frac{t_{max}(N)}{N \cdot t_{nom}}\right|, \left|1 - \frac{t_{min}(N)}{N \cdot t_{nom}}\right|\right)$$

For N < 1000, the following equation is a good fit for the maximum jitter:

$$J(N) = \frac{j_1}{\sqrt{N}} + j_2$$

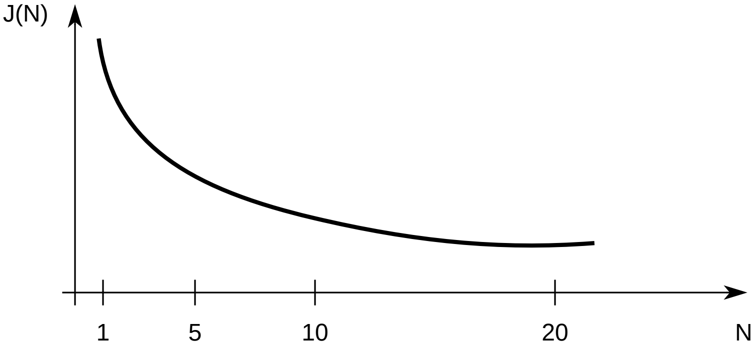


Figure A-5. Maximum Bus Clock Jitter Approximation

This is very important to notice with respect to timers, serial modules where a prescaler will eliminate the effect of the jitter to a large extent.

Table A-19. PLL Characteristics

Conditions are shown in Table A-4 unless otherwise noted							
Num	**C**	**Rating**	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
1	P	Self clock mode frequency	f_{SCM}	1	—	5.5	MHz
2	D	VCO locking range	f_{VCO}	8	—	80	MHz
3	D	Lock detector transition from acquisition to tracking mode	$\|\Delta_{trk}\|$	3	—	4	%[1]
4	D	Lock detection	$\|\Delta_{Lock}\|$	0	—	1.5	%[1]
5	D	Unlock detection	$\|\Delta_{unl}\|$	0.5	—	2.5	%[1]
6	D	Lock detector transition from tracking to acquisition mode	$\|\Delta_{unt}\|$	6	—	8	%[1]
7	C	PLLON total stabilization delay (auto mode)[2]	t_{stab}	—	0.24	—	ms
8	D	PLLON acquisition mode stabilization delay[2]	t_{acq}	—	0.09	—	ms
9	D	PLLON tracking mode stabilization delay[2]	t_{al}	—	0.16	—	ms
10	D	Fitting parameter VCO loop gain	K_1	—	–195	—	MHz/V
11	D	Fitting parameter VCO loop frequency	f_1	—	126	—	MHz
12	D	Charge pump current acquisition mode	$\| i_{ch} \|$	—	38.5	—	µA
13	D	Charge pump current tracking mode	$\| i_{ch} \|$	—	3.5	—	µA
14	C	Jitter fit parameter 1[2]	j_1	—	0.9	1.3	%
15	C	Jitter fit parameter 2[2]	j_2	—	0.02	0.12	%

[1] % deviation from target frequency

[2] f_{osc} = 4 MHz, f_{BUS} = 40 MHz equivalent f_{VCO} = 80 MHz: REFDV = #$00, SYNR = #$09, C_S = 4.7 nF, C_P = 470 pF, R_S = 4.7 kΩ

A.6 LCD

Table A-20. LCD Driver Electrical Characteristics

Characteristic	Symbol	Min.	Typ.	Max.	Unit
LCD Supply Voltage	VLCD	-0.25	-	VDDX + 0.25	V
LCD Output Impedance(BP[3:0],FP[31:0]) for outputs to charge to higher voltage level or to GND [1]	$Z_{BP/FP}$	-	-	5.0	kΩ
LCD Output Current (BP[3:0],FP[31:0]) for outputs to discharge to lower voltage level except GND [2]	$I_{BP/FP}$	50	-	-	uA

[1] Outputs measured one at a time, low impedance voltage source connected to the VLCD pin.

[2] Outputs measured one at a time, low impedance voltage source connected to the VLCD pin.

A.7 MSCAN

Table A-21. MSCAN Wake-up Pulse Characteristics

Conditions are shown in Table A-4 unless otherwise noted							
Num	**C**	**Rating**	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
1	P	MSCAN wakeup dominant pulse filtered	t_{WUP}	—	—	2	µs
2	P	MSCAN wakeup dominant pulse pass	t_{WUP}	5	—	—	µs

A.8 SPI Timing

This section provides electrical parametrics and ratings for the SPI. In Table A-22 the measurement conditions are listed.

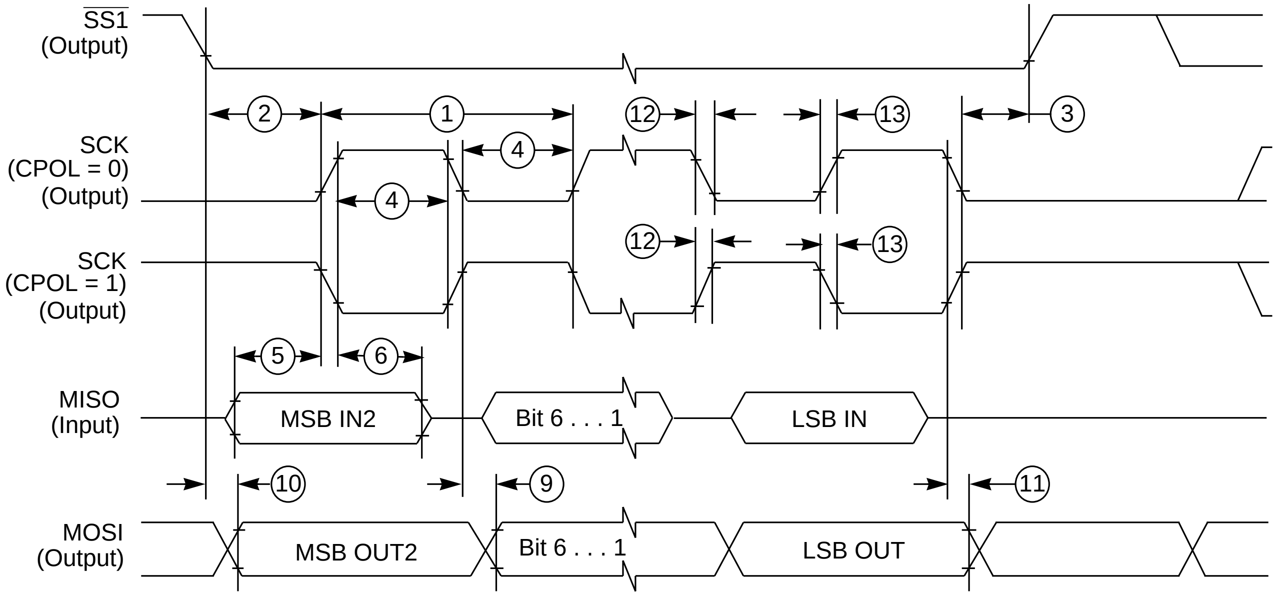
Table A-22. Measurement Conditions

Description	Value	Unit
Drive mode	Full drive mode	—
Load capacitance C_{LOAD}[1], on all outputs	50	pF
Thresholds for delay measurement points	(20% / 80%) V_{DDX}	V

[1] Timing specified for equal load on all SPI output pins. Avoid asymmetric load.

A.8.1 Master Mode

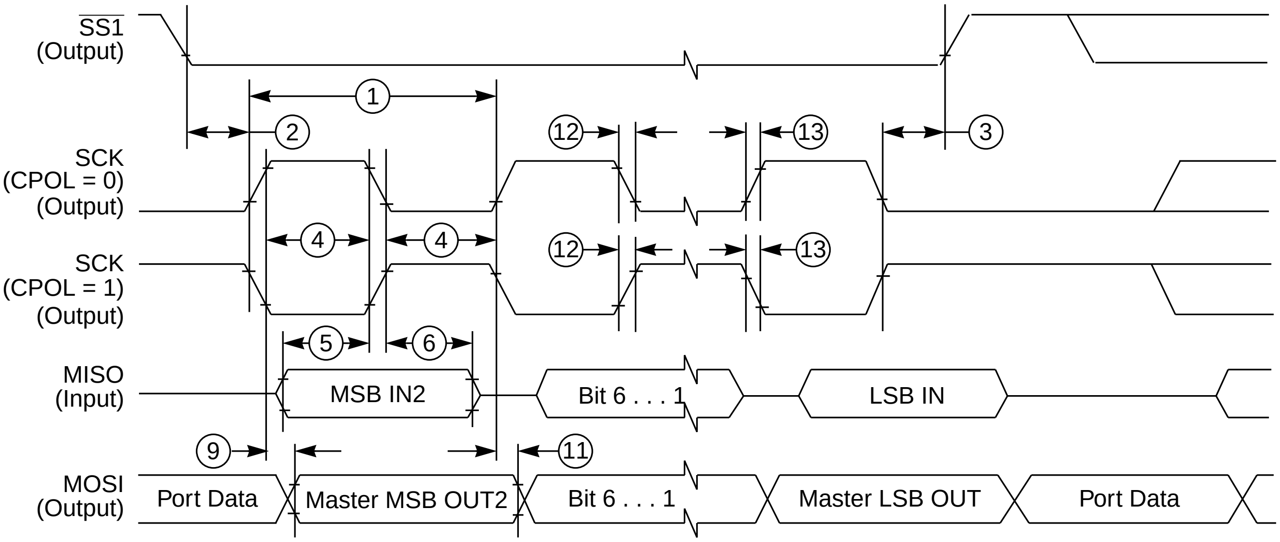
In Figure A-6 the timing diagram for master mode with transmission format CPHA = 0 is depicted.



1. If configured as an output.
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

Figure A-6. SPI Master Timing (CPHA = 0)

In Figure A-7 the timing diagram for master mode with transmission format CPHA=1 is depicted.



1.If configured as output

2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

Figure A-7. SPI Master Timing (CPHA = 1)

In Table A-23 the timing characteristics for master mode are listed.

Table A-23. SPI Master Mode Timing Characteristics

Num	C	Characteristic	Symbol	Min	Typ	Max	Unit
1	D	SCK frequency	f_{sck}	1/2048	—	1/2	f_{bus}
1	D	SCK period	t_{sck}	2	—	2048	t_{bus}
2	D	Enable lead time	t_{lead}	—	1/2	—	t_{sck}
3	D	Enable lag time	t_{lag}	—	1/2	—	t_{sck}
4	D	Clock (SCK) high or low time	t_{wsck}	—	1/2	—	t_{sck}
5	D	Data setup time (inputs)	t_{su}	8	—	—	ns
6	D	Data hold time (inputs)	t_{hi}	8	—	—	ns
9	D	Data valid after SCK edge	t_{vsck}	—	—	29	ns
10	D	Data valid after $\overline{SS}$ fall (CPHA = 0)	t_{vss}	—	—	15	ns
11	D	Data hold time (outputs)	t_{ho}	20	—	—	ns
12	D	Rise and fall time inputs	t_{rfi}	—	—	8	ns
13	D	Rise and fall time outputs	t_{rfo}	—	—	8	ns

A.8.2 Slave Mode

In Figure A-8 the timing diagram for slave mode with transmission format CPHA = 0 is depicted.

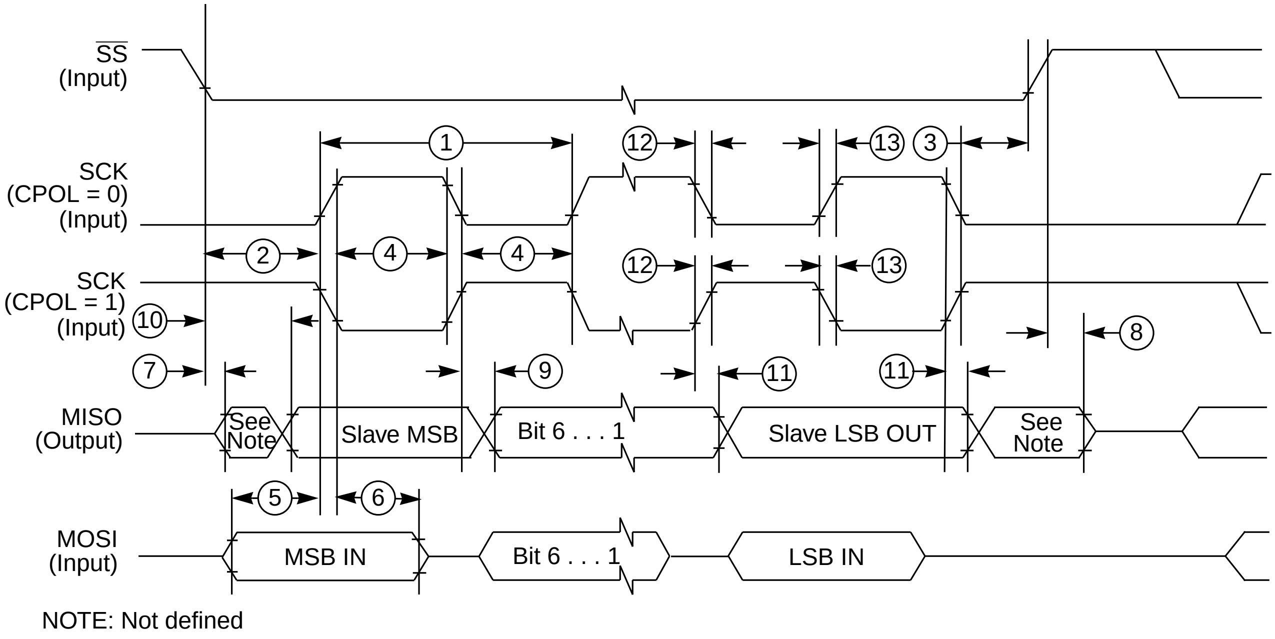


Figure A-8. SPI Slave Timing (CPHA = 0)

In Figure A-9 the timing diagram for slave mode with transmission format CPHA = 1 is depicted.

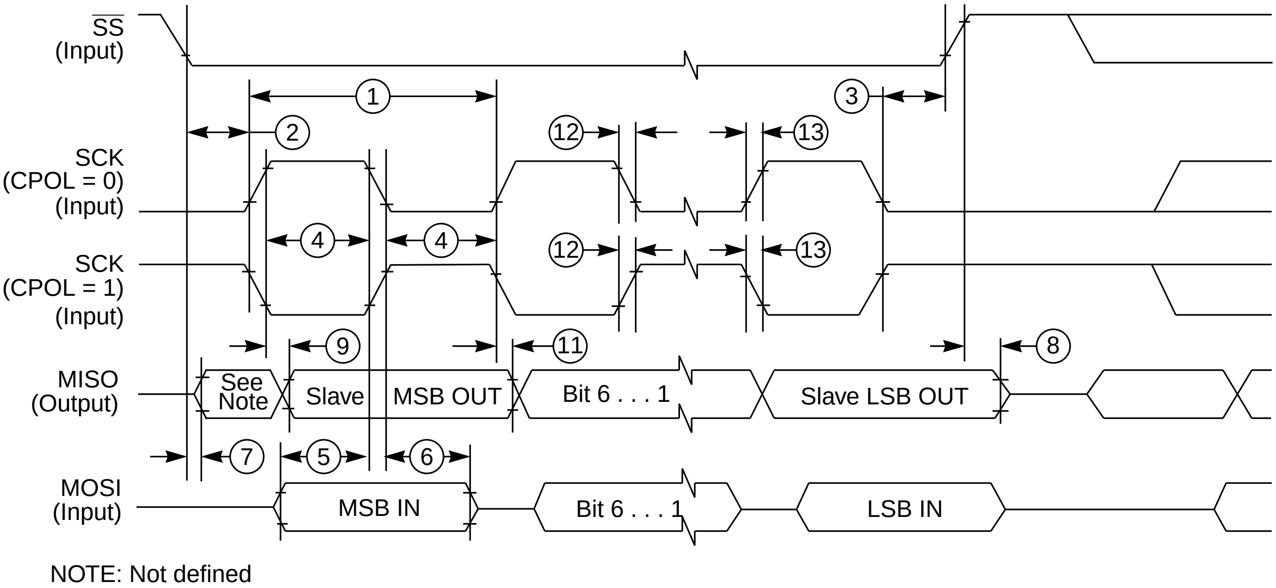


Figure A-9. SPI Slave Timing (CPHA = 1)

In Table A-24 the timing characteristics for slave mode are listed.

Table A-24. SPI Slave Mode Timing Characteristics

Num	C	Characteristic	Symbol	Min	Typ	Max	Unit
1	D	SCK frequency	f_{sck}	DC	—	1/4	f_{bus}
1	D	SCK period	t_{sck}	4	—	∞	t_{bus}
2	D	Enable lead time	t_{lead}	4	—	—	t_{bus}
3	D	Enable lag time	t_{lag}	4	—	—	t_{bus}
4	D	Clock (SCK) high or low time	t_{wsck}	4	—	—	t_{bus}
5	D	Data setup time (inputs)	t_{su}	8	—	—	ns
6	D	Data hold time (inputs)	t_{hi}	8	—	—	ns
7	D	Slave access time (time to data active)	t_a	—	—	20	ns
8	D	Slave MISO disable time	t_{dis}	—	—	22	ns
9	D	Data valid after SCK edge	t_{vsck}	—	—	$29 + 0.5 \cdot t_{bus}$[1]	ns
10	D	Data valid after $\overline{SS}$ fall	t_{vss}	—	—	$29 + 0.5 \cdot t_{bus}$[1]	ns
11	D	Data hold time (outputs)	t_{ho}	20	—	—	ns
12	D	Rise and fall time inputs	t_{rfi}	—	—	8	ns
13	D	Rise and fall time outputs	t_{rfo}	—	—	8	ns

[1] 0.5 t_{bus} added due to internal synchronization delay

A.9 External Bus Timing

The following conditions are assumed for all following external bus timing values:

- Crystal input within 45% to 55% duty
- Equal loads of pins
- Pad full drive (reduced drive must be off)

A.9.1 Normal Expanded Mode (External Wait Feature Disabled)

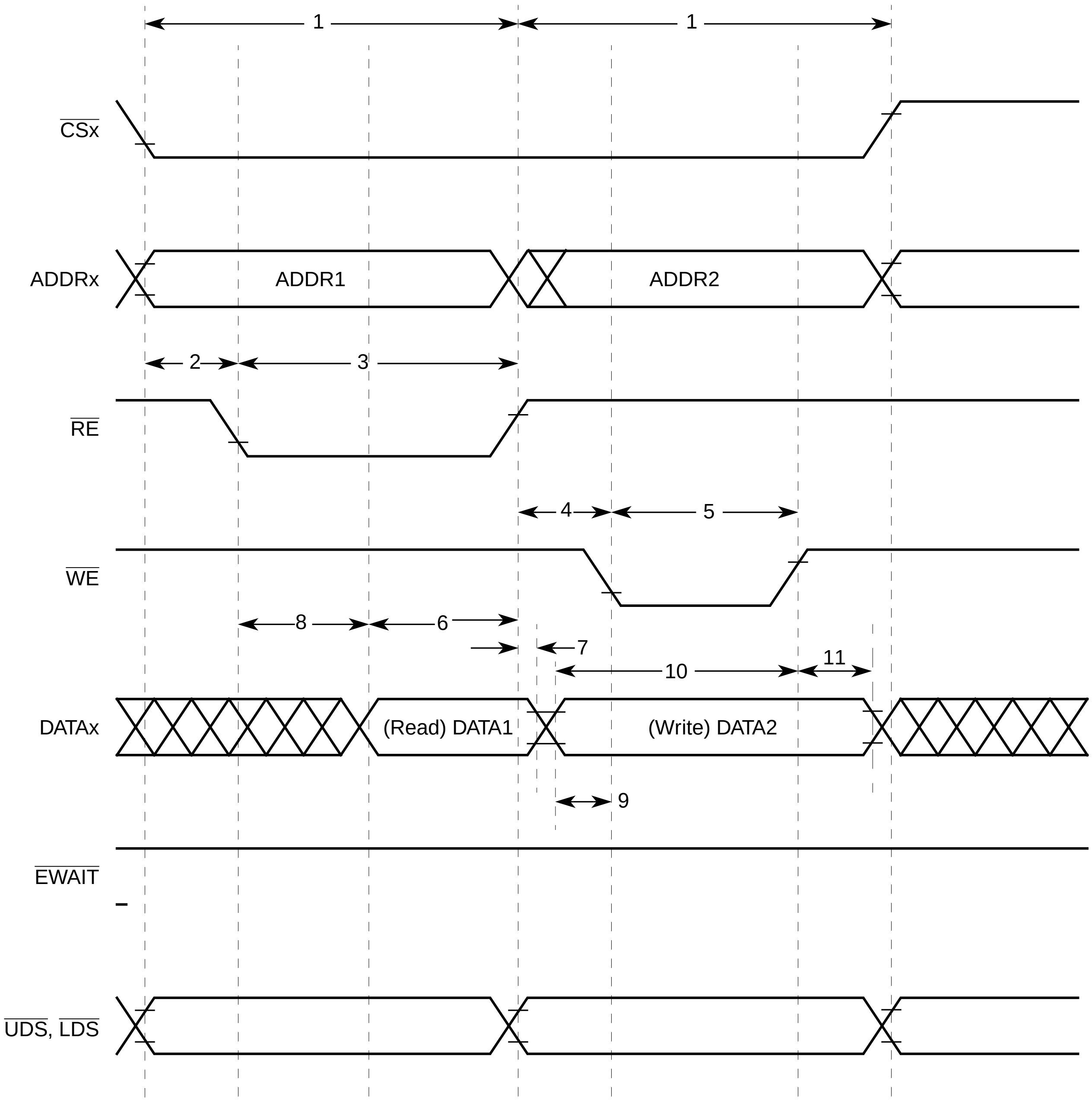


Figure A-10. Example 1a: Normal Expanded Mode — Read Followed by Write

Table A-25. Example 1a: Normal Expanded Mode Timing V_{DD5} = 5.0 V ($\overline{\text{EWAITE}}$ = 0)

No.	C	Characteristic	Symbol	Min	Max	Unit
—	—	Frequency of internal bus	f_i	D.C.	40.0	MHz
—	—	Internal cycle time	t_{cyc}	25	∞	ns
—	—	Frequency of external bus	f_o	D.C.	20.0	MHz
1	—	External cycle time (selected by EXSTR)	t_{cyce}	50	∞	ns
2	D	Address[1] valid to $\overline{\text{RE}}$ fall	t_{ADRE}	5	—	ns
3	D	Pulse width, $\overline{\text{RE}}$	PW_{RE}	35	—	ns
4	D	Address[1] valid to $\overline{\text{WE}}$ fall	t_{ADWE}	5	—	ns
5	D	Pulse width, $\overline{\text{WE}}$	PW_{WE}	23	—	ns
6	D	Read data setup time (if ITHRS = 0)	t_{DSR}	24	—	ns
	D	Read data setup time (if ITHRS = 1)	t_{DSR}	28	—	ns
7	D	Read data hold time	t_{DHR}	0	—	ns
8	D	Read enable access time	t_{ACCR}	11	—	ns
9	D	Write data valid to $\overline{\text{WE}}$ fall	t_{WDWE}	7	—	ns
10	D	Write data setup time	t_{DSW}	31	—	ns
11	D	Write data hold time	t_{DHW}	8	—	ns

[1] Includes the following signals: ADDRx, $\overline{\text{UDS}}$, $\overline{\text{LDS}}$, and $\overline{\text{CSx}}$.

A.9.2 Normal Expanded Mode (External Wait Feature Enabled)

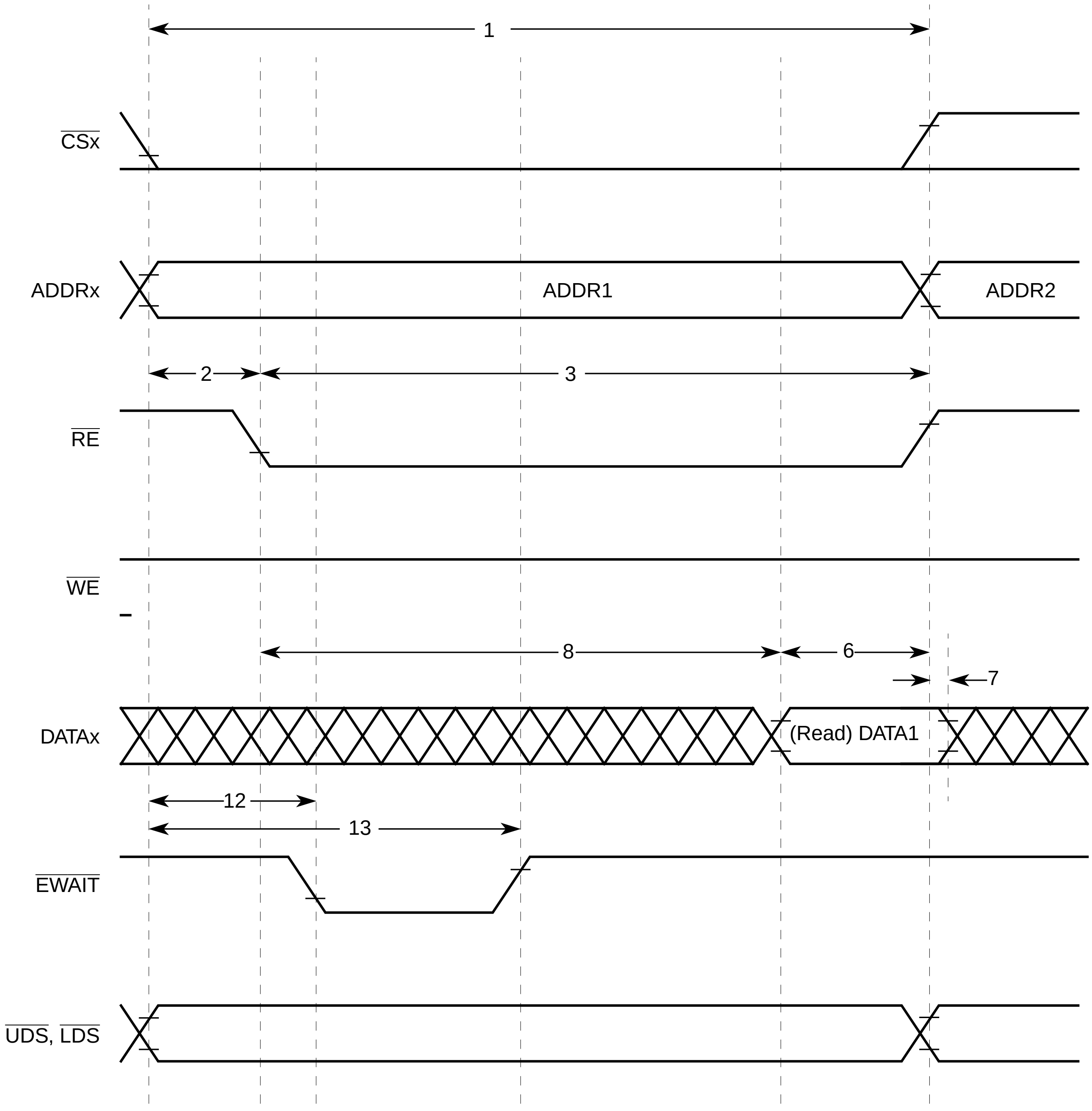


Figure A-11. Example 1b: Normal Expanded Mode — Stretched Read Access

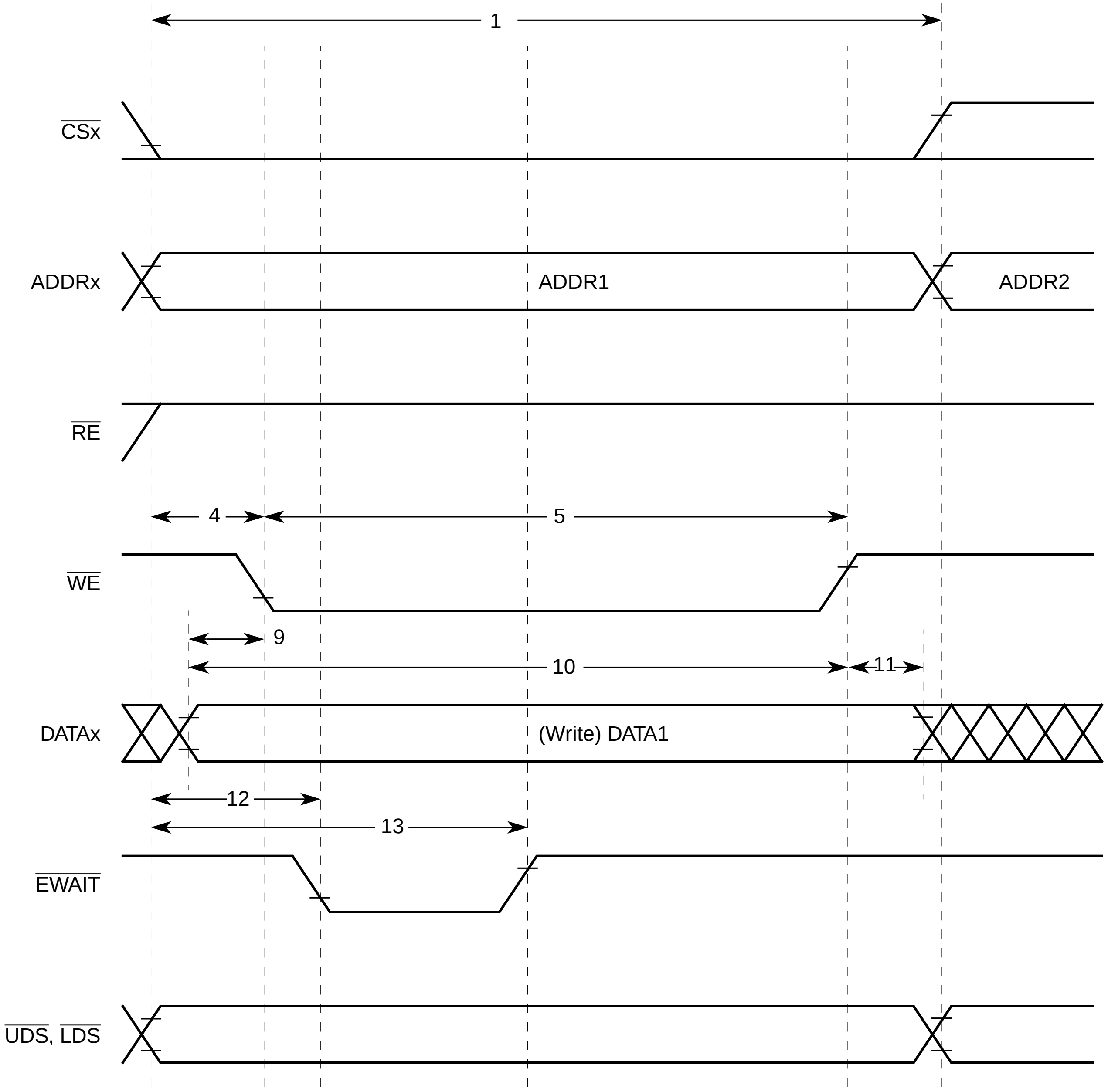


Figure A-12. Example 1b: Normal Expanded Mode — Stretched Write Access

Table A-26. Example 1b: Normal Expanded Mode Timing V_{DD5} = 5.0 V ($\overline{\text{EWAITE}}$ = 1)

No.	C	Characteristic	Symbol	2 Stretch Cycles		3 Stretch Cycles		Unit
				Min	Max	Min	Max	
—	—	Frequency of internal bus	f_i	D.C.	40.0	D.C.	40.0	MHz
—	—	Internal cycle time	t_{cyc}	25	∞	25	∞	ns
—	—	Frequency of external bus	f_o	D.C.	13.3	D.C.	10.0	MHz
—	—	External cycle time (selected by EXSTR)	t_{cyce}	75	∞	100	∞	ns
1	—	External cycle time (EXSTR+1EWAIT)	t_{cycew}	100	∞	125	∞	ns
2	D	Address[1] valid to $\overline{\text{RE}}$ fall	t_{ADRE}	5	—	5	—	ns
3	D	Pulse width, $\overline{\text{RE}}$ [2]	PW_{RE}	85	—	110	—	ns
4	D	Address[1] valid to $\overline{\text{WE}}$ fall	t_{ADWE}	5	—	5	—	ns
5	D	Pulse width, $\overline{\text{WE}}$[2]	PW_{WE}	73	—	98	—	ns
6	D	Read data setup time (if ITHRS = 0)	t_{DSR}	24	—	24	—	ns
	D	Read data setup time (if ITHRS = 1)	t_{DSR}	28	—	28	—	ns
7	D	Read data hold time	t_{DHR}	0	—	0	—	ns
8	D	Read enable access time	t_{ACCR}	71	—	86	—	ns
9	D	Write data valid to $\overline{\text{WE}}$ fall	t_{WDWE}	7	—	7	—	ns
10	D	Write data setup time	t_{DSW}	81	—	106	—	ns
11	D	Write data hold time	t_{DHW}	8	—	8	—	ns
12	D	Address to $\overline{\text{EWAIT}}$ fall	t_{ADWF}	0	20	0	45	ns
13	D	Address to $\overline{\text{EWAIT}}$ rise	t_{ADWR}	37	47	62	72	ns

[1] Includes the following signals: ADDRx, $\overline{\text{UDS}}$, $\overline{\text{LDS}}$, and $\overline{\text{CSx}}$.

[2] Affected by $\overline{\text{EWAIT}}$.

A.9.3 Emulation Single-Chip Mode (Without Wait States)

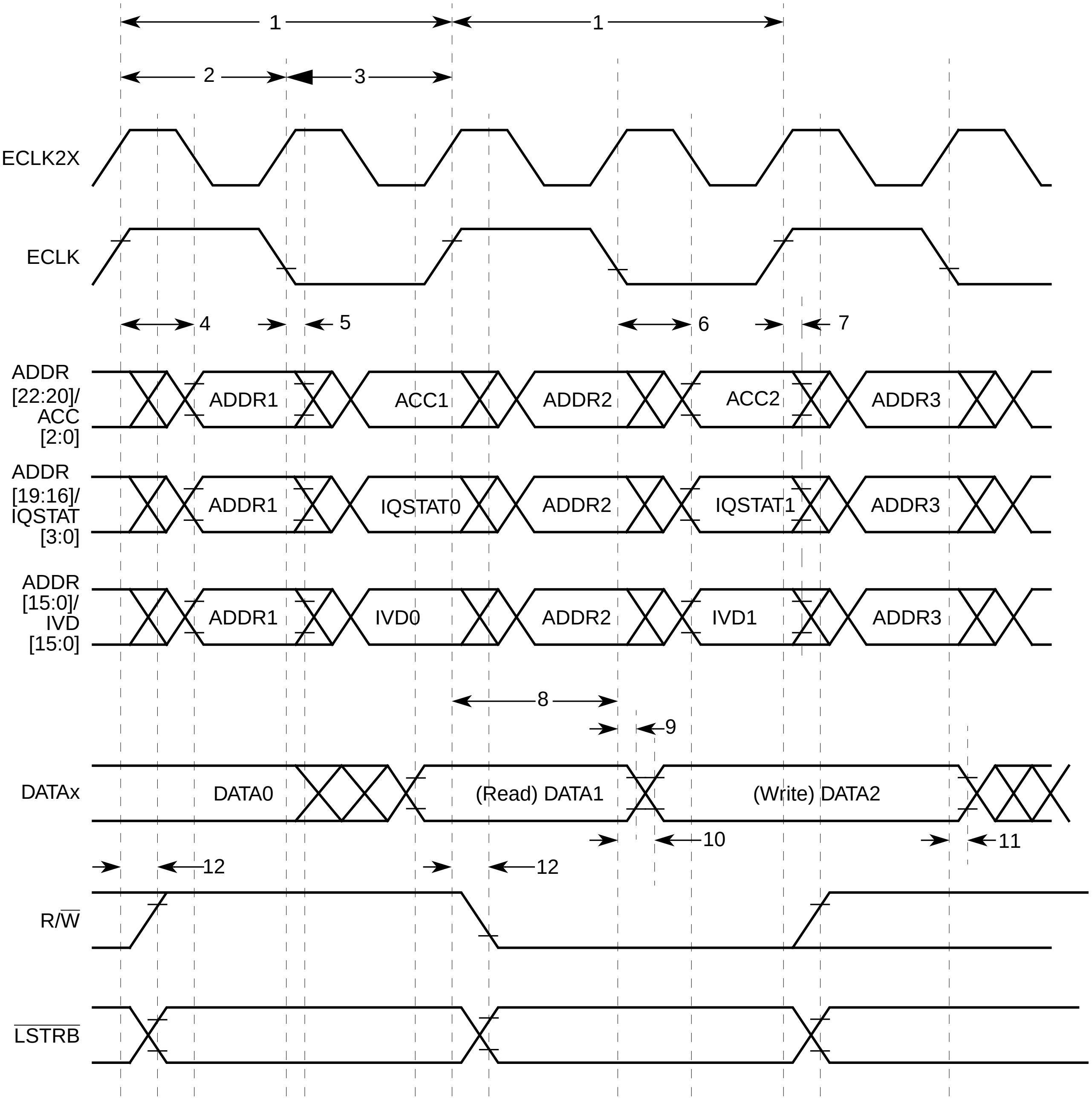


Figure A-13. Example 2a: Emulation Single-Chip Mode — Read Followed by Write

Table A-27. Example 2a: Emulation Single-Chip Mode Timing V_{DD5} = 5.0 V ($\overline{\text{EWAITE}}$ = 0)

No.	C	Characteristic[1]	Symbol	Min	Max	Unit
—	—	Frequency of internal bus	f_i	D.C.	40.0	MHz
1	—	Cycle time	t_{cyc}	25	∞	ns
2	D	Pulse width, E high	PW_{EH}	11.5	—	ns
3	D	Pulse width, E low	PW_{EL}	11.5	—	ns
4	D	Address delay time	t_{AD}	—	5	ns
5	D	Address hold time	t_{AH}	0	—	ns
6	D	IVDx delay time[2]	t_{IVDD}	—	4.5	ns
7	D	IVDx hold time[2]	t_{IVDH}	0	—	ns
8	D	Read data setup time (ITHRS = 1 only)	t_{DSR}	12	—	ns
9	D	Read data hold time	t_{DHR}	0	—	ns
10	D	Write data delay time	t_{DDW}	—	5	ns
11	D	Write data hold time	t_{DHW}	0	—	ns
12	D	Read/write data delay time[3]	t_{RWD}	–1	5	ns

[1] Typical supply and silicon, room temperature only

[2] Includes also ACCx, IQSTATx

[3] Includes $\overline{\text{LSTRB}}$

A.9.4 Emulation Expanded Mode (With Optional Access Stretching)

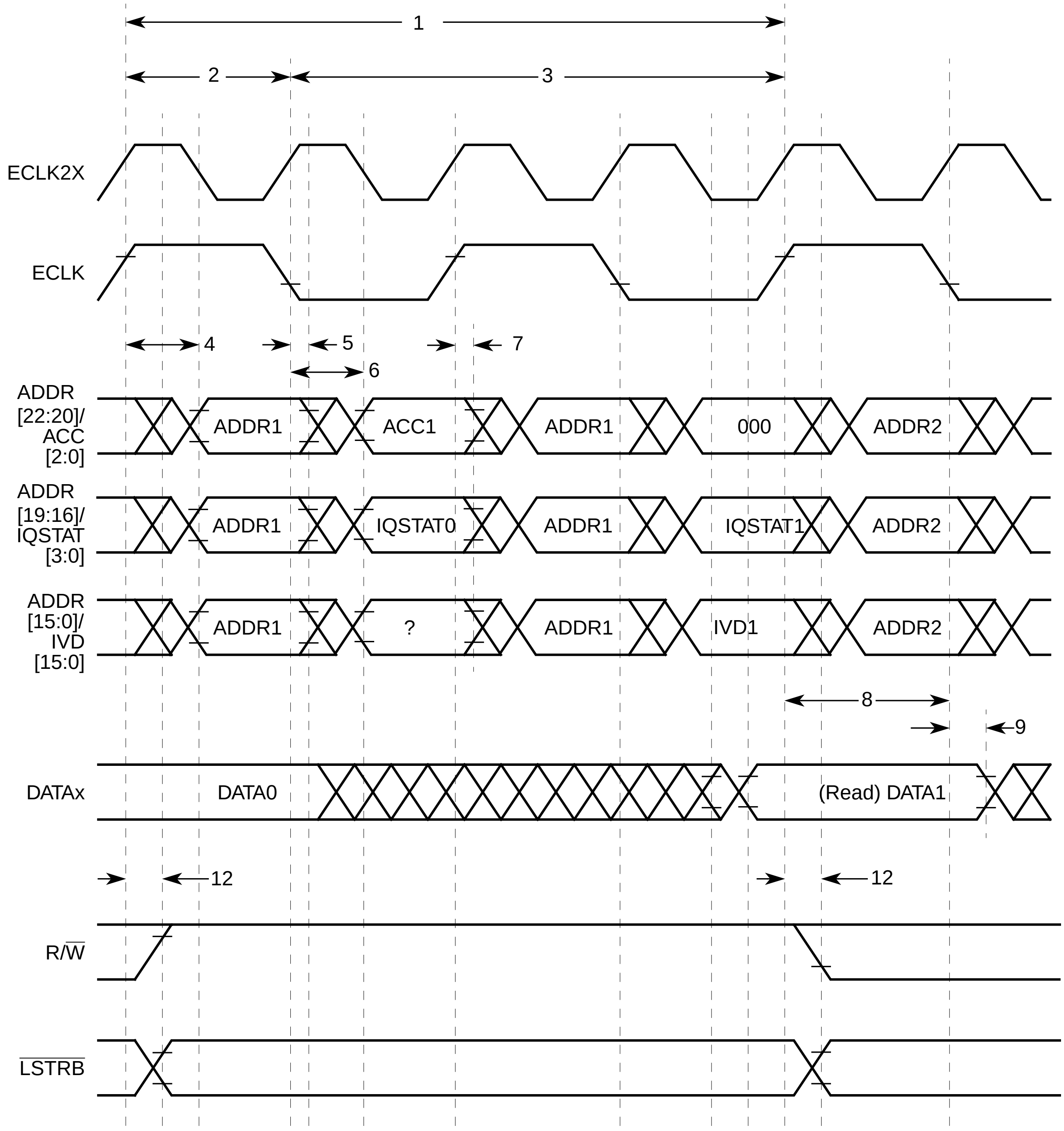


Figure A-14. Example 2b: Emulation Expanded Mode — Read with 1 Stretch Cycle

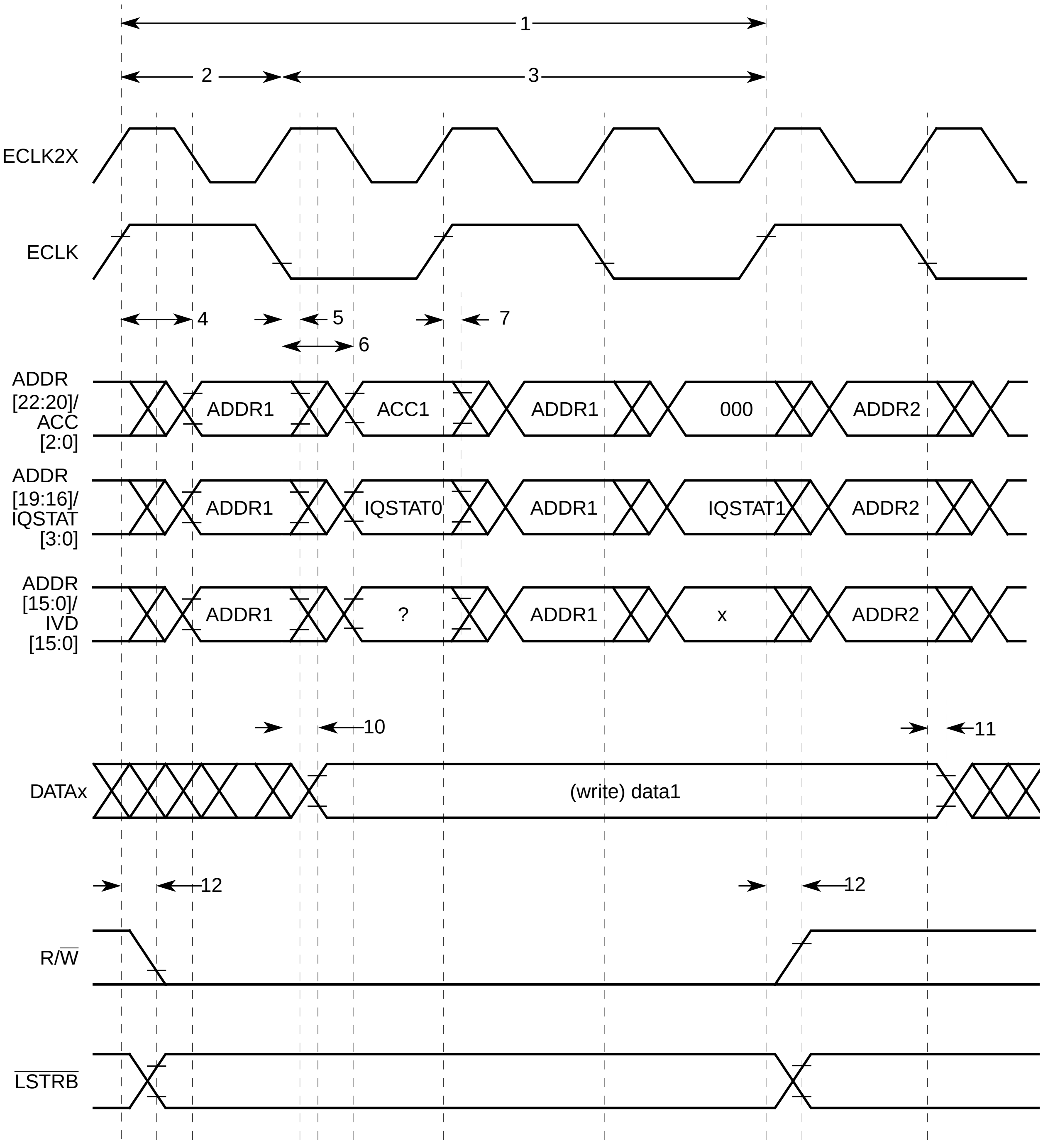


Figure A-15. Example 2b: Emulation Expanded Mode — Write with 1 Stretch Cycle

Table A-28. Example 2b: Emulation Expanded Mode Timing V_{DD5} = 5.0 V ($\overline{\text{EWAITE}}$ = 0)

No.	C	Characteristic[1]	Symbol	1 Stretch Cycle		2 Stretch Cycles		3 Stretch Cycles		Unit
				Min	Max	Min	Max	Min	Max	
—	—	Internal cycle time	t_{cyc}	25	25	25	25	25	25	ns
1	—	Cycle time	t_{cyce}	50	∞	75	∞	100	∞	ns
2	D	Pulse width, E high	PW_{EH}	11.5	14	11.5	14	11.5	14	ns
3	D	E falling to sampling E rising	t_{EFSR}	35	39.5	60	64.5	85	89.5	ns
4	D	Address delay time	t_{AD}	—	5	—	5	—	5	ns
5	D	Address hold time	t_{AH}	0	—	0	—	0	—	ns
6	D	IVD delay time[2]	t_{IVDD}	—	4.5	—	4.5	—	4.5	ns
7	D	IVD hold time[2]	t_{IVDH}	0	—	0	—	0	—	ns
8	D	Read data setup time	t_{DSR}	12	—	12	—	12	—	ns
9	D	Read data hold time	t_{DHR}	0	—	0	—	0	—	ns
10	D	Write data delay time	t_{DDW}	—	5	—	5	—	5	ns
11	D	Write data hold time	t_{DHW}	0	—	0	—	0	—	ns
12	D	Read/write data delay time[3]	t_{RWD}	–1	5	–1	5	–1	5	ns

[1] Typical supply and silicon, room temperature only

[2] Includes also ACCx, IQSTATx

[3] Includes $\overline{\text{LSTRB}}$

A.9.5 External Tag Trigger Timing

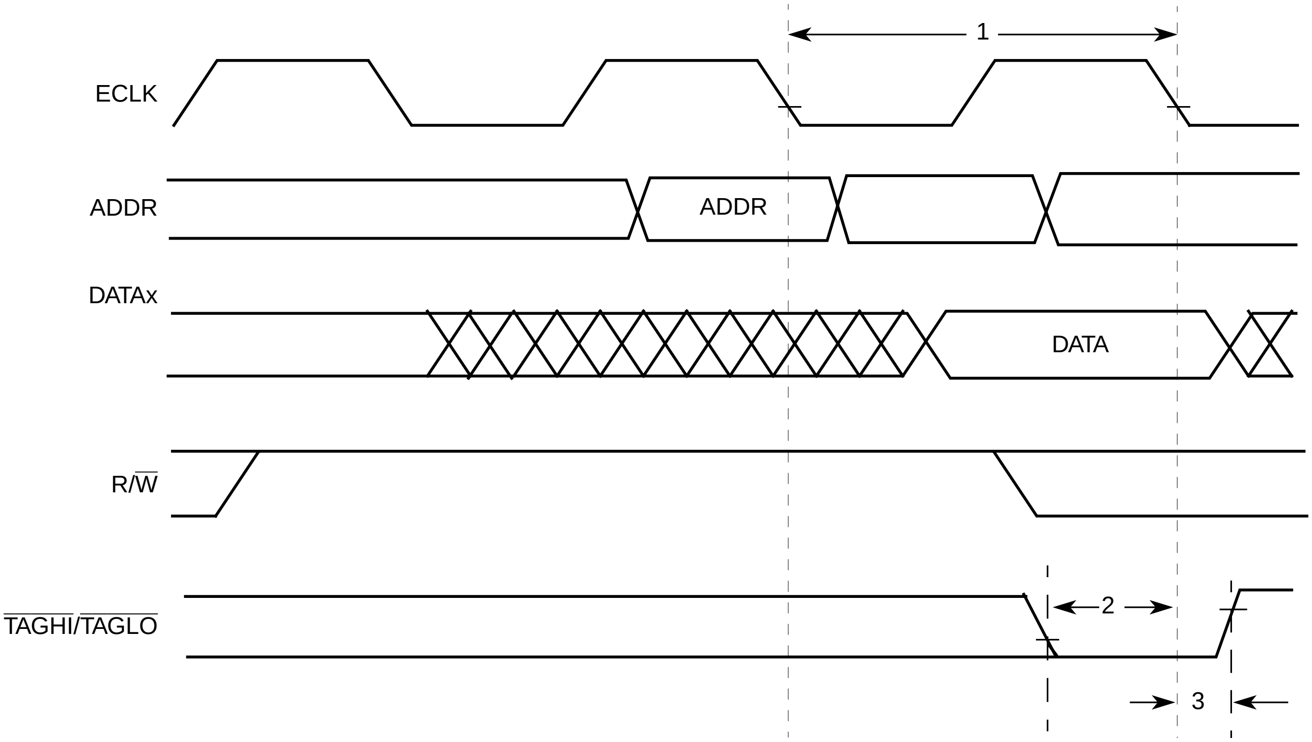


Figure A-16. External Trigger Timing

Table A-29. External Tag Trigger Timing V_{DD5} = 5.0 V

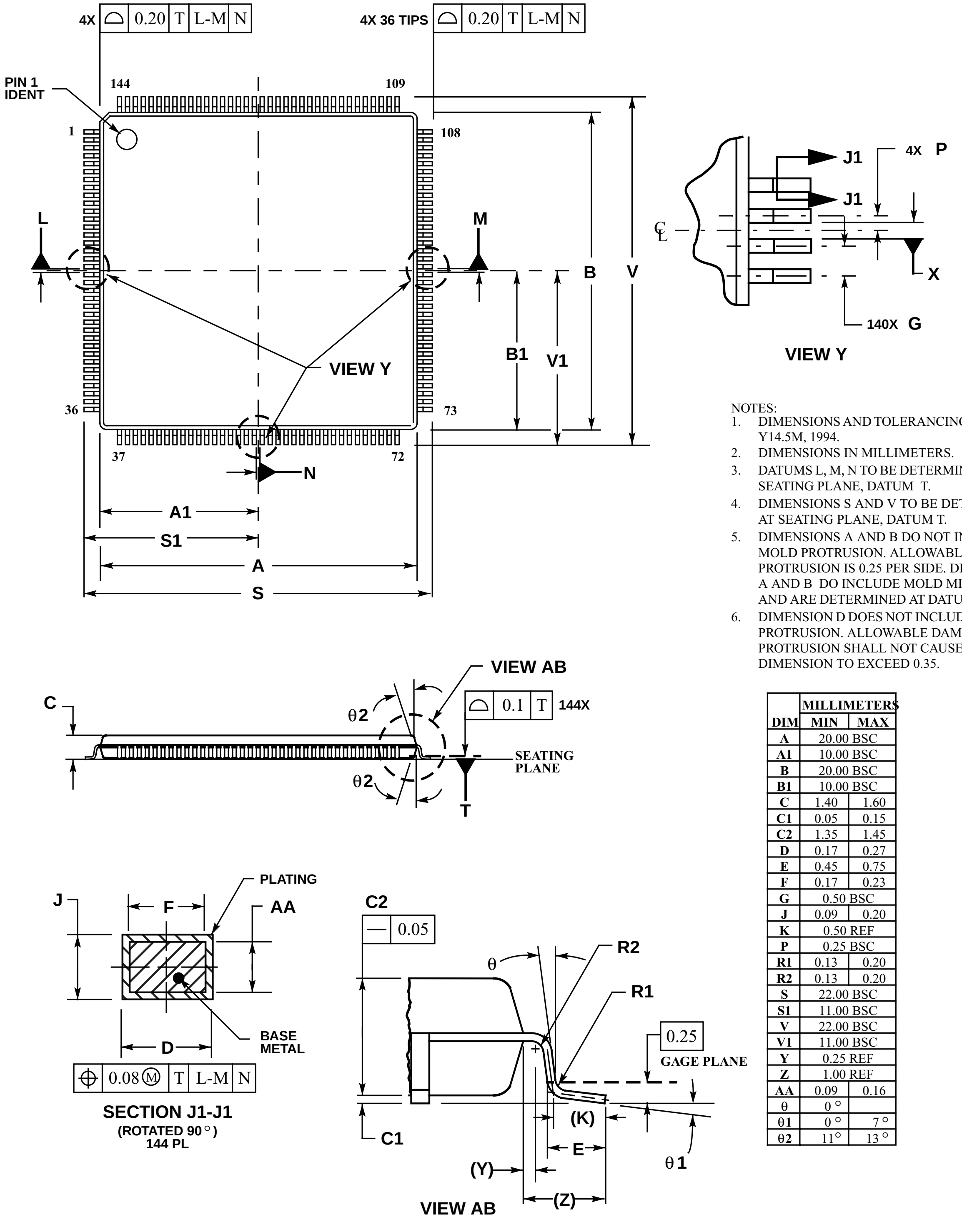
No.	C	Characteristic [1]	Symbol	Min	Max	Unit
1	D	Frequency of internal bus	f_i	D.C.	40.0	MHz
2	D	Cycle time	t_{cyc}	25	∞	ns
3	D	$\overline{TAGHI}$/$\overline{TAGLO}$ setup time	t_{TS}	11.5	—	ns
4	D	$\overline{TAGHI}$/$\overline{TAGLO}$ hold time	t_{TH}	0	—	ns

[1] Typical supply and silicon, room temperature only

Appendix B
Package Information

This section provides the physical dimensions of the MC9S12XHZ512 packages.

B.1 144-Pin LQFP

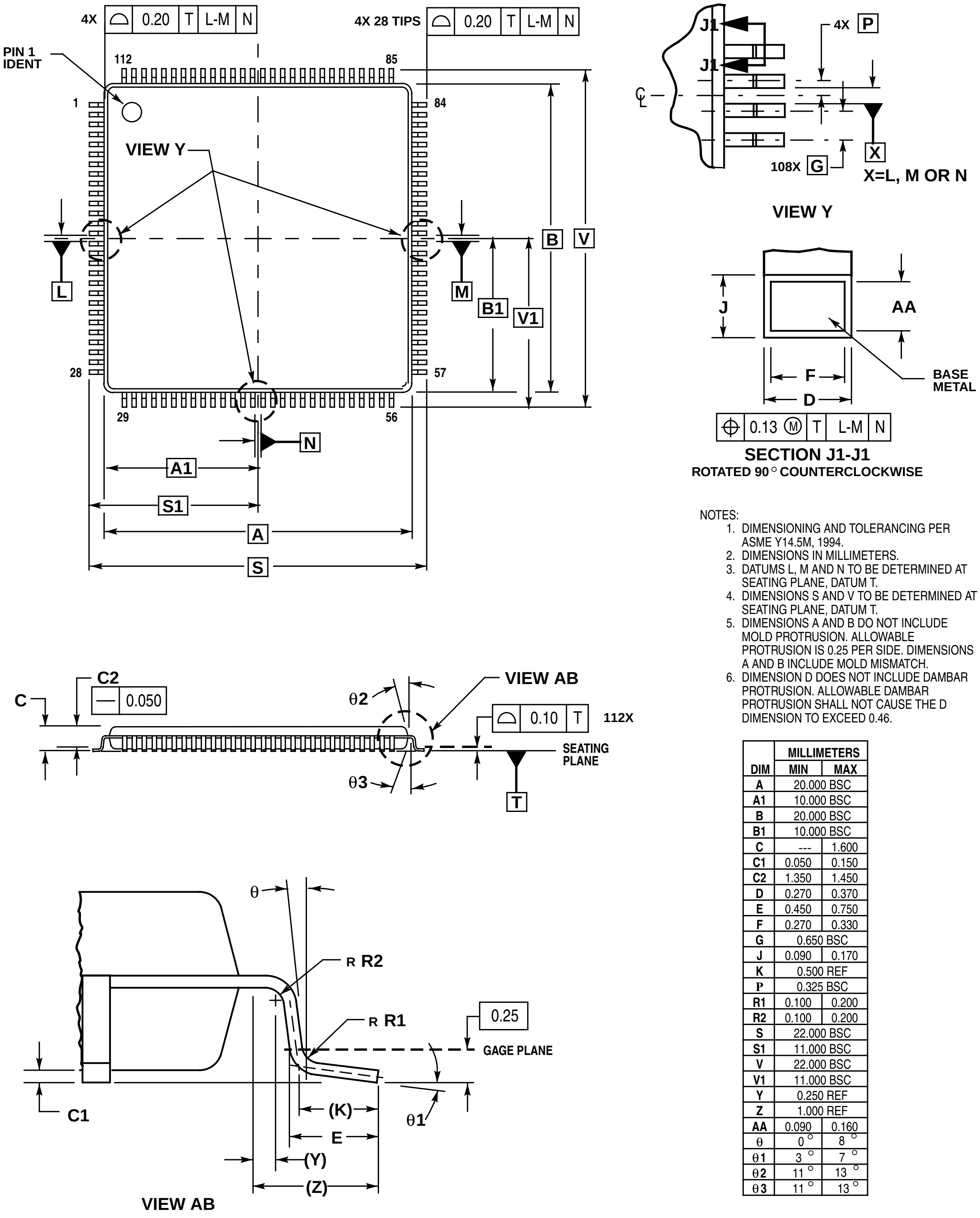


NOTES:
1. DIMENSIONS AND TOLERANCING PER ASME Y14.5M, 1994.
2. DIMENSIONS IN MILLIMETERS.
3. DATUMS L, M, N TO BE DETERMINED AT THE SEATING PLANE, DATUM T.
4. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE, DATUM T.
5. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 PER SIDE. DIMENSIONS A AND B DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE H.
6. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE D DIMENSION TO EXCEED 0.35.

	MILLIMETERS	
DIM	MIN	MAX
A	20.00 BSC	
A1	10.00 BSC	
B	20.00 BSC	
B1	10.00 BSC	
C	1.40	1.60
C1	0.05	0.15
C2	1.35	1.45
D	0.17	0.27
E	0.45	0.75
F	0.17	0.23
G	0.50 BSC	
J	0.09	0.20
K	0.50 REF	
P	0.25 BSC	
R1	0.13	0.20
R2	0.13	0.20
S	22.00 BSC	
S1	11.00 BSC	
V	22.00 BSC	
V1	11.00 BSC	
Y	0.25 REF	
Z	1.00 REF	
AA	0.09	0.16
θ	0°	
θ1	0°	7°
θ2	11°	13°

Figure B-1. 144-Pin LQFP Mechanical Dimensions (Case No. 918-03)

B.2 112-Pin LQFP Package



NOTES:
1. DIMENSIONING AND TOLERANCING PER ASME Y14.5M, 1994.
2. DIMENSIONS IN MILLIMETERS.
3. DATUMS L, M AND N TO BE DETERMINED AT SEATING PLANE, DATUM T.
4. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE, DATUM T.
5. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 PER SIDE. DIMENSIONS A AND B INCLUDE MOLD MISMATCH.
6. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE D DIMENSION TO EXCEED 0.46.

	MILLIMETERS	
DIM	MIN	MAX
A	20.000 BSC	
A1	10.000 BSC	
B	20.000 BSC	
B1	10.000 BSC	
C	---	1.600
C1	0.050	0.150
C2	1.350	1.450
D	0.270	0.370
E	0.450	0.750
F	0.270	0.330
G	0.650 BSC	
J	0.090	0.170
K	0.500 REF	
P	0.325 BSC	
R1	0.100	0.200
R2	0.100	0.200
S	22.000 BSC	
S1	11.000 BSC	
V	22.000 BSC	
V1	11.000 BSC	
Y	0.250 REF	
Z	1.000 REF	
AA	0.090	0.160
θ	0°	8°
θ1	3°	7°
θ2	11°	13°
θ3	11°	13°

Figure B-2. 112-Pin LQFP Mechanical Dimensions (Case No. 987)

Appendix C
PCB Layout Guidelines

The PCB must be carefully laid out to ensure proper operation of the voltage regulator as well as of the MCU itself. The following rules must be observed:

- Every supply pair must be decoupled by a ceramic/tantalum capacitor connected as near as possible to the corresponding pins (C1–C9).
- Central point of the ground star should be the V_{SS1} pin.
- Use low ohmic low inductance connections between V_{SS1}, V_{SS2}, V_{SSA}, $V_{SSX1,2}$ and $V_{SSM1,2,3}$.
- V_{SSPLL} must be directly connected to V_{SS1}.
- Keep traces of V_{SSPLL}, EXTAL and XTAL as short as possible and occupied board area for C10, C11, C14 and Q1 as small as possible.
- Do not place other signals or supplies underneath area occupied by C10, C11, C14 and Q1 and the connection area to the MCU.
- Central power input should be fed in at the V_{DDA}/V_{SSA} pins.

Table C-1. Recommended Components

Component	Purpose	Type	Value
C1	V_{DD1} filter cap	ceramic X7R	>=400 nF
C2	V_{DDA} filter cap	X7R/tantalum	>=100 nF
C3	V_{DDX2} filter cap	X7R/tantalum	>=100 nF
C4	V_{DDR} filter cap	X7R/tantalum	>=100 nF
C5	V_{DDM3} filter cap	X7R/tantalum	>=100 nF
C6	V_{DDM2} filter cap	X7R/tantalum	>=100 nF
C7	V_{DDM1} filter cap	X7R/tantalum	>=100 nF
C8	V_{DDX1} filter cap	X7R/tantalum	>=100 nF
C9	V_{DDPLL} filter cap	ceramic X7R	100 nF .. 220 nF
C10	OSC load cap	See CRG block description chapter	
C11	OSC load cap		
C12	PLL loop filter cap		
C13	PLL loop filter cap		
R1	PLL loop filter res		
Q1	Quartz/Resonator		

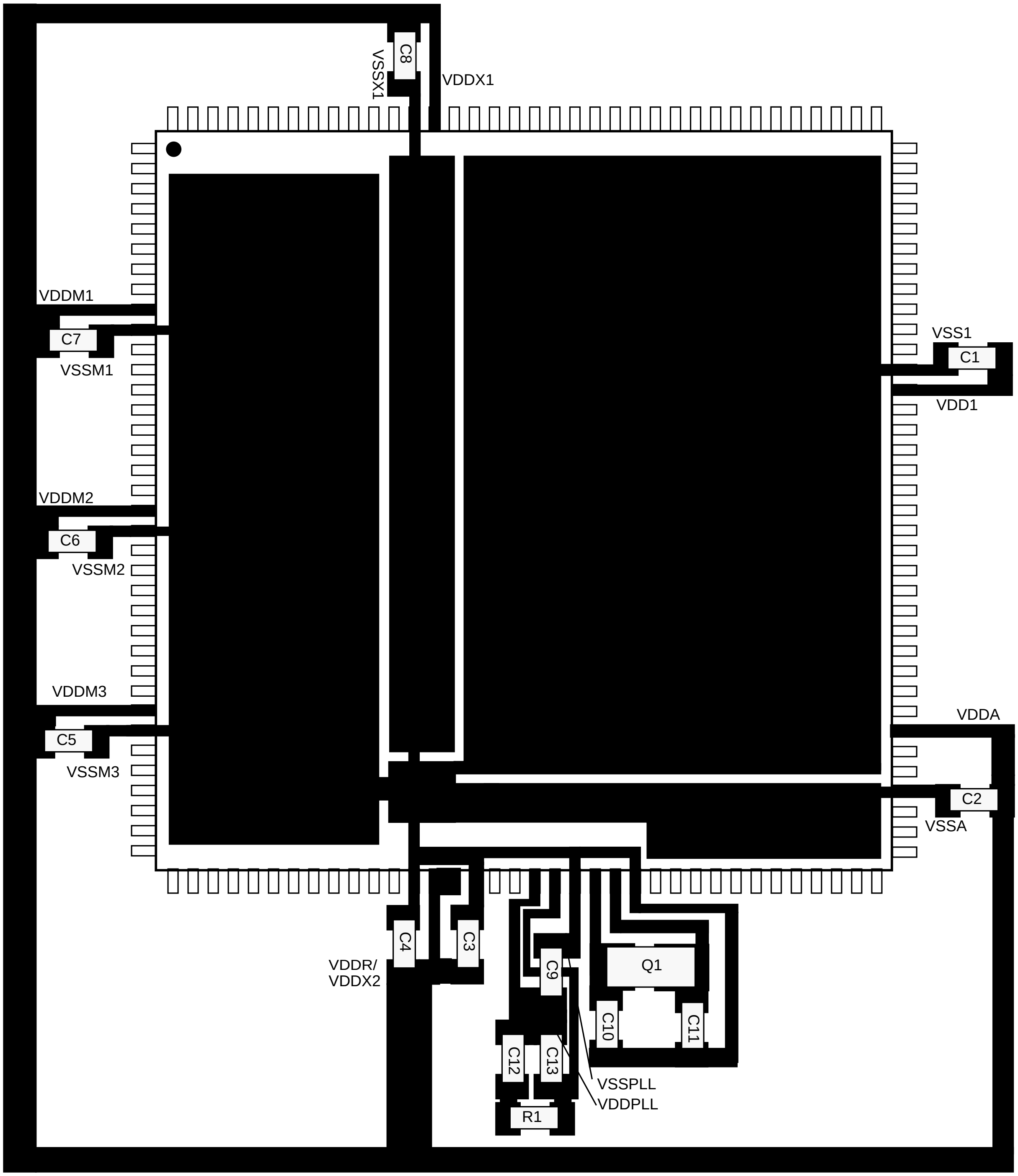


Figure C-1. LQFP144 Recommended PCB Layout

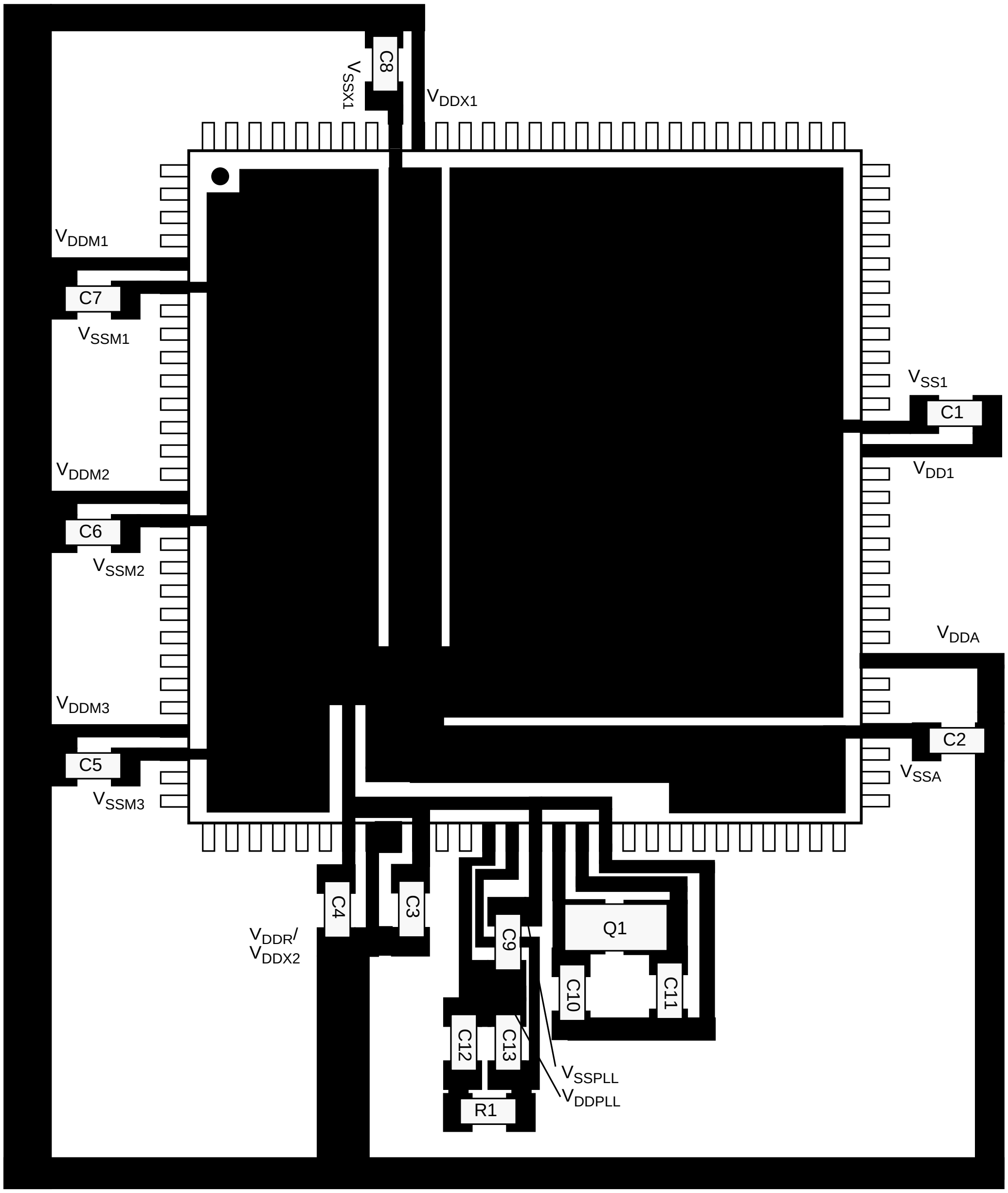


Figure C-2. LQFP112 Recommended PCB Layout

Appendix D
Ordering Information

The following figure provides an ordering number example for the MC9S12XHZ512.

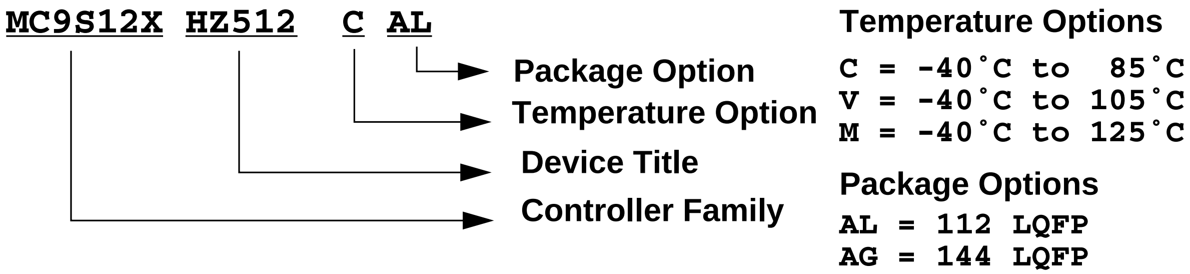


Figure D-1. Order Part Number Example

Customers who place orders using the generic MC partnumbers which are constructed using the above rules will automatically receive our preferred maskset (ie preferred revision of silicon). If the product is updated in the future and a newer maskset is put into production, then the newer maskset may automatically ship against these generic MC partnumbers.

If required, a customer can specify a particular maskset when ordering product. To do this, the customer must order the corresponding "SC" partnumber from the below table. Orders placed against these SC partnumbers will only ever receive one specific maskset. If a new maskset is made available, customers will be notified by PCN (Process Change Notification) but will have to order against a different SC part number in order to receive the new maskset. The marking on the device will be as per the left hand column in the below table independently of whether the MC or the SC partnumber is ordered.

Appendix E
Detailed Register Map

The following tables show the detailed register map of the MC9S12XHZ512.

0x0000–0x0009 Port Integration Module (PIM) Map 1 of 5

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0000	PORTA	R	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA 0
		W								
0x0001	PORTB	R	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
		W								
0x0002	DDRA	R	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		W								
0x0003	DDRB	R	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		W								
0x0004	PORTC	R	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
		W								
0x0005	PORTD	R	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
		W								
0x0006	DDRC	R	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		W								
0x0007	DDRD	R	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		W								
0x0008	PORTE	R	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
		W								
0x0009	DDRE	R	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	0	0
		W								

0x000A–0x000B Module Mapping Control (S12XMMC) Map 1 of 4

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x000A	MMCCTL0	R	0	0	0	0	0	CS2E	CS1E	CS0E
		W								
0x000B	MODE	R	MODC	MODB	MODA	0	0	0	0	0
		W								

0x000C–0x000D Port Integration Module (PIM) Map 2 of 5

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x000C	PUCR	R	PUPKE	BKPUE	0	PUPEE	PUPDE	PUPCE	PUPBE	PUPAE
		W								
0x000D	RDRIV	R	RDPK	0	0	RDPE	RDPD	RDPC	RDPB	RDPA
		W								

0x000E–0x000F External Bus Interface (S12XEBI) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x000E	EBICTL0	R	ITHRS	0	HDBE	ASIZ4	ASIZ3	ASIZ2	ASIZ1	ASIZ0
		W								
0x000F	EBICTL1	R	EWAITE	0	0	0	0	EXSTR2	EXSTR1	EXSTR0
		W								

0x0010–0x0017 Module Mapping Control (S12XMMC) Map 2 of 4

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0010	GPAGE	R	0	GP6	GP5	GP4	GP3	GP2	GP1	GP0
		W								
0x0011	DIRECT	R	DP15	DP14	DP13	DP12	DP11	DP10	DP9	DP8
		W								
0x0012	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0013	MMCCTL1	R	0	0	0	0	0	EROMON	ROMHM	ROMON
		W								
0x0014	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0015	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0016	RPAGE	R	RP7	RP6	RP5	RP4	RP3	RP2	RP1	RP0
		W								
0x0017	EPAGE	R	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
		W								

0x0018–0x001B Miscellaneous Peripheral

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0018	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0019	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x001A	PARTIDH	R	1	1	1	0	0	1	0	0
		W								
0x001B	PARTIDL	R	0	0	0	0	0	0	0	0
		W								

0x001C–0x001F Port Integration Module (PIM) Map 3 of 5

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x001C	ECLKCTL	R	NECLK	NCLKX2	0	0	0	0	EDIV1	EDIV0
		W								

0x001C–0x001F Port Integration Module (PIM) Map 3 of 5

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x001D	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x001E	IRQCR	R	IRQE	IRQEN	0	0	0	0	0	0
		W								
0x001F	SRCR	R	SRRK	0	0	SRRE	SRRD	SRRC	SRRB	SRRA
		W								

0x0020–0x0027 Debug Module (S12XDBG) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0020	DBGC1	R	ARM	0	XGSBPE	BDM	DBGBRK		COMRV	
		W		TRIG						
0x0021	DBGSR	R	TBF	EXTF	0	0	0	SSF2	SSF1	SSF0
		W								
0x0022	DBGTCR	R	TSOURCE		TRANGE		TRCMOD		TALIGN	
		W								
0x0023	DBGC2	R	0	0	0	0	CDCM		ABCM	
		W								
0x0024	DBGTBH	R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		W								
0x0025	DBGTBL	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		W								
0x0026	DBGCNT	R	0	CNT						
		W								
0x0027	DBGSCRX	R	0	0	0	0	SC3	SC2	SC1	SC0
		W								
0x0028[1]	DBGXCTL (COMPA/C)	R	0	NDB	TAG	BRK	RW	RWE	SRC	COMPE
		W								
0x0028[2]	DBGXCTL (COMPB/D)	R	SZE	SZ	TAG	BRK	RW	RWE	SRC	COMPE
		W								
0x0029	DBGXAH	R	0	Bit 22	21	20	19	18	17	Bit 16
		W								
0x002A	DBGXAM	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x002B	DBGXAL	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x002C	DBGXDH	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x002D	DBGXDL	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x002E	DBGXDHM	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x002F	DBGXDLM	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								

[1] This represents the contents if the Comparator A or C control register is blended into this address

[2] This represents the contents if the Comparator B or D control register is blended into this address

0x0030–0x0031 Module Mapping Control (S12XMMC) Map 3 of 4

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0030	PPAGE	R	PIX7	PIX6	PIX5	PIX4	PIX3	PIX2	PIX1	PIX0
		W								
0x0031	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x0032–0x0033 Port Integration Module (PIM) Map 4 of 5

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0032	PORTK	R	PK7	PK6	PK5	PK4	PK3	PK2	PK1	PK0
		W								
0x0033	DDRK	R	DDRK7	DDRK6	DDRK5	DDRK4	DDRK3	DDRK2	DDRK1	DDRK0
		W								

0x0034–0x003F Clock and Reset Generator (CRG) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0034	SYNR	R	0	0	SYN5	SYN4	SYN3	SYN2	SYN1	SYN0
		W								
0x0035	REFDV	R	0	0	REFDV5	REFDV4	REFDV3	REFDV2	REFDV1	REFDV0
		W								
0x0036	CTFLG	R	0	0	0	0	0	0	0	0
		W	Reserved For Factory Test							
0x0037	CRGFLG	R	RTIF	PORF	LVRF	LOCKIF	LOCK	TRACK	SCMIF	SCM
		W								
0x0038	CRGINT	R	RTIE	ILAF	0	LOCKIE	0	0	SCMIE	0
		W								
0x0039	CLKSEL	R	PLLSEL	PSTP	0	0	PLLWAI	0	RTIWAI	COPWAI
		W								
0x003A	PLLCTL	R	CME	PLLON	AUTO	ACQ	FSTWKP	PRE	PCE	SCME
		W								
0x003B	RTICTL	R	RTDEC	RTR6	RTR5	RTR4	RTR3	RTR2	RTR1	RTR0
		W								
0x003C	COPCTL	R	WCOP	RSBCK	0	0	0	CR2	CR1	CR0
		W								
0x003D	FORBYP	R	0	0	0	0	0	0	0	0
		W	Reserved For Factory Test							
0x003E	CTCTL	R	0	0	0	0		0	0	0
		W	Reserved For Factory Test							
0x003F	ARMCOP	R	0	0	0	0	0	0	0	0
		W	Bit 7	6	5	4	3	2	1	Bit 0

0x0040–0x007F Enhanced Capture Timer 16-Bit 8-Channels (ECT) Map (Sheet 1 of 3)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0040	TIOS	R	IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0
		W								
0x0041	CFORC	R	0	0	0	0	0	0	0	0
		W	FOC7	FOC6	FOC5	FOC4	FOC3	FOC2	FOC1	FOC0
0x0042	OC7M	R	OC7M7	OC7M6	OC7M5	OC7M4	OC7M3	OC7M2	OC7M1	OC7M0
		W								
0x0043	OC7D	R	OC7D7	OC7D6	OC7D5	OC7D4	OC7D3	OC7D2	OC7D1	OC7D0
		W								
0x0044	TCNT (hi)	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x0045	TCNT (lo)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0046	TSCR1	R	TEN	TSWAI	TSFRZ	TFFCA	PRNT	0	0	0
		W								
0x0047	TTOV	R	TOV7	TOV6	TOV5	TOV4	TOV3	TOV2	TOV1	TOV0
		W								
0x0048	TCTL1	R	OM7	OL7	OM6	OL6	OM5	OL5	OM4	OL4
		W								
0x0049	TCTL2	R	OM3	OL3	OM2	OL2	OM1	OL1	OM0	OL0
		W								
0x004A	TCTL3	R	EDG7B	EDG7A	EDG6B	EDG6A	EDG5B	EDG5A	EDG4B	EDG4A
		W								
0x004B	TCTL4	R	EDG3B	EDG3A	EDG2B	EDG2A	EDG1B	EDG1A	EDG0B	EDG0A
		W								
0x004C	TIE	R	C7I	C6I	C5I	C4I	C3I	C2I	C1I	C0I
		W								
0x004D	TSCR2	R	TOI	0	0	0	TCRE	PR2	PR1	PR0
		W								
0x004E	TFLG1	R	C7F	C6F	C5F	C4F	C3F	C2F	C1F	C0F
		W								
0x004F	TFLG2	R	TOF	0	0	0	0	0	0	0
		W								
0x0050	TC0 (hi)	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x0051	TC0 (lo)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0052	TC1 (hi)	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x0053	TC1 (lo)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0054	TC2 (hi)	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x0055	TC2 (lo)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								

0x0040–0x007F Enhanced Capture Timer 16-Bit 8-Channels (ECT) Map (Sheet 2 of 3)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0056	TC3 (hi)	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x0057	TC3 (lo)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0058	TC4 (hi)	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x0059	TC4 (lo)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x005A	TC5 (hi)	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x005B	TC5 (lo)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x005C	TC6 (hi)	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x005D	TC6 (lo)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x005E	TC7 (hi)	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x005F	TC7 (lo)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0060	PACTL	R	0	PAEN	PAMOD	PEDGE	CLK1	CLK0	PAOVI	PAI
		W								
0x0061	PAFLG	R	0	0	0	0	0	0	PAOVF	PAIF
		W								
0x0062	PACN3 (hi)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0063	PACN2 (lo)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0064	PACN1 (hi)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0065	PACN0 (lo)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0066	MCCTL	R	MCZI	MODMC	RDMCL	0	0	MCEN	MCPR1	MCPR0
		W				ICLAT	FLMC			
0x0067	MCFLG	R	MCZF	0	0	0	POLF3	POLF2	POLF1	POLF0
		W								
0x0068	ICPAR	R	0	0	0	0	PA3EN	PA2EN	PA1EN	PA0EN
		W								
0x0069	DLYCT	R	DLY7	DLY6	DLY5	DLY4	DLY3	DLY2	DLY1	DLY0
		W								
0x006A	ICOVW	R	NOVW7	NOVW6	NOVW5	NOVW4	NOVW3	NOVW2	NOVW1	NOVW0
		W								
0x006B	ICSYS	R	SH37	SH26	SH15	SH04	TFMOD	PACMX	BUFEN	LATQ
		W								
0x006C	OCPD	R	OCPD7	OCPD6	OCPD5	OCPD4	OCPD3	OCPD2	OCPD1	OCPD0
		W								

0x0040–0x007F Enhanced Capture Timer 16-Bit 8-Channels (ECT) Map (Sheet 3 of 3)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x006D	TIMTST	R	0	0	0	0	0	0	0	0
		W	Reserved For Factory Test							
0x006E	PTPSR	R	PTPS7	PTPS6	PTPS5	PTPS4	PTPS3	PTPS2	PTPS1	PTPS0
		W								
0x006F	PTMCPSR	R	PTMPS7	PTMPS6	PTMPS5	PTMPS4	PTMPS3	PTMPS2	PTMPS1	PTMPS0
		W								
0x0070	PBCTL	R	0	PBEN	0	0	0	0	PBOVI	0
		W								
0x0071	PBFLG	R	0	0	0	0	0	0	PBOVF	0
		W								
0x0072	PA3H	R	PA3H7	PA3H6	PA3H5	PA3H4	PA3H3	PA3H2	PA3H1	PA3H0
		W								
0x0073	PA2H	R	PA2H7	PA2H6	PA2H5	PA2H4	PA2H3	PA2H2	PA2H1	PA2H0
		W								
0x0074	PA1H	R	PA1H7	PA1H6	PA1H5	PA1H4	PA1H3	PA1H2	PA1H1	PA1H 0
		W								
0x0075	PA0H	R	PA0H7	PA0H6	PA0H5	PA0H4	PA0H3	PA0H2	PA0H1	PA0H0
		W								
0x0076	MCCNT (hi)	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x0077	MCCNT (lo)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0078	TC0H (hi)	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x0079	TC0H (lo)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x007A	TC1H (hi)	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x007B	TC1H (lo)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x007C	TC2H (hi)	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x007D	TC2H (lo)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x007E	TC3H (hi)	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x007F	TC3H (lo)	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								

0x0080–0x00AF Analog-to-Digital Converter 10-bit 16-Channels (ATD) Map (Sheet 1 of 3)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0080	ATDCTL0	R	0	0	0	0	WRAP3	WRAP2	WRAP1	WRAP0
		W								
0x0081	ATDCTL1	R	ETRIG SEL	0	0	0	ETRIG CH3	ETRIG CH2	ETRIG CH1	ETRIG CH0
		W								
0x0082	ATDCTL2	R	ADPU	AFFC	AWAI	ETRIGLE	ETRIGP	ETRIGE	ASCIE	ASCIF
		W								
0x0083	ATDCTL3	R	0	S8C	S4C	S2C	S1C	FIFO	FRZ1	FRZ0
		W								
0x0084	ATDCTL4	R	SRES8	SMP1	SMP0	PRS4	PRS3	PRS2	PRS1	PRS0
		W								
0x0085	ATDCTL5	R	DJM	DSGN	SCAN	MULT	CD	CC	CB	CA
		W								
0x0086	ATDSTAT0	R	SCF	0	ETORF	FIFOR	CC3	CC2	CC1	CC0
		W								
0x0087	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0088	ATDTEST0	R	U	U	U	U	U	U	U	U
		W	Reserved For Factory Test							
0x0089	ATDTEST1	R	0	0	0	0	0	0	0	0
		W	Reserved For Factory Test							
0x008A	ATDSTAT2	R	CCF15	CCF14	CCF13	CCF12	CCF11	CCF10	CCF9	CCF8
		W								
0x008B	ATDSTAT1	R	CCF7	CCF6	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0
		W								
0x008C	ATDDIEN0	R	IEN15	IEN14	IEN13	IEN12	IEN11	IEN10	IEN9	IEN8
		W								
0x008D	ATDDIEN1	R	IEN7	IEN6	IEN5	IEN4	IEN3	IEN2	IEN1	IEN0
		W								
0x008E	PORTAD0	R	PTAD15	PTAD14	PTAD13	PTAD12	PTAD11	PTAD10	PTAD9	PTAD8
		W								
0x008F	PORTAD1	R	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
		W								
0x0090	ATDDR0H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x0091	ATDDR0L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x0092	ATDDR1H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x0093	ATDDR1L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x0094	ATDDR2H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x0095	ATDDR2L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x0096	ATDDR3H	R	Bit15	14	13	12	11	10	9	Bit8
		W								

0x0080–0x00AF Analog-to-Digital Converter 10-bit 16-Channels (ATD) Map (Sheet 2 of 3)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0097	ATDDR3L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x0098	ATDDR4H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x0099	ATDDR4L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x009A	ATDDR5H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x009B	ATDDR5L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x009C	ATDDR6H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x009D	ATDDR6L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x009E	ATDDR7H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x009F	ATDDR7L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x00A0	ATDDR8H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x00A1	ATDDR8L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x00A2	ATDDR9H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x00A3	ATDDR9L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x00A4	ATDDR10H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x00A5	ATDDR10L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x00A6	ATDDR11H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x00A7	ATDDR11L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x00A8	ATDDR12H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x00A9	ATDDR12L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x00AA	ATDDR13H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x00AB	ATDDR13L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x00AC	ATDDR14H	R	Bit15	14	13	12	11	10	9	Bit8
		W								

0x0080–0x00AF Analog-to-Digital Converter 10-bit 16-Channels (ATD) Map (Sheet 3 of 3)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00AD	ATDDR14L	R	Bit7	Bit6	0	0	0	0	0	0
		W								
0x00AE	ATDDR15H	R	Bit15	14	13	12	11	10	9	Bit8
		W								
0x00AF	ATDDR15L	R	Bit7	Bit6	0	0	0	0	0	0
		W								

0x00B0–0x00BF Interrupt Module (S12XINT) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00B0	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x00B1	IVBR	R	IVB_ADDR[7:0]							
		W								
0x00B2	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x00B3	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x00B4	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x00B5	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x00B6	INT_XGPRIO	R	0	0	0	0	0	XILVL[2:0]		
		W								
0x00B7	INT_CFADDR	R	INT_CFADDR[7:4]				0	0	0	0
		W								
0x00B8	INT_CFDATA0	R	RQST	0	0	0	0	PRIOLVL[2:0]		
		W								
0x00B8	INT_CFDATA1	R	RQST	0	0	0	0	PRIOLVL[2:0]		
		W								
0x00B9	INT_CFDATA2	R	RQST	0	0	0	0	PRIOLVL[2:0]		
		W								
0x00BA	INT_CFDATA3	R	RQST	0	0	0	0	PRIOLVL[2:0]		
		W								
0x00BB	INT_CFDATA4	R	RQST	0	0	0	0	PRIOLVL[2:0]		
		W								
0x00BC	INT_CFDATA5	R	RQST	0	0	0	0	PRIOLVL[2:0]		
		W								
0x00BE	INT_CFDATA6	R	RQST	0	0	0	0	PRIOLVL[2:0]		
		W								
0x00BF	INT_CFDATA7	R	RQST	0	0	0	0	PRIOLVL[2:0]		
		W								

0x00C0–0x00C7 Inter IC Bus (IIC0) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00C0	IB0AD	R	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	0
		W								
0x00C1	IB0FD	R	IBC7	IBC6	IBC5	IBC4	IBC3	IBC2	IBC1	IBC0
		W								
0x00C2	IB0CR	R	IBEN	IBIE	MS/$\overline{\text{SL}}$	TX/$\overline{\text{RX}}$	TXAK	0	0	IBSWAI
		W						RSTA		
0x00C3	IB0SR	R	TCF	IAAS	IBB	IBAL	0	SRW	IBIF	RXAK
		W								
0x00C4	IB0DR	R	D7	D6	D5	D4	D3	D2	D1	D 0
		W								
0x00C5	IB0CR2	R	GCEN	ADTYPE	0	0	0	ADR10	ADR9	ADR8
		W								
0x00C6	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x00C7	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x00C8–0x00CF Asynchronous Serial Interface (SCI0) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00C8	SCI0BDH[1]	R	IREN	TNP1	TNP0	SBR12	SBR11	SBR10	SBR9	SBR8
		W								
0x00C9	SCI0BDL[1]	R	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
		W								
0x00CA	SCI0CR1[1]	R	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
		W								
0x00C8	SCI0ASR1[2]	R	RXEDGIF	0	0	0	0	BERRV	BERRIF	BKDIF
		W								
0x00C9	SCI0ACR1[2]	R	RXEDGIE	0	0	0	0	0	BERRIE	BKDIE
		W								
0x00CA	SCI0ACR2[2]	R	0	0	0	0	0	BERRM1	BERRM0	BKDFE
		W								
0x00CB	SCI0CR2	R	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
		W								
0x00CC	SCI0SR1	R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
		W								
0x00CD	SCI0SR2	R	AMAP	0	0	TXPOL	RXPOL	BRK13	TXDIR	RAF
		W								
0x00CE	SCI0DRH	R	R8	T8	0	0	0	0	0	0
		W								
0x00CF	SCI0DRL	R	R7	R6	R5	R4	R3	R2	R1	R0
		W	T7	T6	T5	T4	T3	T2	T1	T0

[1] Those registers are accessible if the AMAP bit in the SCI0SR2 register is set to zero

[2] Those registers are accessible if the AMAP bit in the SCI0SR2 register is set to one

0x00D0–0x00D7 Asynchronous Serial Interface (SCI1) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00D0	SCI1BDH[1]	R	IREN	TNP1	TNP0	SBR12	SBR11	SBR10	SBR9	SBR8
		W								
0x00D1	SCI1BDL[1]	R	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
		W								
0x00D2	SCI1CR1[1]	R	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
		W								
0x00D0	SCI1ASR1[2]	R	RXEDGIF	0	0	0	0	BERRV	BERRIF	BKDIF
		W								
0x00D1	SCI1ACR1[2]	R	RXEDGIE	0	0	0	0	0	BERRIE	BKDIE
		W								
0x00D2	SCI1ACR2[2]	R	0	0	0	0	0	BERRM1	BERRM0	BKDFE
		W								
0x00D3	SCI1CR2	R	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
		W								
0x00D4	SCI1SR1	R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
		W								
0x00D5	SCI1SR2	R	AMAP	0	0	TXPOL	RXPOL	BRK13	TXDIR	RAF
		W								
0x00D6	SCI1DRH	R	R8	T8	0	0	0	0	0	0
		W								
0x00D7	SCI1DRL	R	R7	R6	R5	R4	R3	R2	R1	R0
		W	T7	T6	T5	T4	T3	T2	T1	T0

[1] Those registers are accessible if the AMAP bit in the SCI1SR2 register is set to zero

[2] Those registers are accessible if the AMAP bit in the SCI1SR2 register is set to one

0x00D8–0x00DF Serial Peripheral Interface (SPI) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00D8	SPICR1	R	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
		W								
0x00D9	SPICR2	R	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
		W								
0x00DA	SPIBR	R	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
		W								
0x00DB	SPISR	R	SPIF	0	SPTEF	MODF	0	0	0	0
		W								
0x00DC	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x00DD	SPIDR	R	Bit7	6	5	4	3	2	1	Bit0
		W								
0x00DE	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x00DF	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x00E0–0x00FF Reserved

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00E0– 0x00FF	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x0100–0x010F Flash Control Register (FTX512K4) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0100	FCLKDIV	R	FDIVLD	PRDIV8	FDIV5	FDIV4	FDIV3	FDIV2	FDIV1	FDIV0
		W								
0x0101	FSEC	R	KEYEN1	KEYEN0	RNV5	RNV4	RNV3	RNV2	SEC1	SEC0
		W								
0x0102	FTSTMOD	R	0	MRDS		WRALL	0	0	0	0
		W								
0x0103	FCNFG	R	CBEIE	CCIE	KEYACC	0	0	0	0	0
		W								
0x0104	FPROT	R	FPOPEN	RNV6	FPHDIS	FPHS1	FPHS0	FPLDIS	FPLS1	FPLS0
		W								
0x0105	FSTAT	R	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	0	0
		W								
0x0106	FCMD	R	0	CMDB[6:0]						
		W								
0x0107	FCTL	R	NV7	NV6	NV5	NV4	NV3	NV2	NV1	NV0
		W								
0x0108	FADDRHI	R	FADDRHI							
		W								
0x0109	FADDRLO	R	FADDRLO							
		W								
0x010A	FDATAHI	R	FDATAHI							
		W								
0x010B	FDATALO	R	FDATALO							
		W								
0x010C	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x010D	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x010E	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x010F	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x0110–0x011B EEPROM Control Register (EETX4K) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0110	ECLKDIV	R	EDIVLD	PRDIV8	EDIV5	EDIV4	EDIV3	EDIV2	EDIV1	EDIV0
		W								
0x0111	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0112	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0113	ECNFG	R	CBEIE	CCIE	0	0	0	0	0	0
		W								
0x0114	EPROT	R	EPOPEN	RNV6	RNV5	RNV4	EPDIS	EPS2	EPS1	EPS0
		W								
0x0115	ESTAT	R	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	0	0
		W								
0x0116	ECMD	R	0	CMDB[6:0]						
		W								
0x0117	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0118	EADDRHI	R	0	0	0	0	0	EABHI		
		W								
0x0119	EADDRLO	R	EABLO							
		W								
0x011A	EDATAHI	R	EDHI							
		W								
0x011B	EDATALO	R	EDLO							
		W								

0x011C–0x011F Memory Map Control (S12XMMC) Map 4 of 4

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x011C	RAMWPC	R	RPWE	0	0	0	0	0	AVIE	AVIF
		W								
0x011D	RAMXGU	R	1	XGU6	XGU5	XGU4	XGU3	XGU2	XGU1	XGU0
		W								
0x011E	RAMSHL	R	1	SHL6	SHL5	SHL4	SHL3	SHL2	SHL1	SHL0
		W								
0x011F	RAMSHU	R	1	SHU6	SHU5	SHU4	SHU3	SHU2	SHU1	SHU0
		W								

0x0120–0x0137 Liquid Crystal Display 32x4 (LCD) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0120	LCDCR0	R	LCDEN	0	LCLK2	LCLK1	LCLK0	BIAS	DUTY1	DUTY0
		W								
0x0121	LCDCR1	R	0	0	0	0	0	0	LCDSWAI	LCDRPSTP
		W								
0x0122	FPENR0	R	FP7EN	FP6EN	FP5EN	FP4EN	FP3EN	FP2EN	FP1EN	FP0EN
		W								
0x0123	FPENR1	R	FP15EN	FP14EN	FP13EN	FP12EN	FP11EN	FP10EN	FP9EN	FP8EN
		W								
0x0124	FPENR2	R	FP23EN	FP22EN	FP21EN	FP20EN	FP19EN	FP18EN	FP17EN	FP16EN
		W								
0x0125	FPENR3	R	FP31EN	FP30EN	FP29EN	FP28EN	FP27EN	FP26EN	FP25EN	FP24EN
		W								
0x0126	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0127	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0128	LCDRAM0	R	FP1BP3	FP1BP2	FP1BP1	FP1BP0	FP0BP3	FP0BP2	FP0BP1	FP0BP0
		W								
0x0129	LCDRAM1	R	FP3BP3	FP3BP2	FP3BP1	FP3BP0	FP2BP3	FP2BP2	FP2BP1	FP2BP0
		W								
0x012A	LCDRAM2	R	FP5BP3	FP5BP2	FP5BP1	FP5BP0	FP4BP3	FP4BP2	FP4BP1	FP4BP0
		W								
0x012B	LCDRAM3	R	FP7BP3	FP7BP2	FP7BP1	FP7BP0	FP6BP3	FP6BP2	FP6BP1	FP6BP0
		W								
0x012C	LCDRAM4	R	FP9BP3	FP9BP2	FP9BP1	FP9BP0	FP8BP3	FP8BP2	FP8BP1	FP8BP0
		W								
0x012D	LCDRAM5	R	FP11BP3	FP11BP2	FP11BP1	FP11BP0	FP10BP3	FP10BP2	FP10BP1	FP10BP0
		W								
0x012E	LCDRAM6	R	FP13BP3	FP13BP2	FP13BP1	FP13BP0	FP12BP3	FP12BP2	FP12BP1	FP12BP0
		W								
0x012F	LCDRAM7	R	FP15BP3	FP15BP2	FP15BP1	FP15BP0	FP14BP3	FP14BP2	FP14BP1	FP14BP0
		W								
0x0130	LCDRAM8	R	FP17BP3	FP17BP2	FP17BP1	FP17BP0	FP16BP3	FP16BP2	FP16BP1	FP16BP0
		W								
0x0131	LCDRAM9	R	FP19BP3	FP19BP2	FP19BP1	FP19BP0	FP18BP3	FP18BP2	FP18BP1	FP18BP0
		W								
0x0132	LCDRAM10	R	FP21BP3	FP21BP2	FP21BP1	FP21BP0	FP20BP3	FP20BP2	FP20BP1	FP20BP0
		W								
0x0133	LCDRAM11	R	FP23BP3	FP23BP2	FP23BP1	FP23BP0	FP22BP3	FP22BP2	FP22BP1	FP22BP0
		W								
0x0134	LCDRAM12	R	FP25BP3	FP25BP2	FP25BP1	FP25BP0	FP24BP3	FP24BP2	FP24BP1	FP24BP0
		W								

0x0120–0x0137 Liquid Crystal Display 32x4 (LCD) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0135	LCDRAM13	R	FP27BP3	FP27BP2	FP27BP1	FP27BP0	FP26BP3	FP26BP2	FP26BP1	FP26BP0
		W								
0x0136	LCDRAM14	R	FP29BP3	FP29BP2	FP29BP1	FP29BP0	FP28BP3	FP28BP2	FP28BP1	FP28BP0
		W								
0x0137	LCDRAM15	R	FP31BP3	FP31BP2	FP31BP1	FP31BP0	FP30BP3	FP30BP2	FP30BP1	FP30BP0
		W								

0x0138–0x013F Inter IC Bus (IIC1) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0138	IB1AD	R	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	0
		W								
0x0139	IB1FD	R	IBC7	IBC6	IBC5	IBC4	IBC3	IBC2	IBC1	IBC0
		W								
0x013A	IB1CR	R	IBEN	IBIE	MS/$\overline{\text{SL}}$	TX/$\overline{\text{RX}}$	TXAK	0	0	IBSWAI
		W						RSTA		
0x013B	IB1SR	R	TCF	IAAS	IBB	IBAL	0	SRW	IBIF	RXAK
		W								
0x013C	IB1DR	R	D7	D6	D5	D4	D3	D2	D1	D 0
		W								
0x013D	IB1CR2	R	GCEN	ADTYPE	0	0	0	ADR10	ADR9	ADR8
		W								
0x013E	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x013F	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x0140–0x017F Freescale Scalable CAN — MSCAN (CAN0) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0140	CAN0CTL0	R	RXFRM	RXACT	CSWAI	SYNCH	TIME	WUPE	SLPRQ	INITRQ
		W								
0x0141	CAN0CTL1	R	CANE	CLKSRC	LOOPB	LISTEN	BORM	WUPM	SLPAK	INITAK
		W								
0x0142	CAN0BTR0	R	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
		W								
0x0143	CAN0BTR1	R	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
		W								
0x0144	CAN0RFLG	R	WUPIF	CSCIF	RSTAT1	RSTAT0	TSTAT1	TSTAT0	OVRIF	RXF
		W								
0x0145	CAN0RIER	R	WUPIE	CSCIE	RSTATE1	RSTATE0	TSTATE1	TSTATE0	OVRIE	RXFIE
		W								
0x0146	CAN0TFLG	R	0	0	0	0	0	TXE2	TXE1	TXE0
		W								
0x0147	CAN0TIER	R	0	0	0	0	0	TXEIE2	TXEIE1	TXEIE0
		W								

0x0140–0x017F Freescale Scalable CAN — MSCAN (CAN0) Map (continued)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0148	CAN0TARQ	R	0	0	0	0	0	ABTRQ2	ABTRQ1	ABTRQ0
		W								
0x0149	CAN0TAAK	R	0	0	0	0	0	ABTAK2	ABTAK1	ABTAK0
		W								
0x014A	CAN0TBSEL	R	0	0	0	0	0	TX2	TX1	TX0
		W								
0x014B	CAN0IDAC	R	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0
		W								
0x014C	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x014D	CAN0MISC	R	0	0	0	0	0	0	0	BOHOLD
		W								
0x014E	CAN0RXERR	R	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
		W								
0x014F	CAN0TXERR	R	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
		W								
0x0150–0x0153	CAN0IDAR0–CAN0IDAR3	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W								
0x0154–0x0157	CAN0IDMR0–CAN0IDMR3	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W								
0x0158–0x015B	CAN0IDAR4–CAN0IDAR7	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W								
0x015C – 0x015F	CAN0IDMR4–CAN0IDMR7	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W								
0x0160–0x016F	CAN0RXFG	R	FOREGROUND RECEIVE BUFFER (See Detailed MSCAN Foreground Receive and Transmit Buffer Layout)							
		W								
0x0170–0x017F	CAN0TXFG	R	FOREGROUND TRANSMIT BUFFER (See Detailed MSCAN Foreground Receive and Transmit Buffer Layout)							
		W								

Detailed MSCAN Foreground Receive and Transmit Buffer Layout

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xXXX0	Extended ID	R	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
	Standard ID	R	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
	CANxRIDR0	W								
0xXXX1	Extended ID	R	ID20	ID19	ID18	SRR=1	IDE=1	ID17	ID16	ID15
	Standard ID	R	ID2	ID1	ID0	RTR	IDE=0			
	CANxRIDR1	W								
0xXXX2	Extended ID	R	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
	Standard ID	R								
	CANxRIDR2	W								
0xXXX3	Extended ID	R	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
	Standard ID	R								
	CANxRIDR3	W								

Detailed MSCAN Foreground Receive and Transmit Buffer Layout (continued)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0xXXX4 – 0xXXXB	CANxRDSR0– CANxRDSR7	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		W								
0xXXXC	CANRxDLR	R					DLC3	DLC2	DLC1	DLC0
		W								
0xXXXD	Reserved	R								
		W								
0xXXXE	CANxRTSRH	R	TSR15	TSR14	TSR13	TSR12	TSR11	TSR10	TSR9	TSR8
		W								
0xXXXF	CANxRTSRL	R	TSR7	TSR6	TSR5	TSR4	TSR3	TSR2	TSR1	TSR0
		W								
0xXX10	Extended ID CANxTIDR0	R W	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
	Standard ID	R W	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
0xXX0x XX10	Extended ID CANxTIDR1	R W	ID20	ID19	ID18	SRR=1	IDE=1	ID17	ID16	ID15
	Standard ID	R W	ID2	ID1	ID0	RTR	IDE=0			
0xXX12	Extended ID CANxTIDR2	R W	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
	Standard ID	R W								
0xXX13	Extended ID CANxTIDR3	R W	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
	Standard ID	R W								
0xXX14 – 0xXX1B	CANxTDSR0– CANxTDSR7	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0xXX1C	CANxTDLR	R W					DLC3	DLC2	DLC1	DLC0
0xXX1D	CANxTTBPR	R W	PRIO7	PRIO6	PRIO5	PRIO4	PRIO3	PRIO2	PRIO1	PRIO0
0xXX1E	CANxTTSRH	R	TSR15	TSR14	TSR13	TSR12	TSR11	TSR10	TSR9	TSR8
		W								
0xXX1F	CANxTTSRL	R	TSR7	TSR6	TSR5	TSR4	TSR3	TSR2	TSR1	TSR0
		W								

0x0180–0x01BF Freescale Scalable CAN — MSCAN (CAN1) Map (Sheet 1 of 2)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0180	CAN1CTL0	R	RXFRM	RXACT	CSWAI	SYNCH	TIME	WUPE	SLPRQ	INITRQ
		W								
0x0181	CAN1CTL1	R	CANE	CLKSRC	LOOPB	LISTEN	BORM	WUPM	SLPAK	INITAK
		W								
0x0182	CAN1BTR0	R	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
		W								
0x0183	CAN1BTR1	R	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
		W								
0x0184	CAN1RFLG	R	WUPIF	CSCIF	RSTAT1	RSTAT0	TSTAT1	TSTAT0	OVRIF	RXF
		W								
0x0185	CAN1RIER	R	WUPIE	CSCIE	RSTATE1	RSTATE0	TSTATE1	TSTATE0	OVRIE	RXFIE
		W								
0x0186	CAN1TFLG	R	0	0	0	0	0	TXE2	TXE1	TXE0
		W								
0x0187	CAN1TIER	R	0	0	0	0	0	TXEIE2	TXEIE1	TXEIE0
		W								
0x0188	CAN1TARQ	R	0	0	0	0	0	ABTRQ2	ABTRQ1	ABTRQ0
		W								
0x0189	CAN1TAAK	R	0	0	0	0	0	ABTAK2	ABTAK1	ABTAK0
		W								
0x018A	CAN1TBSEL	R	0	0	0	0	0	TX2	TX1	TX0
		W								
0x018B	CAN1IDAC	R	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0
		W								
0x018C	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x018D	CAN1MISC	R	0	0	0	0	0	0	0	BOHOLD
		W								
0x018E	CAN1RXERR	R	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
		W								
0x018F	CAN1TXERR	R	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
		W								
0x0190	CAN1IDAR0	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W								
0x0191	CAN1IDAR1	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W								
0x0192	CAN1IDAR2	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W								
0x0193	CAN1IDAR3	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W								
0x0194	CAN1IDMR0	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W								
0x0195	CAN1IDMR1	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W								

0x0180–0x01BF Freescale Scalable CAN — MSCAN (CAN1) Map (Sheet 2 of 2)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0196	CAN1IDMR2	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W								
0x0197	CAN1IDMR3	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W								
0x0198	CAN1IDAR4	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W								
0x0199	CAN1IDAR5	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W								
0x019A	CAN1IDAR6	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W								
0x019B	CAN1IDAR7	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W								
0x019C	CAN1IDMR4	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W								
0x019D	CAN1IDMR5	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W								
0x019E	CAN1IDMR6	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W								
0x019F	CAN1IDMR7	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W								
0x01A0– 0x01AF	CAN1RXFG	R	FOREGROUND RECEIVE BUFFER (See Detailed MSCAN Foreground Receive and Transmit Buffer Layout)							
		W								
0x01B0– 0x01BF	CAN1TXFG	R	FOREGROUND TRANSMIT BUFFER (See Detailed MSCAN Foreground Receive and Transmit Buffer Layout)							
		W								

0x01C0–0x01FF Motor Controller 10-bit 12-Channels (MC) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x01C0	MCCTL0	R	0	MCPRE1	MCPRE0	MCSWAI	FAST	DITH	0	MCTOIF
		W								
0x01C1	MCCTL1	R	RECIRC	0	0	0	0	0	0	MCTOIE
		W								
0x01C2	MCPER (hi)	R	0	0	0	0	0	P10	P9	P8
		W								
0x01C3	MCPER (lo)	R	P7	P6	P5	P4	P3	P2	P1	P0
		W								
0x01C4– 0x01CF	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x01D0	MCCC0	R	MCOM1	MCOM0	MCAM1	MCAM0	0	0	CD1	CD0
		W								
0x01D1	MCCC1	R	MCOM1	MCOM0	MCAM1	MCAM0	0	0	CD1	CD0
		W								
0x01D2	MCCC2	R	MCOM1	MCOM0	MCAM1	MCAM0	0	0	CD1	CD0
		W								

0x01C0–0x01FF Motor Controller 10-bit 12-Channels (MC) Map (continued)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x01D3	MCCC3	R	MCOM1	MCOM0	MCAM1	MCAM0	0	0	CD1	CD0
		W								
0x01D4	MCCC4	R	MCOM1	MCOM0	MCAM1	MCAM0	0	0	CD1	CD0
		W								
0x01D5	MCCC5	R	MCOM1	MCOM0	MCAM1	MCAM0	0	0	CD1	CD0
		W								
0x01D6	MCCC6	R	MCOM1	MCOM0	MCAM1	MCAM0	0	0	CD1	CD0
		W								
0x01D7	MCCC7	R	MCOM1	MCOM0	MCAM1	MCAM0	0	0	CD1	CD0
		W								
0x01D8	MCCC8	R	MCOM1	MCOM0	MCAM1	MCAM0	0	0	CD1	CD0
		W								
0x01D9	MCCC9	R	MCOM1	MCOM0	MCAM1	MCAM0	0	0	CD1	CD0
		W								
0x01DA	MCCC10	R	MCOM1	MCOM0	MCAM1	MCAM0	0	0	CD1	CD0
		W								
0x01DB	MCCC11	R	MCOM1	MCOM0	MCAM1	MCAM0	0	0	CD1	CD0
		W								
0x01DC–0x01DF	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x01E0	MCDC0 (hi)	R	S	S	S	S	S	D10	D9	D8
		W								
0x01E1	MCDC0 (lo)	R	D7	D6	D5	D4	D3	D2	D1	D0
		W								
0x01E2	MCDC1 (hi)	R	S	S	S	S	S	D10	D9	D8
		W								
0x01E3	MCDC1 (lo)	R	D7	D6	D5	D4	D3	D2	D1	D0
		W								
0x01E4	MCDC2 (hi)	R	S	S	S	S	S	D10	D9	D8
		W								
0x01E5	MCDC2 (lo)	R	D7	D6	D5	D4	D3	D2	D1	D0
		W								
0x01E6	MCDC3 (hi)	R	S	S	S	S	S	D10	D9	D8
		W								
0x01E7	MCDC3 (lo)	R	D7	D6	D5	D4	D3	D2	D1	D0
		W								
0x01E8	MCDC4 (hi)	R	S	S	S	S	S	D10	D9	D8
		W								
0x01E9	MCDC4 (lo)	R	D7	D6	D5	D4	D3	D2	D1	D0
		W								
0x01EA	MCDC5 (hi)	R	S	S	S	S	S	D10	D9	D8
		W								
0x01EB	MCDC5 (lo)	R	D7	D6	D5	D4	D3	D2	D1	D0
		W								
0x01EC	MCDC6 (hi)	R	S	S	S	S	S	D10	D9	D8
		W								

0x01C0–0x01FF Motor Controller 10-bit 12-Channels (MC) Map (continued)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x01ED	MCDC6 (lo)	R W	D7	D6	D5	D4	D3	D2	D1	D0
0x01EE	MCDC7 (hi)	R	S	S	S	S	S	D10	D9	D8
		W								
0x01EF	MCDC7 (lo)	R W	D7	D6	D5	D4	D3	D2	D1	D0
0x01F0	MCDC8 (hi)	R	S	S	S	S	S	D10	D9	D8
		W								
0x01F1	MCDC8 (lo)	R W	D7	D6	D5	D4	D3	D2	D1	D0
0x01F2	MCDC9 (hi)	R	S	S	S	S	S	D10	D9	D8
		W								
0x01F3	MCDC9 (lo)	R W	D7	D6	D5	D4	D3	D2	D1	D0
0x01F4	MCDC10 (hi)	R	S	S	S	S	S	D10	D9	D8
		W								
0x01F5	MCDC10 (lo)	R W	D7	D6	D5	D4	D3	D2	D1	D0
0x01F6	MCDC11 (hi)	R	S	S	S	S	S	D10	D9	D8
		W								
0x01F7	MCDC11 (lo)	R W	D7	D6	D5	D4	D3	D2	D1	D0
0x01F8–0x01FF	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x0200–0x027F Port Integration Module (PIM) Map 5 of 5

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0200	PTT	R W	PTT7	PTT6	PTT5	PTT4	PTT3	PTT2	PTT1	PTT0
0x0201	PTIT	R	PTIT7	PTIT6	PTIT5	PTIT4	PTIT3	PTIT2	PTIT1	PTIT0
		W								
0x0202	DDRT	R W	DDRT7	DDRT6	DDRT5	DDRT4	DDRT3	DDRT2	DDRT1	DDRT0
0x0203	RDRT	R W	RDRT7	RDRT6	RDRT5	RDRT4	RDRT3	RDRT2	RDRT1	RDRT0
0x0204	PERT	R W	PERT7	PERT6	PERT5	PERT4	PERT3	PERT2	PERT1	PERT0
0x0205	PPST	R W	PPST7	PPST6	PPST5	PPST4	PPST3	PPST2	PPST1	PPST0
0x0206	WOMT	R	WOMT7	WOMT6	WOMT5	WOMT4	0	MODRR2	MODRR1	MODRR0
		W								
0x0207	SRRT	R W	SRRT7	SRRT6	SRRT5	SRRT4	SRRT3	SRRT2	SRRT1	SRRT0

0x0200–0x027F Port Integration Module (PIM) Map 5 of 5

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0208	PTS	R	PTS7	PTS6	PTS5	PTS4	PTS3	PTS2	PTS1	PTS0
		W								
0x0209	PTIS	R	PTIS7	PTIS6	PTIS5	PTIS4	PTIS3	PTIS2	PTIS1	PTIS0
		W								
0x020A	DDRS	R	DDRS7	DDRS7	DDRS5	DDRS4	DDRS3	DDRS2	DDRS1	DDRS0
		W								
0x020B	RDRS	R	RDRS7	RDRS6	RDRS5	RDRS4	RDRS3	RDRS2	RDRS1	RDRS0
		W								
0x020C	PERS	R	PERS7	PERS6	PERS5	PERS4	PERS3	PERS2	PERS1	PERS0
		W								
0x020D	PPSS	R	PPSS7	PPSS6	PPSS5	PPSS4	PPSS3	PPSS2	PPSS1	PPSS0
		W								
0x020E	WOMS	R	WOMS7	WOMS6	WOMS5	WOMS4	WOMS3	WOMS2	WOMS1	WOMS0
		W								
0x020F	SRRS	R	SRRS7	SRRS6	SRRS5	SRRS4	SRRS3	SRRS2	SRRS1	SRRS0
		W								
0x0210	PTM	R	0	0	PTM5	PTM4	PTM3	PTM2	PTM1	0
		W								
0x0211	PTIM	R	0	0	PTIM5	PTIM4	PTIM3	PTIM2	PTIM1	0
		W								
0x0212	DDRM	R	0	0	DDRM5	DDRM4	DDRM3	DDRM2	DDRM1	0
		W								
0x0213	RDRM	R	0	0	RDRM5	RDRM4	RDRM3	RDRM2	RDRM1	0
		W								
0x0214	PERM	R	0	0	PERM5	PERM4	PERM3	PERM2	PERM1	0
		W								
0x0215	PPSM	R	0	0	PPSM5	PPSM4	PPSM3	PPSM2	PPSM1	0
		W								
0x0216	WOMM	R	0	0	WOMM5	WOMM4	WOMM3	WOMM2	WOMM1	0
		W								
0x0217	SRRM	R	0	0	SRRM5	SRRM4	SRRM3	SRRM2	SRRM1	0
		W								
0x0218	PTP	R	PTP7	PTP6	PTP5	PTP4	PTP3	PTP2	PTP1	PTP0
		W								
0x0219	PTIP	R	PTIP7	PTIP6	PTIP5	PTIP4	PTIP3	PTIP2	PTIP1	PTIP0
		W								
0x021A	DDRP	R	DDRP7	DDRP7	DDRP5	DDRP4	DDRP3	DDRP2	DDRP1	DDRP0
		W								
0x021B	RDRP	R	RDRP7	RDRP6	RDRP5	RDRP4	RDRP3	RDRP2	RDRP1	RDRP0
		W								
0x021C	PERP	R	PERP7	PERP6	PERP5	PERP4	PERP3	PERP2	PERP1	PERP0
		W								
0x021D	PPSP	R	PPSP7	PPSP6	PPSP5	PPSP4	PPSP3	PPSP2	PPSP1	PPSS0
		W								
0x021E	WOMP	R	WOMP7	WOMP6	WOMP5	WOMP4	0	WOMP2	0	WOMP0
		W								

0x0200–0x027F Port Integration Module (PIM) Map 5 of 5

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x021F	SRRP	R	SRRP7	SRRP6	SRRP5	SRRP4	SRRP3	SRRP2	SRRP1	SRRP0
		W								
0x0220–0x022F	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0230	PTL	R	PTL7	PTL6	PTL5	PTL4	PTL3	PTL2	PTL1	PTL0
		W								
0x0231	PTIL	R	PTIL7	PTIL6	PTIL5	PTIL4	PTIL3	PTIL2	PTIL1	PTIL0
		W								
0x0232	DDRL	R	DDRL7	DDRL6	DDRL5	DDRL4	DDRL3	DDRL2	DDRL1	DDRL0
		W								
0x0233	RDRL	R	RDRL7	RDRL6	RDRL5	RDRL4	RDRL3	RDRL2	RDRL1	RDRL0
		W								
0x0234	PERL	R	PERL7	PERL6	PERL5	PERL4	PERL3	PERL2	PERL1	PERL0
		W								
0x0235	PPSL	R	PPSL7	PPSL6	PPSL5	PPSL4	PPSL3	PPSL2	PPSL1	PPSL0
		W								
0x0236	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0237	SRRL	R	SRRL7	SRRL6	SRRL5	SRRL4	SRRL3	SRRL2	SRRL1	SRRL0
		W								
0x0238	PTU	R	PTU7	PTU6	PTU5	PTU4	PTU3	PTU2	PTU1	PTU0
		W								
0x0239	PTIU	R	PTIU7	PTIU6	PTIU5	PTIU4	PTIU3	PTIU2	PTIU1	PTIU0
		W								
0x023A	DDRU	R	DDRU7	DDRU7	DDRU	DDRU4	DDRU3	DDRU2	DDRU1	DDRU0
		W								
0x023B	SRRU	R	SRRU7	SRRU6	SRRU5	SRRU4	SRRU3	SRRU2	SRRU1	SRRU0
		W								
0x023C	PERU	R	PERU7	PERU6	PERU5	PERU4	PERU3	PERU2	PERU1	PERU0
		W								
0x023D	PPSU	R	PPSU7	PPSU6	PPSU5	PPSU4	PPSU3	PPSU2	PPSU1	PPSU0
		W								
0x023E	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x023F	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0240	PTV	R	PTV7	PTV6	PTV5	PTV4	PTV3	PTV2	PTV1	PTV0
		W								
0x0241	PTIV	R	PTIV7	PTIV6	PTIV5	PTIV4	PTIV3	PTIV2	PTIV1	PTIV0
		W								
0x0242	DDRV	R	DDRV7	DDRV7	DDRV	DDRV4	DDRV3	DDRV2	DDRV1	DDRV0
		W								
0x0243	SRRV	R	SRRV7	SRRV6	SRRV5	SRRV4	SRRV3	SRRV2	SRRV1	SRRV0
		W								
0x0244	PERV	R	PERV7	PERV6	PERV5	PERV4	PERV3	PERV2	PERV1	PERV0
		W								

0x0200–0x027F Port Integration Module (PIM) Map 5 of 5

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0245	PPSV	R	PPSV7	PPSV6	PPSV5	PPSV4	PPSV3	PPSV2	PPSV1	PPSV0
		W								
0x0246	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0247	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0248	PTW	R	PTW7	PTW6	PTW5	PTW4	PTW3	PTW2	PTW1	PTW0
		W								
0x0249	PTIW	R	PTIW7	PTIW6	PTIW5	PTIW4	PTIW3	PTIW2	PTIW1	PTIW0
		W								
0x024A	DDRW	R	DDRW7	DDRW7	DDRW	DDRW4	DDRW3	DDRW2	DDRW1	DDRW0
		W								
0x024B	SRRW	R	SRRW7	SRRW6	SRRW5	SRRW4	SRRW3	SRRW2	SRRW1	SRRW0
		W								
0x024C	PERW	R	PERW7	PERW6	PERW5	PERW4	PERW3	PERW2	PERW1	PERW0
		W								
0x024D	PPSW	R	PPSW7	PPSW6	PPSW5	PPSW4	PPSW3	PPSW2	PPSW1	PPSW0
		W								
0x024E	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x024F	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0250	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0251	PTAD	R	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
		W								
0x0252	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0253	PTIAD	R	PTIAD7	PTIAD6	PTIAD5	PTIAD4	PTIAD3	PTIAD2	PTIAD1	PTIAD0
		W								
0x0254	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0255	DDRAD	R	DDRAD7	DDRAD6	DDRAD5	DDRAD4	DDRAD3	DDRAD2	DDRAD1	DDRAD0
		W								
0x0256	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0257	RDRAD	R	RD1AD7	RDRAD6	RDRAD5	RDRAD4	RDRAD3	RDRAD2	RDRAD1	RDRAD0
		W								
0x0258	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0259	PERAD	R	PERAD7	PERAD6	PERAD5	PERAD4	PERAD3	PERAD2	PERAD1	PERAD0
		W								
0x025A	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x025B	PPSAD	R	PPSAD7	PPSAD6	PPSAD5	PPSAD4	PPSAD3	PPSAD2	PPSAD1	PPSAD0
		W								

0x0200–0x027F Port Integration Module (PIM) Map 5 of 5

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x025C	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x025D	PIEAD	R	PIEAD7	PIEAD6	PIEAD5	PIEAD4	PIEAD3	PIEAD2	PIEAD1	PIEAD0
		W								
0x025E	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x025F	PIFAD	R	PIFAD7	PIFAD6	PIFAD5	PIFAD4	PIFAD3	PIFAD2	PIFAD1	PIFAD0
		W								
0x0260–0x027F	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x0280–0x0287 Stepper Stall Detector (SSD4) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0280	RTZ4CTL	R	ITG	DCOIL	RCIR	POL	0	0	STEP	
		W								
0x0281	MDC4CTL	R	MCZIE	MODMC	RDMCL	PRE	0	MCEN	0	AOVIE
		W					FLMC			
0x0282	SSD4CTL	R	RTZE	SDCPU	SSDWAI	FTST	0	0	ACLKS	
		W								
0x0283	SSD4FLG	R	MCZIF	0	0	0	0	0	0	AOVIF
		W								
0x0284	MDC4CNT(hi)	R	MDCCNT[15:8]							
		W								
0x0285	MDC4CNT(lo)	R	MDCCNT[7:0]							
		W								
0x0286	ITG4ACC(hi)	R	ITGACC[15:8]							
		W								
0x0287	ITG4ACC(lo)	R	ITGACC[7:0]							
		W								

0x0288–0x028F Stepper Stall Detector (SSD0) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0288	RTZ0CTL	R	ITG	DCOIL	RCIR	POL	0	0	STEP	
		W								
0x0289	MDC0CTL	R	MCZIE	MODMC	RDMCL	PRE	0	MCEN	0	AOVIE
		W					FLMC			
0x028A	SSD0CTL	R	RTZE	SDCPU	SSDWAI	FTST	0	0	ACLKS	
		W								
0x028B	SSD0FLG	R	MCZIF	0	0	0	0	0	0	AOVIF
		W								
0x028C	MDC0CNT(hi)	R	MDCCNT[15:8]							
		W								

0x0288–0x028F Stepper Stall Detector (SSD0) Map (continued)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x028D	MDC0CNT(lo)	R	MDCCNT[7:0]							
		W								
0x028E	ITG0ACC(hi)	R	ITGACC[15:8]							
		W								
0x028F	ITG0ACC(lo)	R	ITGACC[7:0]							
		W								

0x0290–0x0297 Stepper Stall Detector (SSD1) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0290	RTZ1CTL	R	ITG	DCOIL	RCIR	POL	0	0	STEP	
		W								
0x0291	MDC1CTL	R	MCZIE	MODMC	RDMCL	PRE	0	MCEN	0	AOVIE
		W					FLMC			
0x0292	SSD1CTL	R	RTZE	SDCPU	SSDWAI	FTST	0	0	ACLKS	
		W								
0x0293	SSD1FLG	R	MCZIF	0	0	0	0	0	0	AOVIF
		W								
0x0294	MDC1CNT(hi)	R	MDCCNT[15:8]							
		W								
0x0295	MDC1CNT(lo)	R	MDCCNT[7:0]							
		W								
0x0296	ITG1ACC(hi)	R	ITGACC[15:8]							
		W								
0x0297	ITG1ACC(lo)	R	ITGACC[7:0]							
		W								

0x0298–0x029F Stepper Stall Detector (SSD2) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0298	RTZ2CTL	R	ITG	DCOIL	RCIR	POL	0	0	STEP	
		W								
0x0299	MDC2CTL	R	MCZIE	MODMC	RDMCL	PRE	0	MCEN	0	AOVIE
		W					FLMC			
0x029A	SSD2CTL	R	RTZE	SDCPU	SSDWAI	FTST	0	0	ACLKS	
		W								
0x029B	SSD2FLG	R	MCZIF	0	0	0	0	0	0	AOVIF
		W								
0x029C	MDC2CNT(hi)	R	MDCCNT[15:8]							
		W								
0x029D	MDC2CNT(lo)	R	MDCCNT[7:0]							
		W								
0x029E	ITG2ACC(hi)	R	ITGACC[15:8]							
		W								
0x029F	ITG2ACC(lo)	R	ITGACC[7:0]							
		W								

0x02A0–0x02A7 Stepper Stall Detector (SSD3) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x02A0	RTZ3CTL	R	ITG	DCOIL	RCIR	POL	0	0	STEP	
		W								
0x02A1	MDC3CTL	R	MCZIE	MODMC	RDMCL	PRE	0	MCEN	0	AOVIE
		W					FLMC			
0x02A2	SSD3CTL	R	RTZE	SDCPU	SSDWAI	FTST	0	0	ACLKS	
		W								
0x02A3	SSD3FLG	R	MCZIF	0	0	0	0	0	0	AOVIF
		W								
0x02A4	MDC3CNT(hi)	R	MDCCNT[15:8]							
		W								
0x02A5	MDC3CNT(lo)	R	MDCCNT[7:0]							
		W								
0x02A6	ITG3ACC(hi)	R	ITGACC[15:8]							
		W								
0x02A7	ITG3ACC(lo)	R	ITGACC[7:0]							
		W								

0x02A8–0x02AF Stepper Stall Detector (SSD5) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x02A8	RTZ5CTL	R	ITG	DCOIL	RCIR	POL	0	0	STEP	
		W								
0x02A9	MDC5CTL	R	MCZIE	MODMC	RDMCL	PRE	0	MCEN	0	AOVIE
		W					FLMC			
0x02AA	SSD5CTL	R	RTZE	SDCPU	SSDWAI	FTST	0	0	ACLKS	
		W								
0x02AB	SSD5FLG	R	MCZIF	0	0	0	0	0	0	AOVIF
		W								
0x02AC	MDC5CNT(hi)	R	MDCCNT[15:8]							
		W								
0x02AD	MDC5CNT(lo)	R	MDCCNT[7:0]							
		W								
0x02AE	ITG5ACC(hi)	R	ITGACC[15:8]							
		W								
0x02AF	ITG5ACC(lo)	R	ITGACC[7:0]							
		W								

0x02B0–0x02EF Reserved

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x02B0–0x02EF	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x02F0–0x02F7 Voltage Regulator (VREG_3V3) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x02F0	VREGHTCL	R	Reserved for Factory Test							
		W								
0x02F1	VREGCTRL	R	0	0	0	0	0	LVDS	LVIE	LVIF
		W								
0x02F2	VREGAPICL	R	APICLK	0	0	0	0	APIFE	APIE	APIF
		W								
0x02F3	VREGAPITR	R	APITR5	APITR4	APITR3	APITR2	APITR1	APITR0	0	0
		W								
0x02F4	VREGAPIRH	R	0	0	0	0	APIR11	APIR10	APIR9	APIR8
		W								
0x02F5	VREGAPIRL	R	APIR7	APIR6	APIR5	APIR4	APIR3	APIR2	APIR1	APIR0
		W								
0x02F6	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x02F7	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x02F8–0x02FF Reserved

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x02F8– 0x02FF	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x0300–0x0327 Pulse Width Modulator 8-Bit 8-Channel (PWM) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0300	PWME	R	PWME7	PWME6	PWME5	PWME4	PWME3	PWME2	PWME1	PWME0
		W								
0x0301	PWMPOL	R	PPOL7	PPOL6	PPOL5	PPOL4	PPOL3	PPOL2	PPOL1	PPOL0
		W								
0x0302	PWMCLK	R	PCLK7	PCLK6	PCLK5	PCLK4	PCLK3	PCLK2	PCLK1	PCLK0
		W								
0x0303	PWMPRCLK	R	0	PCKB2	PCKB1	PCKB0	0	PCKA2	PCKA1	PCKA0
		W								
0x0304	PWMCAE	R	CAE7	CAE6	CAE5	CAE4	CAE3	CAE2	CAE1	CAE0
		W								
0x0305	PWMCTL	R	CON67	CON45	CON23	CON01	PSWAI	PFRZ	0	0
		W								
0x0306	PWMTST Test Only	R	0	0	0	0	0	0	0	0
		W								
0x0307	PWMPRSC	R	0	0	0	0	0	0	0	0
		W								
0x0308	PWMSCLA	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0309	PWMSCLB	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								

0x0300–0x0327 Pulse Width Modulator 8-Bit 8-Channel (PWM) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x030A	PWMSCNTA	R	0	0	0	0	0	0	0	0
		W								
0x030B	PWMSCNTB	R	0	0	0	0	0	0	0	0
		W								
0x030C	PWMCNT0	R	Bit 7	6	5	4	3	2	1	Bit 0
		W	0	0	0	0	0	0	0	0
0x030D	PWMCNT1	R	Bit 7	6	5	4	3	2	1	Bit 0
		W	0	0	0	0	0	0	0	0
0x030E	PWMCNT2	R	Bit 7	6	5	4	3	2	1	Bit 0
		W	0	0	0	0	0	0	0	0
0x030F	PWMCNT3	R	Bit 7	6	5	4	3	2	1	Bit 0
		W	0	0	0	0	0	0	0	0
0x0310	PWMCNT4	R	Bit 7	6	5	4	3	2	1	Bit 0
		W	0	0	0	0	0	0	0	0
0x0311	PWMCNT5	R	Bit 7	6	5	4	3	2	1	Bit 0
		W	0	0	0	0	0	0	0	0
0x0312	PWMCNT6	R	Bit 7	6	5	4	3	2	1	Bit 0
		W	0	0	0	0	0	0	0	0
0x0313	PWMCNT7	R	Bit 7	6	5	4	3	2	1	Bit 0
		W	0	0	0	0	0	0	0	0
0x0314	PWMPER0	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0315	PWMPER1	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0316	PWMPER2	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0317	PWMPER3	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0318	PWMPER4	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0319	PWMPER5	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x031A	PWMPER6	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x031B	PWMPER7	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x031C	PWMDTY0	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x031D	PWMDTY1	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x031E	PWMDTY2	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x031F	PWMDTY3	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0320	PWMDTY4	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								

0x0300–0x0327 Pulse Width Modulator 8-Bit 8-Channel (PWM) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0321	PWMDTY5	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0322	PWMDTY6	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0323	PWMDTY7	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0324	PWMSDN	R	PWMIF	PWMIE	0	PWMLVL	0	PWM7IN	PWM7INL	PWM7 ENA
		W			PWM RSTRT					
0x0325	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0326	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0327	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x0328–0x033F Reserved

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0328–0x033F	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x0340–0x0367 Periodic Interrupt Timer (PIT) Map

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0340	PITCFLMT	R	PITE	PITSWAI	PITFRZ	0	0	0	0	0
		W							PFLMT1	PFLMT0
0x0341	PITFLT	R	0	0	0	0	0	0	0	0
		W					PFLT3	PFLT2	PFLT1	PFLT0
0x0342	PITCE	R	0	0	0	0	PCE3	PCE2	PCE1	PCE0
		W								
0x0343	PITMUX	R	0	0	0	0	PMUX3	PMUX2	PMUX1	PMUX0
		W								
0x0344	PITINTE	R	0	0	0		PINTE3	PINTE2	PINTE1	PINTE0
		W								
0x0345	PITTF	R	0	0	0	0	PTF3	PTF2	PTF1	PTF0
		W								
0x0346	PITMTLD0	R	PMTLD7	PMTLD6	PMTLD5	PMTLD4	PMTLD3	PMTLD2	PMTLD1	PMTLD0
		W								
0x0347	PITMTLD1	R	PMTLD7	PMTLD6	PMTLD5	PMTLD4	PMTLD3	PMTLD2	PMTLD1	PMTLD0
		W								
0x0348	PITLD0 (hi)	R	PLD15	PLD14	PLD13	PLD12	PLD11	PLD10	PLD9	PLD8
		W								
0x0349	PITLD0 (lo)	R	PLD7	PLD6	PLD5	PLD4	PLD3	PLD2	PLD1	PLD0
		W								

0x0340–0x0367 Periodic Interrupt Timer (PIT) Map (continued)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x034A	PITCNT0 (hi)	R W	PCNT15	PCNT14	PCNT13	PCNT12	PCNT11	PCNT10	PCNT9	PCNT8
0x034B	PITCNT0 (lo)	R W	PCNT7	PCNT6	PCNT5	PCNT4	PCNT3	PCNT2	PCNT1	PCNT0
0x034C	PITLD1 (hi)	R W	PLD15	PLD14	PLD13	PLD12	PLD11	PLD10	PLD9	PLD8
0x034D	PITLD1 (lo)	R W	PLD7	PLD6	PLD5	PLD4	PLD3	PLD2	PLD1	PLD0
0x034E	PITCNT1 (hi)	R W	PCNT15	PCNT14	PCNT13	PCNT12	PCNT11	PCNT10	PCNT9	PCNT8
0x034F	PITCNT1 (lo)	R W	PCNT7	PCNT6	PCNT5	PCNT4	PCNT3	PCNT2	PCNT1	PCNT0
0x0350	PITLD2 (hi)	R W	PLD15	PLD14	PLD13	PLD12	PLD11	PLD10	PLD9	PLD8
0x0351	PITLD2 (lo)	R W	PLD7	PLD6	PLD5	PLD4	PLD3	PLD2	PLD1	PLD0
0x0352	PITCNT2 (hi)	R W	PCNT15	PCNT14	PCNT13	PCNT12	PCNT11	PCNT10	PCNT9	PCNT8
0x0353	PITCNT2 (lo)	R W	PCNT7	PCNT6	PCNT5	PCNT4	PCNT3	PCNT2	PCNT1	PCNT0
0x0354	PITLD3 (hi)	R W	PLD15	PLD14	PLD13	PLD12	PLD11	PLD10	PLD9	PLD8
0x0355	PITLD3 (lo)	R W	PLD7	PLD6	PLD5	PLD4	PLD3	PLD2	PLD1	PLD0
0x0356	PITCNT3 (hi)	R W	PCNT15	PCNT14	PCNT13	PCNT12	PCNT11	PCNT10	PCNT9	PCNT8
0x0357	PITCNT3 (lo)	R W	PCNT7	PCNT6	PCNT5	PCNT4	PCNT3	PCNT2	PCNT1	PCNT0
0x0358– 0x0367	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x0368–0x037F Reserved

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0368– 0x037F	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x0380–0x03BF XGATE Map (Sheet 1 of 3)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0380	XGMCTL	R	0	0	0	0	0	0	0	XGIEM
		W	XGEM	XGFRZM	XGDBGM	XGSSM	XGFACTM		XGS WEIFM	
0x0381	XGMCTL	R	XGE	XGFRZ	XGDBG	XGSS	XGFACT	0	XGSWEIF	XGIE
		W								
0x0382	XGCHID	R	0	XGCHID[6:0]						
		W								
0x0383	Reserved	R								
		W								
0x0384	Reserved	R								
		W								
0x0385	Reserved	R								
		W								
0x0386	XGVBR	R	XGVBR[15:8]							
		W								
0x0387	XGVBR	R	XGVBR[7:1]							0
		W								
0x0388	XGIF	R	0	0	0	0	0	0	0	XGIF_78
		W								
0x0389	XGIF	R	XGIF_77	XGIF_76	XGIF_75	XGIF_74	XGIF_73	XGIF_72	XGIF_71	XGIF_70
		W								
0x038A	XGIF	R	XGIF_6F	XGIF_6E	XGIF_6D	XGIF_6C	XGIF_6B	XGIF_6A	XGIF_69	XGIF_68
		W								
0x023B	XGIF	R	XGIF_67	XGIF_66	XGIF_65	XGIF_64	XGIF_63	XGIF_62	XGIF_61	XGIF_60
		W								
0x023C	XGIF	R	XGIF_5F	XGIF_5E	XGIF_5D	XGIF_5C	XGIF_5B	XGIF_5A	XGIF_59	XGIF_58
		W								
0x038D	XGIF	R	XGIF_57	XGIF_56	XGIF_55	XGIF_54	XGIF_53	XGIF_52	XGIF_51	XGIF_50
		W								
0x038E	XGIF	R	XGIF_4F	XGIF_4E	XGIF_4D	XGIF_4C	XGIF_4B	XGIF_4A	XGIF_49	XGIF_48
		W								
0x038F	XGIF	R	XGIF_47	XGIF_46	XGIF_45	XGIF_44	XGIF_43	XGIF_42	XGIF_41	XGIF_40
		W								
0x0390	XGIF	R	XGIF_3F	XGIF_3E	XGIF_3D	XGIF_3C	XGIF_3B	XGIF_3A	XGIF_39	XGIF_38
		W								
0x0391	XGIF	R	XGIF_37	XGIF_36	XGIF_35	XGIF_34	XGIF_33	XGIF_32	XGIF_31	XGIF_30
		W								
0x0392	XGIF	R	XGIF_2F	XGIF_2E	XGIF_2D	XGIF_2C	XGIF_2B	XGIF_2A	XGIF_29	XGIF_28
		W								
0x0393	XGIF	R	XGIF_27	XGIF_26	XGIF_25	XGIF_24	XGIF_23	XGIF_22	XGIF_21	XGIF_20
		W								
0x0394	XGIF	R	XGIF_1F	XGIF_1E	XGIF_1D	XGIF_1C	XGIF_1B	XGIF_1A	XGIF_19	XGIF_18
		W								
0x0395	XGIF	R	XGIF_17	XGIF_16	XGIF_15	XGIF_14	XGIF_13	XGIF_12	XGIF_11	XGIF_10
		W								

0x0380–0x03BF XGATE Map (Sheet 2 of 3)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0396	XGIF	R	XGIF_0F	XGIF_0E	XGIF_0D	XGIF_0C	XGIF_0B	XGIF_0A	XGIF_09	0
		W								
0x0397	XGIF	R	0	0	0	0	0	0	0	0
		W								
0x0398	XGSWT (hi)	R	0	0	0	0	0	0	0	0
		W	XGSWTM[7:0]							
0x0399	XGSWT (lo)	R	XGSWT[7:0]							
		W								
0x039A	XGSEM (hi)	R	0	0	0	0	0	0	0	0
		W	XGSEMM[7:0]							
0x039B	XGSEM (lo)	R	XGSEM[7:0]							
		W								
0x039C	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x039D	XGCCR	R	0	0	0	0	XGN	XGZ	XGV	XGC
		W								
0x039E	XGPC (hi)	R	XGPC[15:8]							
		W								
0x039F	XGPC (lo)	R	XGPC[7:0]							
		W								
0x03A0	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x03A1	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x03A2	XGR1 (hi)	R	XGR1[15:8]							
		W								
0x03A3	XGR1 (lo)	R	XGR1[7:0]							
		W								
0x03A4	XGR2 (hi)	R	XGR2[15:8]							
		W								
0x03A5	XGR2 (lo)	R	XGR2[7:0]							
		W								
0x03A6	XGR3 (hi)	R	XGR3[15:8]							
		W								
0x03A7	XGR3 (lo)	R	XGR3[7:0]							
		W								
0x03A8	XGR4 (hi)	R	XGR4[15:8]							
		W								
0x03A9	XGR4 (lo)	R	XGR4[7:0]							
		W								
0x03AA	XGR5 (hi)	R	XGR5[15:8]							
		W								
0x03AB	XGR5(lo)	R	XGR5[7:0]							
		W								
0x03AC	XGR6 (hi)	R	XGR6[15:8]							
		W								

0x0380–0x03BF XGATE Map (Sheet 3 of 3)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x03AD	XGR6 (lo)	R W	XGR6[7:0]							
0x03AE	XGR7 (hi)	R W	XGR7[15:8]							
0x03AF	XGR7 (lo)	R W	XGR7[7:0]							
0x03B0– 0x03BF	Reserved	R	0	0	0	0	0	0	0	0
		W								

0x03C0–0x07FF Reserved

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x03C0 –0x07FF	Reserved	R	0	0	0	0	0	0	0	0
		W								

How to Reach Us:

Home Page:
www.freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
1-800-521-6274 or 480-768-2130

Europe, Middle East, and Africa:
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)

Japan:
Freescale Semiconductor Japan Ltd.
Technical Information Center
3-20-1, Minami-Azabu, Minato-ku
Tokyo 106-0047, Japan
0120-191014 or +81-3-3440-3569

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
852-26668334

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150

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

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. The ARM POWERED logo is a registered trademark of ARM Limited. ARM7TDMI-S is a trademark of ARM Limited. Java and all other Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. The Bluetooth trademarks are owned by their proprietor and used by Freescale Semiconductor, Inc. under license.

© Freescale Semiconductor, Inc. 2007. All rights reserved.

MC9S12XHZ512
Rev. 1.06
10/2010