Freescale Semiconductor

MCF51EM256RM
Rev. 8.1, 05/2012

MCF51EM256 Reference Manual

This is the MCF51EM256 Reference Manual set consisting of the following files:

- MCF51EM256 Reference Manual Addendum, Rev 2
- MCF51EM256 Reference Manual, Rev 8

freescale™

Freescale Semiconductor
Reference Manual Addendum

MCF51EM256 RMAD
Rev. 2, 05/2012

MCF51EM256 Reference Manual Addendum

This addendum describes corrections or updates to the *MCF51EM256 Reference Manual*, file named as MCF51EM256RM. Please check our website at http://www.freescale.com/coldfire, for the latest updates.

The current version available of the *MCF51EM256 Reference Manual* is Revision 8.0.

Table of Contents

1 Addendum for Revision 8.0

Table 1. MCF51EM256RM Rev 8.0 Addendum

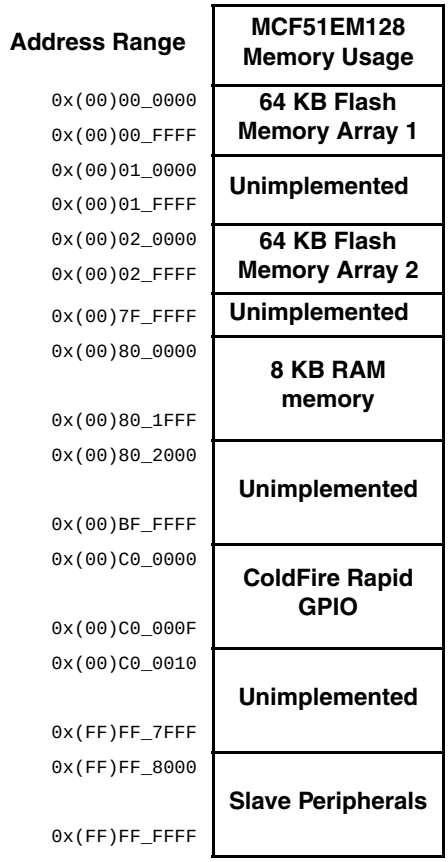
Location	Description
Figure 3-1, “MCF51EM256 Series Memory Maps” in Chapter 3, “Memory”	Updated the memory map, to reflect the actual EM128 silicon. The updated section of the figure is depicted below: 

Table 1. MCF51EM256RM Rev 8.0 Addendum

Location	Description

Sub-section “Calibration Procedure for Improved Linearity ” for Chapter 21 ”Analog-to-Digital Converter (ADC16)”	Added a new sub-section to “Calibration Function” section:
	For applications using the ADC16 in differential mode, improved linearity may be achieved by using an adjusted calibration procedure as detailed below. The ADC16 does perform to the published datasheet specification using the original calibration procedure. The adjusted calibration procedure corrects potential calibration offset errors and diminishes linearity error spikes that may occur near the ¼, ½ and ¾ point of the full scale range.
	
	Adjusted calibration procedure:
	
	• Perform auto calibration as defined in the reference manual.
	• Then, rewrite ADCCLP1-4, ADCCLM0-4, ADCPG and ADCMG using ADCLP0 and the following calculations:
	
	ADCCLP1 = ADCCLP0 << 1; /* CLP1 is 2x CLP0 */
	ADCCLP2 = ADCCLP1 << 1; /* CLP2 is 2x CLP1 */
	ADCCLP3 = ADCCLP2 << 1; /* CLP3 is 2x CLP2 */
	ADCCLP4 = ADCCLP3 << 1; /* CLP4 is 2x CLP3 */
	ADCCLM0 = ADCCLP0; /* minus side calibration values are set equal to the plus side */
	ADCCLM1 = ADCCLP1; /* minus side calibration values are set equal to the plus side */
	ADCCLM2 = ADCCLP2; /* minus side calibration values are set equal to the plus side */
	ADCCLM3 = ADCCLP3; /* minus side calibration values are set equal to the plus side */
	ADCCLM4 = ADCCLP4; /* minus side calibration values are set equal to the plus side */
	
	ADCCLMD = ADCCLPD;
	ADCCLMS = ADCCLPS;
	
	calSum = ADCCLP0 + ADCCLP1 + ADCCLP2 + ADCCLP3 + ADCCLP4 + ADCCLPS; /* recalculate the plus gain factor */
	calSum /= 2;
	calSum += 0x8000;
	ADCPG = calSum;
	
	calSum = 0;
	calSum = ADCCLM0 + ADCCLM1 + ADCCLM2 + ADCCLM3 + ADCCLM4 + ADCCLMS; /* recalculate the minus gain factor */
	calSum /= 2;
	calSum += 0x8000;
	ADCMG = calSum;

2 Revision History

Table 2 provides a revision history for this document.

Table 2. Revision History Table

Rev. Number	Substantive Changes	Date of Release
1.0	Initial release. Updated chapter 3, “Memory.”	01/2012
2.0	Added a new section “Calibration Procedure for Improved Linearity ” for Chapter 21 ”Analog-to-Digital Converter (ADC16)”.	05/2012

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

MCF51EM256 RMAD
Rev. 2
05/2012

MCF51EM256 Series ColdFire® Integrated Microcontroller Reference Manual

Devices Supported:

MCF51EM256

MCF51EM128

Document Number: MCF51EM256RM
Rev. 8
4/2010

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd. Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

MCF51EM256RM
Rev. 8
4/2010

Chapter 1
Device Overview

Chapter 2
Pins and Connections

Chapter 3
Memory

Chapter 4
Parallel Input/Output Control

Chapter 5
Rapid GPIO (RGPIO)

Chapter 6
Modes of Operation

Chapter 7
Resets, Interrupts, and General System Control

Chapter 8
ColdFire Core

Chapter 9
Multiply-Accumulate Unit (MAC)

Chapter 10
Interrupt Controller (CF1_INTC)

Chapter 11
Internal Clock Source (ICS)

Chapter 12
8-Bit Serial Peripheral Interface (SPI)

Chapter 13
8- or 16-Bit Serial Peripheral Interface (SPI16)

Chapter 14
Serial Communication Interface (SCI)

Chapter 15
Inter-Integrated Circuit (IIC)

Chapter 16
Timer/PWM Module(TPM)

Chapter 17
Independent Real Time Clock (IRTC)

Chapter 18
8-Bit Modulo Timer (MTIM)

Chapter 19
16-Bit Modulo Timer (MTIM16)

Chapter 20
Cyclic Redundancy Check (CRC)

Chapter 21
Analog-to-Digital Converter (ADC16)

Chapter 22
Programmable Delay Block (PDB)

Chapter 23
Voltage Reference (VREF)

Chapter 24
LCD Driver Module

Chapter 25
Programmable Reference Analog Comparator (PRACMP)

Chapter 26
Version 1 ColdFire Debug (CF1_DEBUG)

Appendix A
Revision History

Chapter 1
Device Overview

This chapter provides a brief introduction to the MCF51EM256 series devices. Top level features and block diagrams are shown; the V1 Coldfire CPU is discussed; and the subject of system clocks is introduced. Later chapters provide details on memory maps, pinouts, and specific sub-systems.

1.1 MCF51EM256 Series Microcontrollers

1.1.1 Definition

The MCF51EM256 series microcontrollers are systems-on-chips (SoCs) that are based on the V1 ColdFire core and the following features:

- Operate at processor core speeds up to 50.33 MHz (peripherals operate at half of this speed) at 3.6 V to 2.5 V and 20 MHz at 2.5 V to 1.8 V
- ColdFire V1 core with MAC unit
- LCD module driver
- Up to 256 KB of flash memory
- Up to 16 KB of RAM
- Independent RTC with separate time base, power domain, and 32 bytes of RAM
- A collection of communications peripherals, including UART, IIC and SPI
- Integrated 16-bit SAR analog-to-digital converter
- Infrared communication modulation/demodulation support by the interconnection of SCI, TPM and PRACMP modules

These devices are ideal for use in energy meters and monitoring applications.

1.1.2 MCF51EM256 Series Devices

The MCF51EM256 series devices are in various packages, as shown in Table 1-1.

Table 1-1. Package Availability of MCF51EM256 Series

Device Number	100-Pin LQFP	80-Pin LQFP
MCF51EM256	X	X
MCF51EM128	X	X

1.1.3 MCF51EM256 Series Device Comparison

Table 1-2 summarizes the feature set available in the MCF51EM256 series MCUs.

Table 1-2. . MCF51EM256 Series Features by MCU and Package

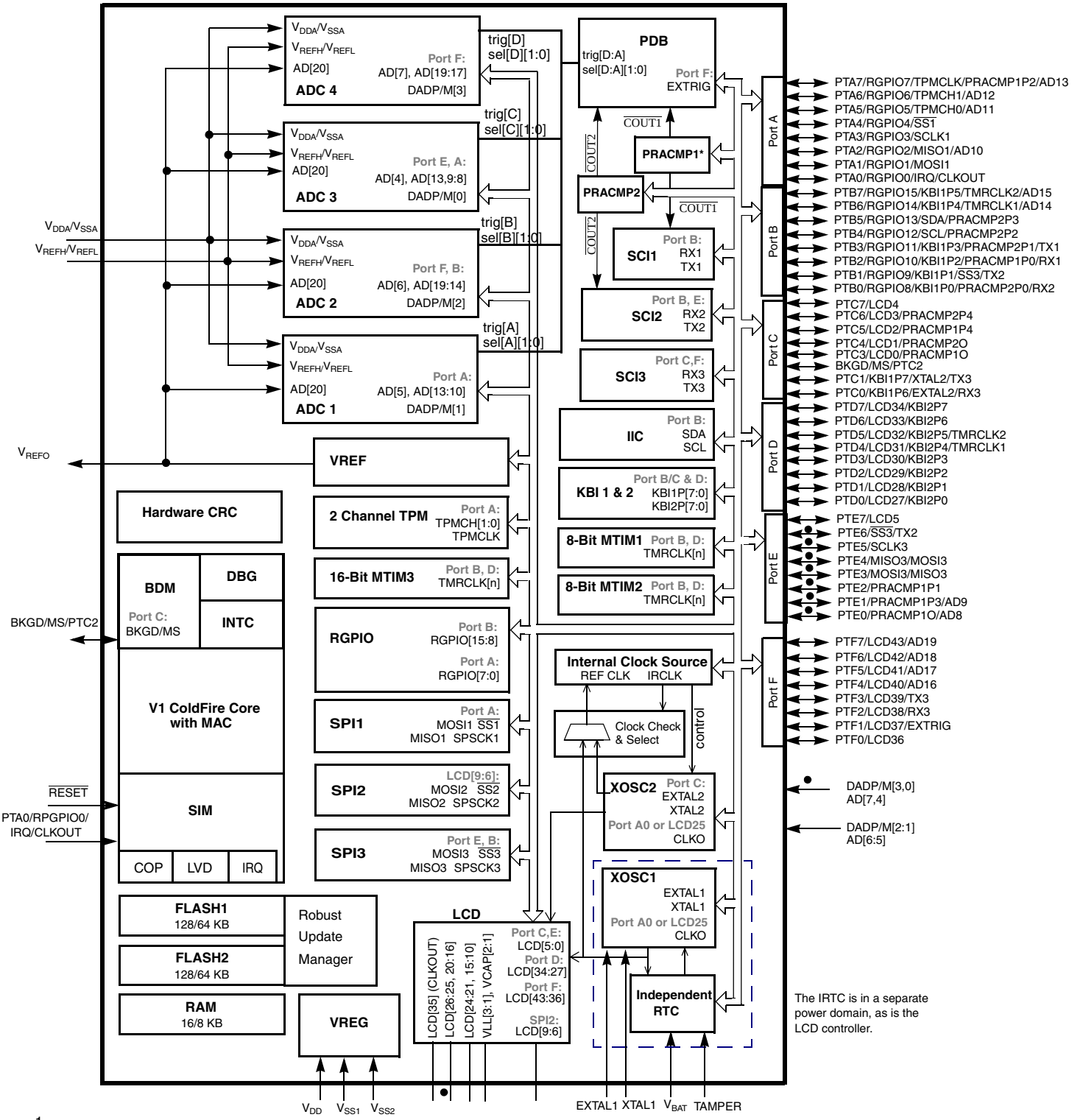
Feature	MCF51EM256		MCF51EM128	
Flash size (bytes)	262144		131072	
RAM size (bytes)	16384		8192	
Robust flash update supported	Yes			
Pin quantity	100	80	100	80
PRACMP1 inputs	5	3	5	3
PRACMP2 inputs	5			
ADC modules	4		4	
ADC differential channels[1]	4	2	4	2
ADC single-ended channels	16	12	16	12
DBG	Yes			
ICS	Yes			
IIC	Yes			
IRQ	Yes			
IRTC	Yes			
VREF	Yes			
LCD drivers	44	37	44	37
Rapid GPIO[2]	16	16	16	16
Port I/O[3]	47	40	47	40
Keyboard interface 1	8			
Keyboard interface 2	8			
SCI1	Yes			
SCI2	Yes			
SCI3	Yes			
SPI1 (FIFO)	Yes			
SPI2 (standard)	Yes			
SPI3 (standard)	Yes	No	Yes	No
MTIM1 (8-bit)	Yes			
MTIM2 (8-bit)	Yes			

Table 1-2. . MCF51EM256 Series Features by MCU and Package (continued)

Feature	MCF51EM256	MCF51EM128
MTIM3 (16-it)	Yes	
TPM channels	2	
PDB	Yes	
XOSC1[4]	Yes	
XOSC2[5]	Yes	

1 Each differential channel is comprised of 2 pin inputs
2 RGPIO is muxed with standard Port I/O
3 Port I/O count does not include the ouput only PTC2/BKGD/MS.
4 IRTC crystal input and possible crystal input to the ICS module
5 Main external crystal input for the ICS module

Figure 1-1 shows the connections between the MCF51EM256 series MCU pins and functional units.



1 Pins with • are not present on 80-pin devices.

2 PRACMP1 has two less available inputs on the 80-pin devices.

Figure 1-1. MCF51EM256 Series Family Block Diagram

1.1.4 Functional Units

Table 1-3 describes the functional units of the MCF51EM256 series microcontrollers.

Table 1-3. MCF51EM256 Series Functional Units

Unit	Function
ADC (analog-to-digital converter)	Measures analog voltages at up to 16 bits of resolution. Each ADC has up to four differential and 24 single-ended inputs.
BDM (background debug module)	Provides single pin debugging interface (part of the V1 ColdFire core)
CF1 CORE (V1 ColdFire core) with MAC unit	Executes programs, handles interrupts and containes multiply-accumulate hardware (MAC).
PRACMP1, PRACMP2 (comparators)	Analog comparators for comparing external analog signals against each other, or a variety of reference levels.
COP (computer operating poperly)	Software watchdog
IRQ (interrupt request)	Single pin high priority interrupt (part of the V1 ColdFire core)
CRC (cyclic Redundancy Check)	High-speed CRC calculation
DBG (debug)	Provides debugging and emulation capabilities (part of the V1 ColdFire core)
FLASH (flash memory)	Provides storage for program code, constants and variables
IIC (inter-integrated circuits)	Supports standard IIC communications protocol and SMBus
INTC (interrupt controller)	Controls and prioritizes all device interrupts
KBI1 & KBI2	Keyboard Interfaces 1 and 2
LCD	Liquid crystal display driver
LVD (low voltage detect)	Provides an interrupt to the CF1CORE in the event that the supply voltage drops below a critical value. The LVD can also be programmed to reset the device upon a low voltage event
ICS (internal clock source)	Provides clocking options for the device, including a three frequency-locked loops (FLLs) for multiplying slower reference clock sources
IRTC (independent real-time clock)	The independent real time clock provides an independent time-base with optional interrupt, battery backup and tamper protection
VREF (voltage reference)	The voltage reference output is available for both on and off-chip use
MTIM1, MTIM2 (modulo timers)	8-bit modulo timers with configurable clock inputs and interrupt generation on overflow
MTIM3 (modulo timer)	16-bit modulo timer with configurable clock inputs and interrupt generation on overflow
PDB (programmable delay block)	This timer is optimized for scheduling ADC conversions
RAM (random-access memory)	Provides stack and variable storage
RGPIO (rapid general-purpose input/output)	Allows for I/O port access at CPU clock speeds and is used to implement GPIO functionality for PTA and PTB.

Table 1-3. MCF51EM256 Series Functional Units (continued)

Unit	Function
SCI1, SCI2, SCI3(serial communications interfaces)	Serial communications UARTs capable of supporting RS-232 and LIN protocols
SIM (system integration unit)	
SPI1 (FIFO), SPI2, SPI3 (serial peripheral interfaces)	SPI1 has full-complementary drive outputs. SPI2 may be configured with full-complementary drive output via LCD control registers. SPI3 has open drain outputs on SCLK and (MISO or MOSI). These coupled with off-chip pullup resistors, allow interface to an external 5V SPI.
TPM (Timer/PWM Module)	Timer/PWM module can be used for a variety of generic timer operations as well as pulse-width modulation
VREG (voltage regulator)	Controls power management across the device
XOSC1 and XOSC2 (crystal oscillators)	These devices incorporate redundant crystal oscillators in separate power domains.One is intended primarily for use by the IRTC, and the other by the CPU and other peripherals.

1.1.5 Module Versions

Table 1-4 shows the functional versions of the on-chip peripherals

Table 1-4. Versions of On-Chip Modules

Module	Version
Analog-to-Digital Converter (ADC16)	1
Central Processing Unit — V1 ColdFire Central Processing Unit (CPU)	2
Independent Real-Time Clock (IRTC)	1
Inter-Integrated Circuit (IIC)	3
Internal Clock Source (ICS)	3
Low Power Oscillator (XOSC1, XOSC2)	1
8-Bit Modulo Timer (MTIM1 & MTIM2)	1
16-Bit Modulo Timer (MTIM3)	1
V1 ColdFire In-Circuit Debug/Emulator (DBG)	1
Programmable Reference Analog Comparator (PRACMP1 & PRACMP2)	1
Serial Communications Interface (SCI1, SCI2, SCI3)	4
8- or 16 Bit Serial Peripheral Interface (SPI1)	2
8-Bit Serial Peripheral Interface (SPI2, SPI3)	4
Timer Pulse-Width Modulator (TPM)	3
LCD Display Driver	1
Keyboard Interrupt(KBI1,KBI2)	2

Table 1-4. Versions of On-Chip Modules (continued)

Module	Version
Voltage Reference (VREF)	1
Voltage Regulator (VREG)	1
Interrupt Request (IRQ)	3
Programmable Delay Block (PDB)	1
System Integration Module (SIM)	1
Cyclic Redundancy Check	3
Flash Wrapper	1
GPIO	2
Port Control	1

1.2 V1 ColdFire Core

The MCF51EM256 series devices contain a version of the V1 ColdFire platform that is optimized for area and low power. The CPU implements ColdFire instruction set architecture revision C (ISA_C) with added capabilities:

- Hardware MAC support for 16×16 ± 32 and 32×32 ± 32 bit multiply-accumulate operations (32-bit accumulator)
- Upward compatibility with all other ColdFire cores (V2–V5)

An integrated multi-master crossbar switch on the ColdFire system busses provides access to system resources by the CPU.

For more details on the V1 ColdFire core, see Chapter 8, “ColdFire Core.”

1.2.1 User Programming Model

Table 1-5 illustrates the integer portion of the user programming model. It consists of the following registers:

- 16 general-purpose 32-bit registers (D0–D7, A0–A7)
- 32-bit program counter (PC)
- 8-bit condition code register (CCR)

1.2.2 Supervisor Programming Model

System programmers use the supervisor programming model to implement operating system functions. All accesses that affect the control features of ColdFire processors must be made in supervisor mode and can be accessed only by privileged instructions. The supervisor programming model consists of the registers available in user mode as well as the registers listed in Table 1-5.

Table 1-5. Version 1 ColdFire CPU Programming Model

Register	Width (bits)	Reset Value
Supervisor/User Registers		
Data Register 0 (D0)	32	0xCF1*_**29[1]
Data Register 1 (D1)	32	0x010*0_10*0[1]
Data Register 2–7 (D2–D7)	32	Undefined
Address Register 0–6 (A0–A6)	32	Undefined
Supervisor/User A7 Stack Pointer (A7)	32	Undefined
Program Counter (PC)	32	Contents of memory at 0x00_0004
Condition Code Register (CCR)	8	Undefined
Supervisor Registers		
Status Register (SR)	16	0x27--
User/Supervisor A7 Stack Pointer (OTHER_A7)	32	Contents of memory at 0x00_0000
Vector Base Register (VBR)	32	0x0000_0000
CPU Configuration Register (CPUCR)	32	0x0000_0000

[1] * depends on the configurations and/or flash/RAM size, refer to Chapter 8, “ColdFire Core,” for details of the configurations.

1.3 System Clock Generation and Distribution

1.3.1 Internal Clock Source

Figure 1-2 shows a simplified view of the internal clock source module.

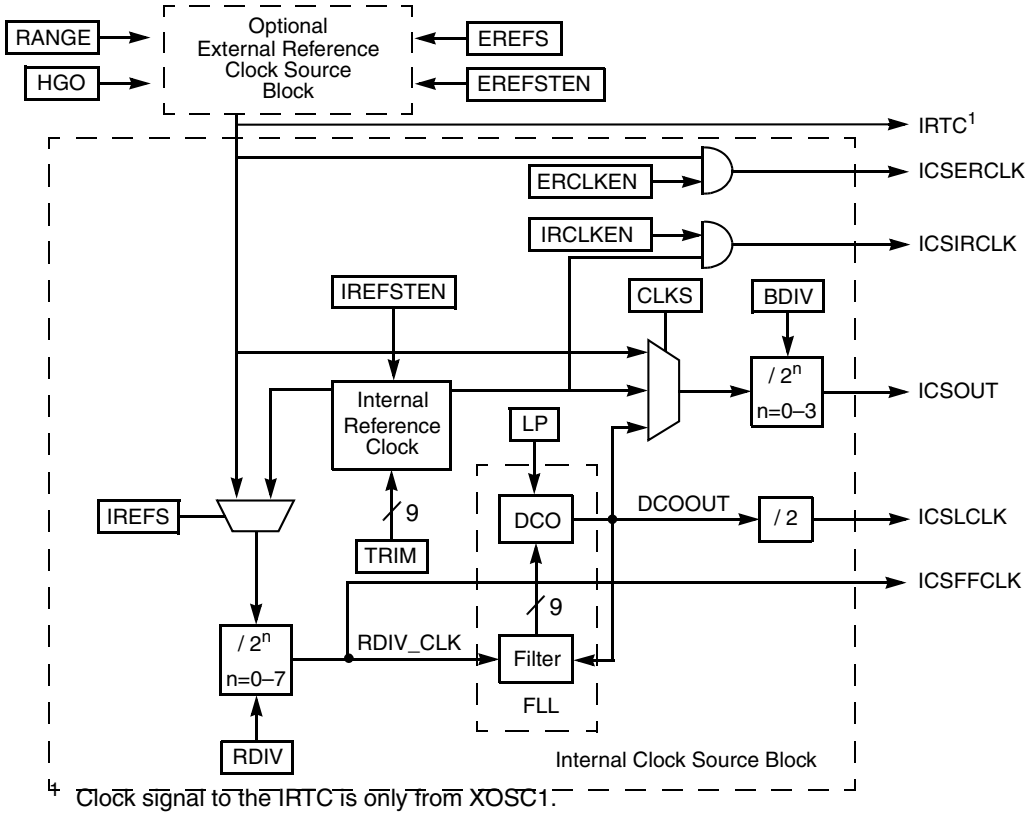
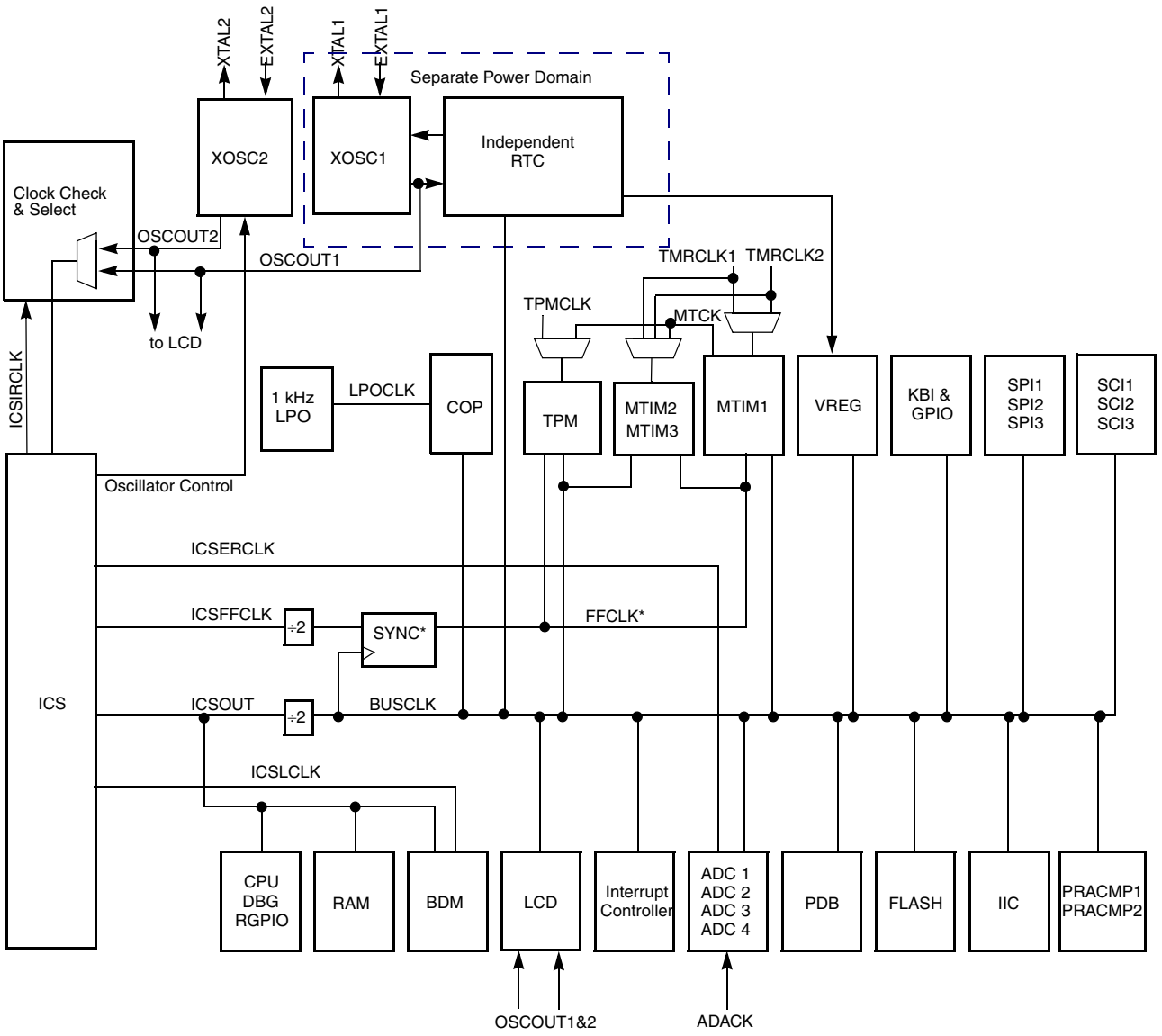


Figure 1-2. Simplified ICS Block Diagram

1.3.2 Clock Distribution Diagram

Figure 1-3 shows how clocks from the ICS and XOSC are distributed to the microcontroller's other functional units. Some modules in the microcontroller have selectable clock inputs. All memory-mapped registers associated with the modules (except RGPIO) are clocked with BUSCLK. The RGPIO registers are clocked with the CPU clock (ICSOUT).

A unique feature of this product family is the independent real time clock (IRTC) and secondary 32 kHz crystal oscillator operating on an independent power domain. The IRTC can be powered by an independent coin battery, and will continue to count even when the main CPU supply is down. The "Clock Check & Select" function allows the CPU to check that oscillator outputs are active prior to switching to one of the two as the reference for the ICS.



Note: The ADCs have minimum and maximum frequency requirements. See the ADC chapter and the *MCF51EM256 Series Data Sheet*. Flash memory has frequency requirements for program and erase operations. Each ADC also has its own internal asynchronous clock source, which is not shown above.

* The fixed frequency clock (FFCLK) is internally synchronized to the bus clock (BUSCLK) and must not exceed one half of the bus clock frequency.

Figure 1-3. Clock Distribution Diagram

1.3.3 System Clocks

Table 1-6 describes each of the system clocks.

Table 1-6. System Clocks

Clock	Description
OSCOUT1	This is the direct output of XOSC1 and is used as the clock source for the independent real time clock (IRTC) module and LCD. This signal can also be consumed by the ICS. See Chapter 11, "Internal Clock Source (ICS) and Chapter 17, "Independent Real Time Clock (IRTC)," for details.
OSCOUT2	This is the direct output of XOSC2, and is normally consumed by the ICS. It can also be consumed by the LCD module.
ICSOUT	This clock source is used as the CPU clock and is divided by 2 to generate the peripheral bus clock. Control bits in the ICS control registers determine which of three clock sources is connected: • Internal reference clock • External reference clock • Frequency-locked loop (FLL) output This clock drives the CPU, debug, RAM, and BDM directly and is divided by 2 to clock all peripherals (BUSCLK). See Chapter 11, "Internal Clock Source (ICS)," for details on configuring the ICSOUT clock.
ICSLCLK	This clock source runs at 1/2 the rate of the low frequency DCO (10/20 MHz digitally controlled oscillator). Development tools can select this internal self-clocked source to speed up BDC communications in systems where the bus clock is slow. Please see Figure 11-6, Figure 26-2, and their accompanying text for details.
ICSERCLK	ICS external reference clock — This is the external reference clock and can be selected as an alternate clock for the ADCs.
ICSIRCLK	ICS internal reference clock — This is the internal reference clock and it is not used outside of the ICS module.
ICSFFCLK	ICS fixed-frequency clock — This generates the fixed frequency clock (FFCLK) after being synchronized to the bus clock. The FFCLK can be selected as clock source for the MTIM and TPM modules. The frequency of the FFCLK is determined by the settings of the ICS.
LPOCLK	Low-power oscillator clock — This clock is generated from an internal low-power oscillator that is completely independent of the ICS module. The LPOCLK can be selected as the clock source to the COP.
TMRCLK1 & TMRCLK2	Optional external clock sources for the MTIMs. These clocks must be limited to one-quarter the frequency of the bus clock for synchronization. The SIMIPS register (see Section 7.7.14, "SIM Clock Options Register (SIMCO)") determines which clock source supplies the TCLK input to the MTIMs.
MTIM1 Output (MTCK)	The MTIM1 output can be used in place of TMRCLK1, TMRCLK2 and TPMCLK as the external clock source into the MTIM2, MTIM3 and TPM (see Section 7.7.14, "SIM Clock Options Register (SIMCO)"). This allows the user to use MTIM1 as a non-power-of-two prescaler of the bus clock.
TPMCLK	Optional external clock source for the TPM module (see Section 7.7.14, "SIM Clock Options Register (SIMCO)"). This clock must be limited to one-quarter the frequency of the bus clock for synchronization.
ADACK	Each ADC module also has an internally generated asynchronous clock which allows it to run in STOP mode (ADACK). This signal is not available externally.
CLKOUT	This is an optional output of the device which can be used to deliver any of a number of the on-chip clocks, including the crystal oscillator outputs, bus clock, etc. See Section 7.7.14, "SIM Clock Options Register (SIMCO)," for details.

1.3.4 Clock Gating

In order to save power, peripheral clocks can be shut off by programming the system clock gating registers. For details, refer to Section 7.6, “Peripheral Clock Gating.”

1.3.5 ICS Modes of Operation

The ICS operates in one of the modes described in Table 1-7. This information has been abbreviated for clarity’s sake. See the Chapter 11, “Internal Clock Source (ICS),” for additional details.

Table 1-7. ICS Clock Modes

Mode	Description
FLL Engaged Internal (FEI)	Default. ICSOUT is derived from the FLL clock, which is controlled by the internal reference clock. The FLL clock frequency locks to a multiple of the internal reference frequency. ICSLCLK is derived from the FLL.
FLL Engaged External (FEE)	ICSOUT is derived from the FLL clock, which is controlled by the external reference clock. The external reference clock that is enabled can be produced by an external crystal, ceramic resonator, or another external clock source connected to the required crystal oscillator (XOSC).
FLL Bypassed Internal (FBI)	ICSOUT is derived from the internal reference clock; the FLL is operational, but its output clock is not used. This mode is useful to allow the FLL to acquire its target frequency while the ICSOUT clock is driven from the internal reference clock.
FLL Bypassed External (FBE)	ICSOUT is derived from the external reference clock; the FLL is operational, but its output clock is not used. This mode is useful to allow the FLL to acquire its target frequency while ICSOUT is driven from the external reference clock.
FLL Bypassed Internal Low Power (FBILP)	ICSOUT is derived from the internal reference clock. The FLLs are disabled, and ICSLCLK is not available for BDC communications. If the BDM becomes enabled, the mode switches to one of the bypassed internal modes.
FLL Bypassed External Low Power (FBELP)	ICSOUT is derived from the external reference clock. The external reference clock that is enabled can be produced by an external crystal, ceramic resonator, or another external clock source connected to the required crystal oscillator (XOSC). The FLLs are disabled, and ICSLCLK is not available for BDC communications. If the BDM becomes enabled, the mode switches to one of the bypassed external mode.
STOP	Entered whenever the MCU enters a stop state. The FLL is disabled, and all ICS clock signals are static except that ICSIRCLK can be active in stop mode under certain conditions.

1.3.6 ICS Mode State Diagram

Figure 1-4 shows the valid state transitions for the ICS. The arrows indicate the permitted mode transistions. See Chapter 11, “Internal Clock Source (ICS),” for additional details.

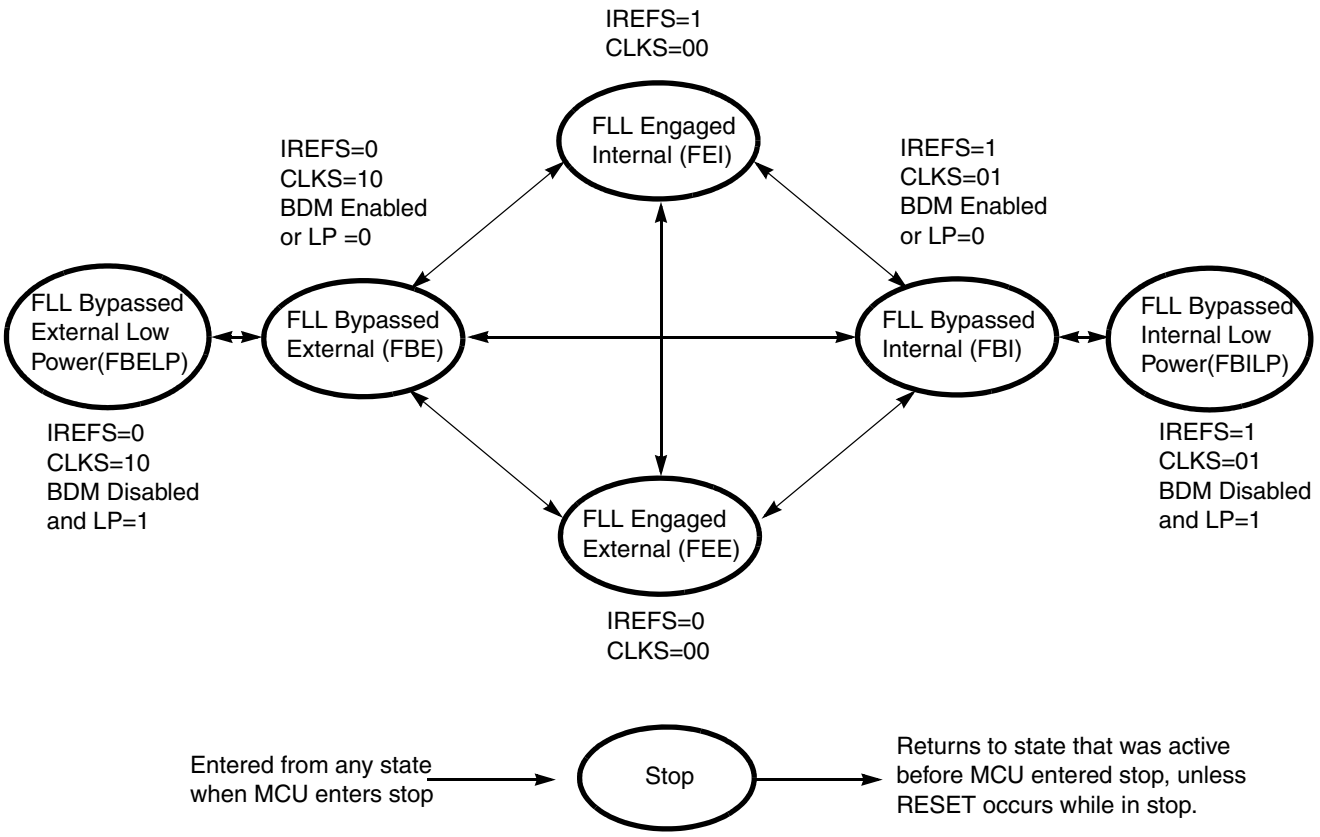


Figure 1-4. Clock Switching Modes

1.4 ADC Connections

The MCF51EM256 series include four separately controllable ADCs. Assignment of device ADC pins is spread over each of the available ADCs as shown in Table 21-1.

Chapter 2
Pins and Connections

This chapter describes signals that connect to package pins. It includes pinout diagrams, recommended system connections, and detailed discussions of signals.

2.1 Device Pin Assignment

2.1.1 Pinout: 80-Pin LQFP

Pins not available on the 80-pin LQFP are automatically disabled for reduced current consumption. No user interaction is needed. Software access to the functions on these pins will be ignored. Figure 2-1 shows the pinout of the 80-pin LQFP.

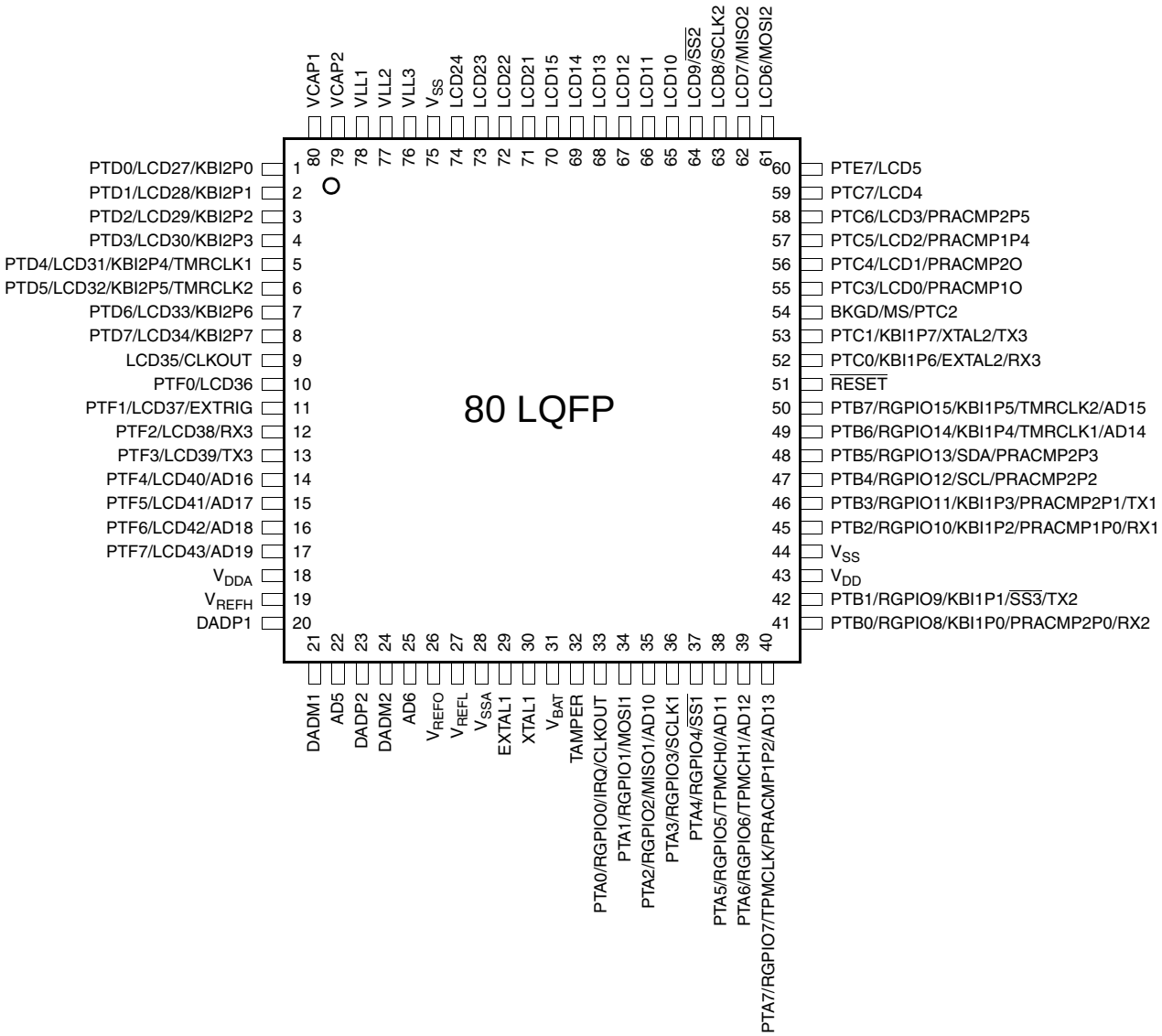


Figure 2-1. 80-Pin LQFP Pinout

2.1.2 Pinout: 100-Pin LQFP

Figure 2-2 shows the pinout configuration for the 100-pin LQFP. Pins which are blacked out do not have an equivalent pin on the 80-pin LQFP package.

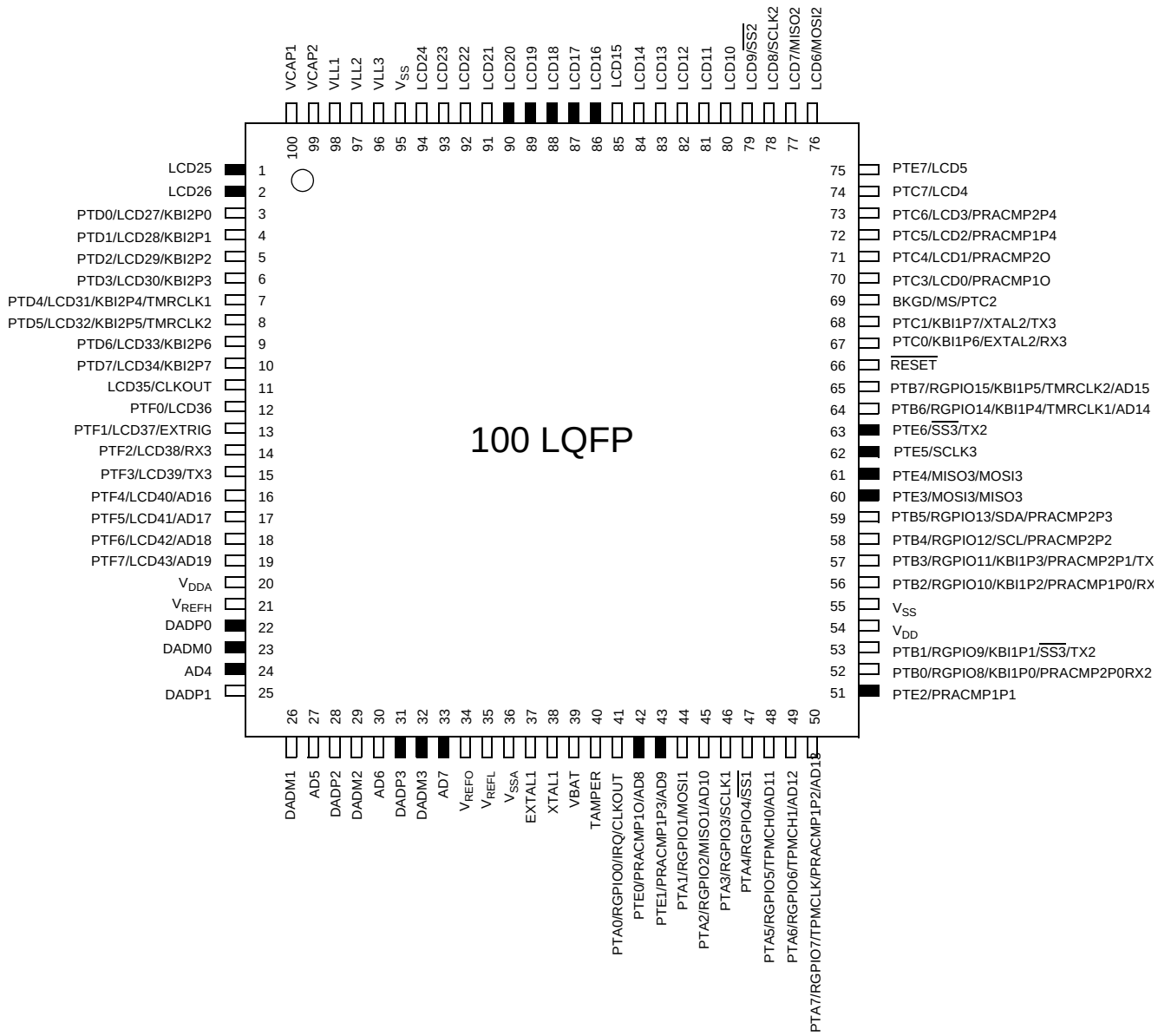


Figure 2-2. 100-Pin LQFP Pinout

2.1.3 Pin Assignments

The MCF51EM256 series will be available in 80-pin LQFP and 100-pin LQFP package options. Table 2-1 summaries the functions available on each pin of the various package configurations. The "Default Function" column specifies the function of the given pin upon exiting the reset state. Alternate functions 1, 2 and 3 can be assigned to each pin under software control via the MC[1] registers. The RGPIO_ENB register is used to select between GPIO and RGPIO functions on pins that have both.

Pins not available on smaller packages are automatically disabled for reduced current consumption. No user interaction is needed. Software access to the functions on these pins will be ignored

Table 2-1. MCF51EM256 Series Package Pin Assignments

100 LQFP	80 LQFP	Default Function	ALT1	ALT2	ALT3	Comment
1	—	LCD25				
2	—	LCD26				
3	1	PTD0	LCD27	KBI2P0		
4	2	PTD1	LCD28	KBI2P1		
5	3	PTD2	LCD29	KBI2P2		
6	4	PTD3	LCD30	KBI2P3		
7	5	PTD4	LCD31	KBI2P4	TMRCLK1	
8	6	PTD5	LCD32	KBI2P5	TMRCLK2	
9	7	PTD6	LCD33	KBI2P6		
10	8	PTD7	LCD34	KBI2P7		
11	9	LCD35	CLKOUT			
12	10	PTF0	LCD36			
13	11	PTF1	LCD37		EXTRIG	
14	12	PTF2	LCD38		RX3	
15	13	PTF3	LCD39		TX3	
16	14	PTF4	LCD40		AD16	
17	15	PTF5	LCD41		AD17	
18	16	PTF6	LCD42		AD18	
19	17	PTF7	LCD43		AD19	
20	18	V_{DDA}				
21	19	V_{REFH}				
22	—	DADP0				
23	—	DADM0				
24	—	AD4				

1. Port Mux Control registers.

Table 2-1. MCF51EM256 Series Package Pin Assignments (continued)

100 LQFP	80 LQFP	Default Function	ALT1	ALT2	ALT3	Comment
25	20	DADP1				
26	21	DADM1				
27	22	AD5				
28	23	DADP2				
29	24	DADM2				
30	25	AD6				
31	—	DADP3				
32	—	DADM3				
33	—	AD7				
34	26	V_{REFO}				
35	27	V_{REFL}				
36	28	V_{SSA}				
37	29	EXTAL1				
38	30	XTAL1				
39	31	V_{BAT}				
40	32	TAMPER				
41	33	PTA0/RGPIO0	IRQ	CLKOUT		
42	—	PTE0		PRACMP1O	AD8	
43	—	PTE1		PRACMP1P3	AD9	
44	34	PTA1/RGPIO1	MOSI1			RGPIO_ENB is used to select between standard GPIO and RGPIO
45	35	PTA2/RGPIO2	MISO1		AD10	
46	36	PTA3/RGPIO3	SCLK1			
47	37	PTA4/RGPIO4	$\overline{SS1}$			
48	38	PTA5/RGPIO5	TPMCH0		AD11	
49	39	PTA6/RGPIO6	TPMCH1		AD12	
50	40	PTA7/RGPIO7	TPMCLK	PRACMP1P2	AD13	
51	—	PTE2		PRACMP1P1		
52	41	PTB0/RGPIO8	KBI1P0	PRACMP2P0	RX2	RGPIO_ENB is used to select between standard GPIO and RGPIO
53	42	PTB1/RGPIO9	KBI1P1	$\overline{SS3}$	TX2	2X Drive Output RGPIO_ENB is used to select between standard GPIO and RGPIO

Table 2-1. MCF51EM256 Series Package Pin Assignments (continued)

100 LQFP	80 LQFP	Default Function	ALT1	ALT2	ALT3	Comment
54	43	V_{DD}				
55	44	V_{SS}				
56	45	PTB2/RGPIO10	KBI1P2	PRACMP1P0	RX1	RGPIO_ENB is used to select between standard GPIO and RGPIO
57	46	PTB3/RGPIO11	KBI1P3	PRACMP2P1	TX1	2X drive output RGPIO_ENB is used to select between standard GPIO and RGPIO
58	47	PTB4/RGPIO12	SCL	PRACMP2P2		RGPIO_ENB is used to select between standard GPIO and RGPIO
59	48	PTB5/RGPIO13	SDA	PRACMP2P3		
60	—	PTE3	MOSI3	MISO3		
61	—	PTE4	MISO3	MOSI3		Open Drain
62	—	PTE5	SCLK3			Open Drain
63	—	PTE6	$\overline{SS3}$	TX2		Open Drain
64	49	PTB6/RGPIO14	KBI1P4	TMRCLK1	AD14	RGPIO_ENB is used to select between standard GPIO and RGPIO
65	50	PTB7/RGPIO15	KBI1P5	TMRCLK2	AD15	
66	51	$\overline{RESET}$				This pin is an open drain device and has an internal pullup. There is no clamp diode to V_{DD}.
67	52	PTC0	KBI1P6	EXTAL2	RX3	
68	53	PTC1	KBI1P7	XTAL2	TX3	
69	54	BKGD/MS	PTC2			This pin has an internal pullup. PTC2 can only be programmed as an output.
70[1]	55[1]	PTC3	LCD0	PRACMP1O		
71[1]	56[1]	PTC4	LCD1	PRACMP2O		
72[1]	57[1]	PTC5	LCD2		PRACMP1P4	
73[1]	58[1]	PTC6	LCD3		PRACMP2P4	
74[1]	59[1]	PTC7	LCD4			
75[1]	60[1]	PTE7	LCD5			
76[1]	61[1]	LCD6	MOSI2			
77[1]	62[1]	LCD7	MISO2			
78[1]	63[1]	LCD8	SCLK2			
79[1]	64[1]	LCD9	$\overline{SS2}$			
80	65	LCD10				

Table 2-1. MCF51EM256 Series Package Pin Assignments (continued)

100 LQFP	80 LQFP	Default Function	ALT1	ALT2	ALT3	Comment
81	66	LCD11				
82	67	LCD12				
83	68	LCD13				
84	69	LCD14				
85	70	LCD15				
86	—	LCD16				
87	—	LCD17				
88	—	LCD18				
89	—	LCD19				
90	—	LCD20				
91	71	LCD21				
92	72	LCD22				
93	73	LCD23				
94	74	LCD24				
95	75	V_{SS}				
96	76	VLL3				
97	77	VLL2				
98	78	VLL1				
99	79	VCAP2				
100	80	VCAP1				

[1] These pins that are shared with the LCD are open-drain by default if not used as LCD pins. To configure this pins as full complementary drive outputs, you must have the LCD modules bits configured as follow: FCDEN =1, VSUPPLY = 11 and RVEN = 0. The Input levels and internal pullup resistors are referenced to VLL3. Referer to the LCD chapter for further information.

NOTE

The reset state for all LCD pins is open drain operation. Do not share any function on LCD pins that is not compatible with this reset state.

Drive strength, pullup enable[1], slew rate and input filter settings in the GPIO apply to any digital use of the associated pin.

The following tables break out pin options on a peripheral by peripheral basis.

1. There is one special case with regard to pullup enable functions. PTA0 can be programmed to operate as IRQ. When in that mode, the pullup enable is controlled via IRQSC[IRQPDD].

Table 2-2. RGPIO Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	RGPIO
41	33	PTA0/RGPIO0	PTA0/RGPIO0/IRQ/CLKOUT	RGPIO0
44	34	PTA1/RGPIO1	PTA1/RGPIO1/MOSI1	RGPIO1
56	45	PTB2/RGPIO10	PTB2/RGPIO10/KBI1P2/PRACMP1P0/RX1	RGPIO10
57	46	PTB3/RGPIO11	PTB3/RGPIO11/KBI1P3/PRACMP2P1/TX1	RGPIO11
58	47	PTB4/RGPIO12	PTB4/RGPIO12/SCL/PRACMP2P2	RGPIO12
59	48	PTB5/RGPIO13	PTB5/RGPIO13/SDA/PRACMP2P3	RGPIO13
64	49	PTB6/RGPIO14	PTB6/RGPIO14/KBI1P4/TMRCLK1/AD14	RGPIO14
65	50	PTB7/RGPIO15	PTB7/RGPIO15/KBI1P5/TMRCLK2/AD15	RGPIO15
45	35	PTA2/RGPIO2	PTA2/RGPIO2/MISO1/AD10	RGPIO2
46	36	PTA3/RGPIO3	PTA3/RGPIO3/SCLK1	RGPIO3
47	37	PTA4/RGPIO4	PTA4/RGPIO4/$\overline{SS1}$	RGPIO4
48	38	PTA5/RGPIO5	PTA5/RGPIO5/TPMCH0/AD11	RGPIO5
49	39	PTA6/RGPIO6	PTA6/RGPIO6/TPMCH1/AD12	RGPIO6
50	40	PTA7/RGPIO7	PTA7/RGPIO7/TPMCLK/PRACMP1P2/AD13	RGPIO7
52	41	PTB0/RGPIO8	PTB0/RGPIO8/KBI1P0/PRACMP2P0/RX2	RGPIO8
53	42	PTB1/RGPIO9	PTB1/RGPIO9/KBI1P1/$\overline{SS3}$/TX2	RGPIO9

Table 2-3. PTA Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	PTA
41	33	PTA0/RGPIO0	PTA0/RGPIO0/IRQ/CLKOUT	PTA0
44	34	PTA1/RGPIO1	PTA1/RGPIO1/MOSI1	PTA1
45	35	PTA2/RGPIO2	PTA2/RGPIO2/MISO1/AD10	PTA2
46	36	PTA3/RGPIO3	PTA3/RGPIO3/SCLK1	PTA3
47	37	PTA4/RGPIO4	PTA4/RGPIO4/$\overline{SS1}$	PTA4
48	38	PTA5/RGPIO5	PTA5/RGPIO5/TPMCH0/AD11	PTA5
49	39	PTA6/RGPIO6	PTA6/RGPIO6/TPMCH1/AD12	PTA6
50	40	PTA7/RGPIO7	PTA7/RGPIO7/TPMCLK/PRACMP1P2/AD13	PTA7

Table 2-4. PTB Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	PTB	Comment
52	41	PTB0/RGPIO8	PTB0/RGPIO8/KBI1P0/PRACMP2P0/RX2	PTB0	
53	42	PTB1/RGPIO9	PTB1/RGPIO9/KBI1P1/$\overline{SS3}$/TX2	PTB1	2X Drive
56	45	PTB2/RGPIO10	PTB2/RGPIO10/KBI1P2/PRACMP1P0/RX1	PTB2	
57	46	PTB3/RGPIO11	PTB3/RGPIO11/KBI1P3/PRACMP2P1/TX1	PTB3	2X Drive
58	47	PTB4/RGPIO12	PTB4/RGPIO12/SCL/PRACMP2P2	PTB4	
59	48	PTB5/RGPIO13	PTB5/RGPIO13/SDA/PRACMP2P3	PTB5	
64	49	PTB6/RGPIO14	PTB6/RGPIO14/KBI1P4/TMRCLK1/AD14	PTB6	
65	50	PTB7/RGPIO15	PTB7/RGPIO15/KBI1P5/TMRCLK2/AD15	PTB7	

Table 2-5. PTC Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	PTC
67	52	PTC0	PTC0/KBI1P6/EXTAL2/RX3	PTC0
68	53	PTC1	PTC1/KBI1P7/XTAL2/TX3	PTC1
69	54	BKGD/MS	BKGD/MS/PTC2	PTC2
70	55	PTC3	PTC3/LCD0/PRACMP1O	PTC3
71	56	PTC4	PTC4/LCD1/PRACMP2O	PTC4
72	57	PTC5	PTC5/LCD2/PRACMP1P4	PTC5
73	58	PTC6	PTC6/LCD3/PRACMP2P4	PTC6
74	59	PTC7	PTC7/LCD4	PTC7

Table 2-6. PTD Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	PTD
3	1	PTD0	PTD0/LCD27/KBI2P0	PTD0
4	2	PTD1	PTD1/LCD28/KBI2P1	PTD1
5	3	PTD2	PTD2/LCD29/KBI2P2	PTD2
6	4	PTD3	PTD3/LCD30/KBI2P3	PTD3
7	5	PTD4	PTD4/LCD31/KBI2P4/TMRCLK1	PTD4
8	6	PTD5	PTD5/LCD32/KBI2P5/TMRCLK2	PTD5
9	7	PTD6	PTD6/LCD33/KBI2P6	PTD6
10	8	PTD7	PTD7/LCD34/KBI2P7	PTD7

Table 2-7. PTE Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	PTE	Comment
42		PTE0	PTE0/PRACMP1O/AD8	PTE0	
43		PTE1	PTE1/PRACMP1P3/AD9	PTE1	
51		PTE2	PTE2/PRACMP1P1	PTE2	
60		PTE3	PTE3/MOSI3/MISO3	PTE3	
61		PTE4	PTE4/MISO3/MOSI3	PTE4	Open Drain
62		PTE5	PTE5/SCLK3	PTE5	Open Drain
63		PTE6	PTE6/$\overline{SS3}$/TX2	PTE6	Open Drain
75	60	PTE7	PTE7/LCD5	PTE7	

Table 2-8. PTF Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	PTF
12	10	PTF0	PTF0/LCD36	PTF0
13	11	PTF1	PTF1/LCD37/EXTRIG	PTF1
14	12	PTF2	PTF2/LCD38/RX3	PTF2
15	13	PTF3	PTF3/LCD39/TX3	PTF3
16	14	PTF4	PTF4/LCD40/AD16	PTF4
17	15	PTF5	PTF5/LCD41/AD17	PTF5
18	16	PTF6	PTF6/LCD42/AD18	PTF6
19	17	PTF7	PTF7/LCD43/AD19	PTF7

Table 2-9. KBI1 Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	KBI1
52	41	PTB0/RGPIO8	PTB0/RGPIO8/KBI1P0/PRACMP2P0/RX2	KBI1P0
53	42	PTB1/RGPIO9	PTB1/RGPIO9/KBI1P1/$\overline{SS3}$/TX2	KBI1P1
56	45	PTB2/RGPIO10	PTB2/RGPIO10/KBI1P2/PRACMP1P0/RX1	KBI1P2
57	46	PTB3/RGPIO11	PTB3/RGPIO11/KBI1P3/PRACMP2P1/TX1	KBI1P3
64	49	PTB6/RGPIO14	PTB6/RGPIO14/KBI1P4/TMRCLK1/AD14	KBI1P4
65	50	PTB7/RGPIO15	PTB7/RGPIO15/KBI1P5/TMRCLK2/AD15	KBI1P5
67	52	PTC0	PTC0/KBI1P6/EXTAL2/RX3	KBI1P6
68	53	PTC1	PTC1/KBI1P7/XTAL2/TX3	KBI1P7

Table 2-10. KBI2 Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	KBI2
3	1	PTD0	PTD0/LCD27/KBI2P0	KBI2P0
4	2	PTD1	PTD1/LCD28/KBI2P1	KBI2P1
5	3	PTD2	PTD2/LCD29/KBI2P2	KBI2P2
6	4	PTD3	PTD3/LCD30/KBI2P3	KBI2P3
7	5	PTD4	PTD4/LCD31/KBI2P4/TMRCLK1	KBI2P4
8	6	PTD5	PTD5/LCD32/KBI2P5/TMRCLK2	KBI2P5
9	7	PTD6	PTD6/LCD33/KBI2P6	KBI2P6
10	8	PTD7	PTD7/LCD34/KBI2P7	KBI2P7

Table 2-11. SPI1 Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	SPI1
45	35	PTA2/RGPIO2	PTA2/RGPIO2/MISO1/AD10	MISO1
44	34	PTA1/RGPIO1	PTA1/RGPIO1/MOSI1	MOSI1
46	36	PTA3/RGPIO3	PTA3/RGPIO3/SCLK1	SCLK1
47	37	PTA4/RGPIO4	PTA4/RGPIO4/$\overline{SS1}$	$\overline{SS1}$

Table 2-12. SPI2 Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	SPI2
77	62	LCD7	LCD7/MISO2	MISO2
76	61	LCD6	LCD6/MOSI2	MOSI2
78	63	LCD8	LCD8/SCLK2	SCLK2
79	64	LCD9	LCD9/$\overline{SS2}$	$\overline{SS2}$

Table 2-13. SPI3 Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	SPI3	Comment
61		PTE4	PTE4/MISO3/MOSI3	MISO3/MOSI3	Open Drain
60		PTE3	PTE3/MOSI3/MISO3	MOSI3/MISO3	
62		PTE5	PTE5/SCLK3	SCLK3	Open Drain
53	42	PTB1/RGPIO9	PTB1/RGPIO9/KBI1P1/$\overline{SS3}$/TX2	$\overline{SS3}$	
63		PTE6	PTE6/$\overline{SS3}$/TX2	$\overline{SS3}$	Open Drain

Table 2-14. LCD Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	LCD
70	55	PTC3	PTC3/LCD0/PRACMP1O	LCD0
71	56	PTC4	PTC4/LCD1/PRACMP2O	LCD1
72	57	PTC5	PTC5/LCD2/PRACMP1P4	LCD2
73	58	PTC6	PTC6/LCD3/PRACMP2P4	LCD3
74	59	PTC7	PTC7/LCD4	LCD4
75	60	PTE7	PTE7/LCD5	LCD5
76	61	LCD6	LCD6/MOSI2	LCD6
77	62	LCD7	LCD7/MISO2	LCD7
78	63	LCD8	LCD8/SCLK2	LCD8

Table 2-14. LCD Pinout Summary (continued)

100 LQFP	80 LQFP	Default Function	Composite Pin Name	LCD
79	64	LCD9	LCD9/$\overline{SS2}$	LCD9
80	65	LCD10	LCD10	LCD10
81	66	LCD11	LCD11	LCD11
82	67	LCD12	LCD12	LCD12
83	68	LCD13	LCD13	LCD13
84	69	LCD14	LCD14	LCD14
85	70	LCD15	LCD15	LCD15
86		LCD16	LCD16	LCD16
87		LCD17	LCD17	LCD17
88		LCD18	LCD18	LCD18
89		LCD19	LCD19	LCD19
90		LCD20	LCD20	LCD20
91	71	LCD21	LCD21	LCD21
92	72	LCD22	LCD22	LCD22
93	73	LCD23	LCD23	LCD23
94	74	LCD24	LCD24	LCD24
1		LCD25	LCD25	LCD25
2		LCD26	LCD26	LCD26
3	1	PTD0	PTD0/LCD27/KBI2P0	LCD27
4	2	PTD1	PTD1/LCD28/KBI2P1	LCD28
5	3	PTD2	PTD2/LCD29/KBI2P2	LCD29
6	4	PTD3	PTD3/LCD30/KBI2P3	LCD30
7	5	PTD4	PTD4/LCD31/KBI2P4/TMRCLK1	LCD31
8	6	PTD5	PTD5/LCD32/KBI2P5/TMRCLK2	LCD32
9	7	PTD6	PTD6/LCD33/KBI2P6	LCD33
10	8	PTD7	PTD7/LCD34/KBI2P7	LCD34
11	9	LCD35	LCD35/CLKOUT	LCD35
12	10	PTF0	PTF0/LCD36	LCD36
13	11	PTF1	PTF1/LCD37/EXTRIG	LCD37
14	12	PTF2	PTF2/LCD38/RX3	LCD38
15	13	PTF3	PTF3/LCD39/TX3	LCD39
16	14	PTF4	PTF4/LCD40/AD16	LCD40

Table 2-14. LCD Pinout Summary (continued)

100 LQFP	80 LQFP	Default Function	Composite Pin Name	LCD
17	15	PTF5	PTF5/LCD41/AD17	LCD41
18	16	PTF6	PTF6/LCD42/AD18	LCD42
19	17	PTF7	PTF7/LCD43/AD19	LCD43
100	80	VCAP1	VCAP1	VCAP1
99	79	VCAP2	VCAP2	VCAP2
98	78	VLL1	VLL1	VLL1
97	77	VLL2	VLL2	VLL2
96	76	VLL3	VLL3	VLL3

Table 2-15. IIC Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	IIC
58	47	PTB4/RGPIO12	PTB4/RGPIO12/SCL/PRACMP2P2	SCL
59	48	PTB5/RGPIO13	PTB5/RGPIO13/SDA/PRACMP2P3	SDA

Table 2-16. Other Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	Other	Module
69	54	BKGD/MS	BKGD/MS/PTC2	PTC2	Debug pin
11	9	LCD35	LCD35/CLKOUT	CLKO	Clock output
66	51	$\overline{\text{RESET}}$	$\overline{\text{RESET}}$	$\overline{\text{RESET}}$	
13	11	PTF1	PTF1/LCD37/EXTRIG	EXTRIG	PDB
41	33	PTA0/RGPIO0	PTA0/RGPIO0/IRQ/CLKO	IRQ/CLKO	IRQ module
7	5	PTD4	PTD4/LCD31/KBI2P4/TMRCLK1	TMRCLK1	Timer input
64	49	PTB6/RGPIO14	PTB6/RGPIO14/KBI1P4/TMRCLK1/AD14	TMRCLK1	
8	6	PTD5	PTD5/LCD32/KBI2P5/TMRCLK2	TMRCLK2	
65	50	PTB7/RGPIO15	PTB7/RGPIO15/KBI1P5/TMRCLK2/AD15	TMRCLK2	
34	26	V_{REFO}	V_{REFO}	V_{REFO}	Voltage reference

Table 2-17. P/G Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	P/G
54	43	V_{DD}	V_{DD}	V_{DD}
20	18	V_{DDA}	V_{DDA}	V_{DDA}
55	44	V_{SS}	V_{SS}	V_{SS}
95	75	V_{SS}	V_{SS}	V_{SS}
36	28	V_{SSA}	V_{SSA}	V_{SSA}
39	31	V_{BAT}	V_{BAT}	V_{BAT}

Table 2-18. SCI1 Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	SCI1	Comment
56	45	PTB2/RGPIO10	PTB2/RGPIO10/KBI1P2/PRACMP1P0/RX1	RX1	
57	46	PTB3/RGPIO11	PTB3/RGPIO11/KBI1P3/PRACMP2P1/TX1	TX1	2X Drive

Table 2-19. SCI2 Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	SCI2	Comment
52	41	PTB0/RGPIO8	PTB0/RGPIO8/KBI1P0/PRACMP2P0/RX2	RX2	
53	42	PTB1/RGPIO9	PTB1/RGPIO9/KBI1P1/$\overline{\text{SS3}}$/TX2	TX2	2X Drive
63		PTE6	PTE6/$\overline{\text{SS3}}$/TX2	TX2	Open Drain

Table 2-20. SCI3 Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	SCI3
14	12	PTF2	PTF2/LCD38/RX3	RX3
15	13	PTF3	PTF3/LCD39/TX3	TX3
67	52	PTC0	PTC0/KBI1P6/EXTAL2/RX3	RX3
68	53	PTC1	PTC1/KBI1P7/XTAL2/TX3	TX3

Table 2-21. PRACMP1 Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	PRACMP1
42		PTE0	PTE0/PRACMP1O/AD8	PRACMP1O
70	55	PTC3	PTC3/LCD0/PRACMP1O	PRACMP1O
52	41	PTB0/RGPIO8	PTB0/RGPIO8/KBI1P0/PRACMP2P0/RX2	PRACMP1P0
51		PTE2	PTE2/PRACMP1P1	PRACMP1P1
50	40	PTA7/RGPIO7	PTA7/RGPIO7/TPMCLK/PRACMP1P2/AD13	PRACMP1P2
43		PTE1	PTE1/PRACMP1P3/AD9	PRACMP1P3
72	57	PTC5	PTC5/LCD2/PRACMP1P4	PRACMP1P4

Table 2-22. PRACMP2 Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	PRACMP2
71	56	PTC4	PTC4/LCD1/PRACMP2O	PRACMP2O
56	45	PTB2/RGPIO10	PTB2/RGPIO10/KBI1P2/PRACMP1P0/RX1	PRACMP2P0
57	46	PTB3/RGPIO11	PTB3/RGPIO11/KBI1P3/PRACMP2P1/TX1	PRACMP2P1
58	47	PTB4/RGPIO12	PTB4/RGPIO12/SCL/PRACMP2P2	PRACMP2P2
59	48	PTB5/RGPIO13	PTB5/RGPIO13/SDA/PRACMP2P3	PRACMP2P3
73	58	PTC6	PTC6/LCD3/PRACMP2P4	PRACMP2P4

Table 2-23. ADC Pinout Summary[1]

100 LQFP	80 LQFP	Default Function	Composite Pin Name	ADC
23		DADM0	DADM0	DADM0
26	21	DADM1	DADM1	DADM1
29	24	DADM2	DADM2	DADM2
32		DADM3	DADM3	DADM3
22		DADP0	DADP0	DADP0

Table 2-23. ADC Pinout Summary[1] (continued)

100 LQFP	80 LQFP	Default Function	Composite Pin Name	ADC
25	20	DADP1	DADP1	DADP1
28	23	DADP2	DADP2	DADP2
31		DADP3	DADP3	DADP3
21	19	V_{REFH}	V_{REFH}	VREFH
35	27	V_{REFL}	V_{REFL}	VREFL
24		AD4	AD4	AD4
27	22	AD5	AD5	AD5
30	25	AD6	AD6	AD6
33		AD7	AD7	AD7
42		PTE0	PTE0/PRACMP1O/AD8	AD8
43		PTE1	PTE1/PRACMP1P3/AD9	AD9
45	35	PTA2/RGPIO2	PTA2/RGPIO2/MISO1/AD10	AD10
48	38	PTA5/RGPIO5	PTA5/RGPIO5/TPMCH0/AD11	AD11
49	39	PTA6/RGPIO6	PTA6/RGPIO6/TPMCH1/AD12	AD12
50	40	PTA7/RGPIO7	PTA7/RGPIO7/TPMCLK/PRACMP1P2/AD13	AD13
64	49	PTB6/RGPIO14	PTB6/RGPIO14/KBI1P4/TMRCLK1/AD14	AD14
65	50	PTB7/RGPIO15	PTB7/RGPIO15/KBI1P5/TMRCLK2/AD15	AD15
16	14	PTF4	PTF4/LCD40/AD16	AD16
17	15	PTF5	PTF5/LCD41/AD17	AD17
18	16	PTF6	PTF6/LCD42/AD18	AD18
19	17	PTF7	PTF7/LCD43/AD19	AD19

[1] Refer to Table 21-1 "ADC Channel Assignments" for details of the ADC channel assignment.

Table 2-24. TPM Pinout Summary

100 LQFP	80 LQFP	Default Function	Composite Pin Name	TPM
48	38	PTA5/RGPIO5	PTA5/RGPIO5/TPMCH0/AD11	TPMCH0
49	39	PTA6/RGPIO6	PTA6/RGPIO6/TPMCH1/AD12	TPMCH1
50	40	PTA7/RGPIO7	PTA7/RGPIO7/TPMCLK/PRACMP1P2/AD13	TPMCLK

Table 2-25. IRTC Pinout Summary[1]

100 LQFP	80 LQFP	Default Function	Composite Pin Name
37	29	EXTAL1	EXTAL1
38	30	XTAL1	XTAL1
39	31	VBAT	VBAT
40	32	TAMPER	TAMPER
67	52	PTC0	PTC0/KBI1P6/EXTAL2/RX3
68	53	PTC1	PTC1KBI1P7/XTAL2/TX3

1 These pins are in IRTC power domain.

2.2 Basic System Connections

Figure 2-3 shows pin connections that are common to MCF51EM256 series application systems.

The following notes apply to Figure 2-3:

1. $\overline{\text{RESET}}$ pin can only be used to reset into user mode, you can not enter BDM using $\overline{\text{RESET}}$ pin. BDM can be entered by holding MS low during POR or writing a 1 to BDM_RESET in the ColdFire XCSR register with MS low after issuing BDM command.
2. RC filter on $\overline{\text{RESET}}$ pin recommended for noisy environments.

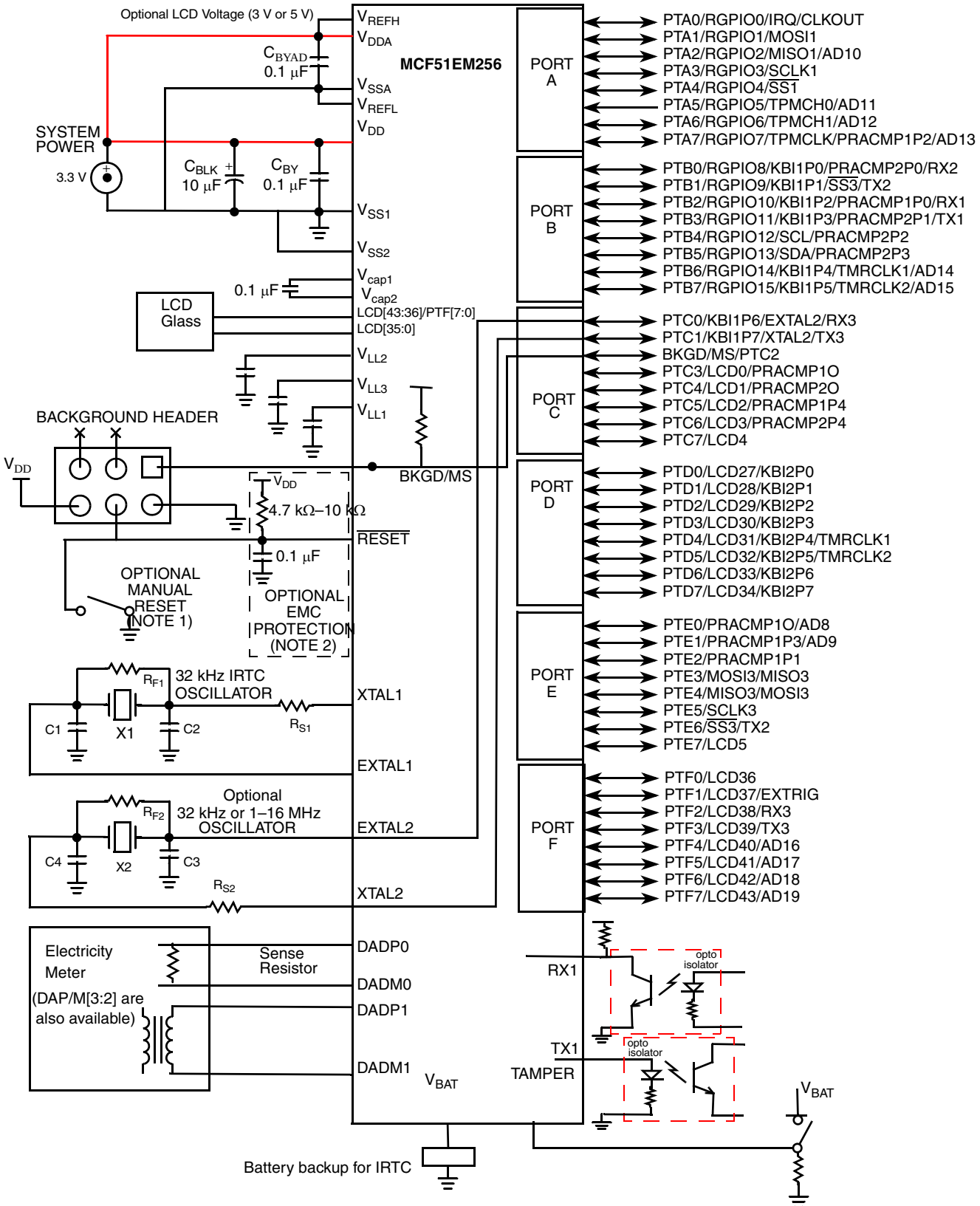
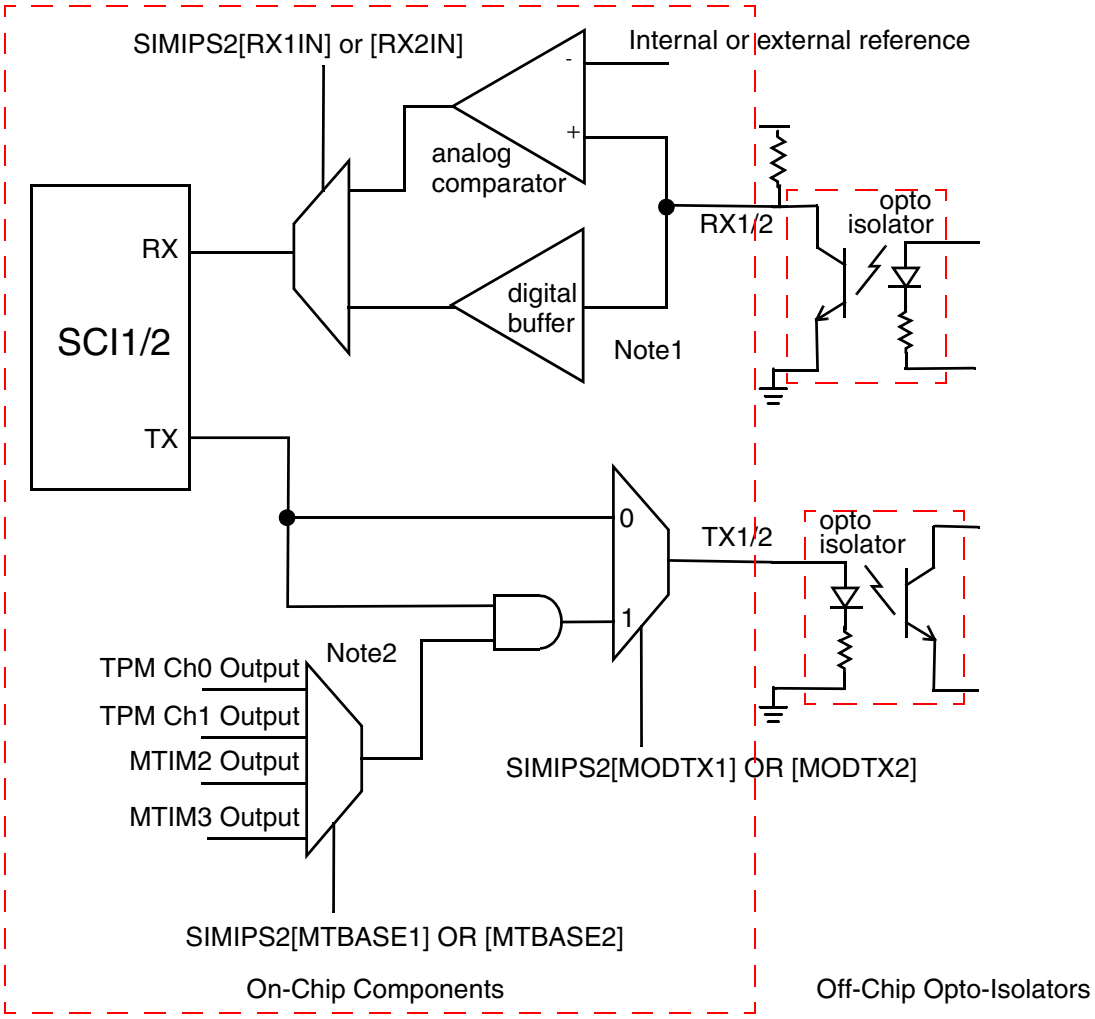


Figure 2-3. Basic System Connections

2.2.1 Interfacing the SCIs to Off-Chip Circuits

SCI1 and SCI2 are designed with twice the normal I/O drive capability on the TX pins. The RX pins can either be fed directly from the digital I/O buffer, or those signals can be pre-conditioned using comparators 1 and 2 as shown in Figure 2-4. Similarly, the TX outputs can be modulated with the output of one of the timers before being passed off chip.



1 This is the default SCI Rx digital input.

2 The TPM channels have to be configured to Toggle on output compare. The TPM channel pins can be used as regular GPIOs. Please see configurations in Section 4.7, "Pin Mux Controls."

Figure 2-4. On-Chip Signal Conditioning Associated with SCI RX and TX Pins

Controls for the circuitry shown in Figure 2-4 are discussed in Section 7.7.16, "Internal Peripheral Select Register 2 (SIMIPS2)."

2.2.2 Power

V_{DD}, V_{BAT} and $V_{SS1,}V_{SS2}$ are the primary power supply pins for the microcontroller. V_{DD}, V_{SS1} and V_{SS2} supplies power to all I/O buffer circuitry and to an internal voltage regulator; V_{BAT} only supply the power of IRTC and OSC1. The internal voltage regulator provides regulated lower-voltage source to the CPU and other internal circuitry of the microcontroller.

Typically, application systems have two separate capacitor values across the power pins. In this case, there must be a bulk electrolytic capacitor, such as a 10 μF tantalum capacitor, to provide bulk charge storage for the overall system and a 0.1 μF ceramic bypass capacitor located as close to the microcontroller power pins as practical to suppress high-frequency noise. Each MCF51EM256 series V_{DD} pin must have a 0.1 μF ceramic bypass capacitor for best noise suppression.

V_{DDA} and V_{SSA} are the analog power supply pins for the microcontroller. This voltage source supplies power to the ADC module. A 0.1 μF ceramic bypass capacitor should be located as close to the microcontroller power pins as practical to suppress high-frequency noise. V_{DDA} supplies PRACMP module as well.

2.2.3 Oscillator

Immediately after reset, the microcontroller uses an internally generated clock provided by the internal clock source (ICS) module.

The oscillator (XOSC) in this microcontroller is a Pierce oscillator that can accommodate a crystal or ceramic resonator. Optionally, an external clock source can be connected to the EXTAL input pin.

Refer to Figure 2-3 for the following discussion. R_S (when used) and R_F must be low-inductance resistors such as carbon composition resistors. Wire-wound resistors, and some metal film resistors, have too much inductance. C1 and C2 normally must be high-quality ceramic capacitors that are specifically designed for high-frequency applications.

R_F is used to provide a bias path to keep the EXTAL input in its linear range during crystal startup; its value is not generally critical. Typical systems use 1 MΩ to 10 MΩ. Higher values are sensitive to humidity and lower values reduce gain and (in extreme cases) could prevent startup.

C1 and C2 are typically in the 5 pF to 25 pF range and are chosen to match the requirements of a specific crystal or resonator. Be sure to take into account printed circuit board (PCB) capacitance and microcontroller pin capacitance when selecting C1 and C2. The crystal manufacturer typically specifies a load capacitance which is the series combination of C1 and C2 (which are usually the same size). As a first-order approximation, use 10 pF as an estimate of combined pin and PCB capacitance for each oscillator pin (EXTAL and XTAL).

2.2.4 $\overline{\text{RESET}}$ Pin

The $\overline{\text{RESET}}$ pin defaults to hardware reset upon a power-on-reset event. During stop2, the $\overline{\text{RESET}}$ pin can be used to wake the device from that state. There is a direct analog connection from this pad to the power management controller wakeup pin.

The internal pullup on this pin is enabled upon any device reset.

$\overline{\text{RESET}}$ is normally connected to the standard 6-pin background debug connector, so a development system can directly reset the MCU system. If desired, a manual external reset can be added by supplying a simple switch to ground (pull reset pin low to force a reset).

In EMC-sensitive applications, an external RC filter is recommended on this pin. See Figure 2-3 for an example.

2.2.5 IRQ

The IRQ pin function acts as a non-maskable interrupt to the V1 ColdFire core. This pin can be reprogrammed to function as PTA0.

2.2.6 Background / Mode Select (BKGD/MS)

During a power-on-reset (POR) or background debug force reset (see bit ENBDM in Section 26.3.2, "Extended Configuration/Status Register (XCSR)," for more information), the BKGD/MS pin functions as a mode select pin. Immediately after any reset, the pin functions as the background pin and can be used for background debug communication. The internal pullup on this pin is enabled upon any device reset.

If the BKGD/MS pin is unconnected, the microcontroller will enter normal operating mode at the rising edge of the internal reset after a POR or forced BDC reset. If a debug system is connected to the 6-pin standard background debug header, it can hold BKGD/MS low during a POR or immediately after issuing a background debug force reset[1], which forces the microcontroller to halt mode.

The BKGD/MS pin is used primarily for background debug controller (BDC) communications using a custom protocol that uses 16 clock cycles of the target microcontroller's BDC clock per bit time. The target microcontroller's BDC clock could be as fast as the bus clock rate, so there must never be any significant capacitance connected to the BKGD/MS pin that could interfere with background serial communications.

Although the BKGD/MS pin is a pseudo open-drain pin, the background debug communication protocol provides brief, actively driven, high speed-up pulses to ensure fast rise times. Small capacitances from cables and the absolute value of the internal pullup device play almost no role in determining rise and fall times on the BKGD/MS pin.

The BKGD/MS select pin can be reprogrammed to operate as PTC2 on this device. It must only be programmed for use as an output, as an external signal driving this pin during startup may cause the device to boot into debug mode.

1. Specifically, BKGD must be held low through the first 16 cycles after deassertion of the internal reset.

2.2.7 ADC Reference Pins (V_{REFH}, V_{REFL})

V_{REFH} and V_{REFL} are the voltage reference high and voltage reference low inputs, respectively, for the ADC modules. A 0.1 μF ceramic bypass capacitor must be located as near to the ADC reference pins as practical to suppress high-frequency noise.

2.2.8 General-Purpose I/O and Peripheral Ports

The MCF51EM256 series microcontrollers support up to 48 general-purpose I/O pins (including one output-only pin), which are shared with on-chip peripheral functions (timers, serial I/O, ADC, etc.).

When a port pin is configured as a general-purpose output or a peripheral uses the port pin as an output, software can select one of the two drive strengths and enable or disable slew rate control.

When a port pin is configured as a general-purpose input or a peripheral uses the port pin as an input, software can enable a pullup device. Pad cells on these devices also include an optional low pass filter in the inputs. Again, this can be enabled via software control.

Immediately after reset, all of these pins (excluding the BKGD/MS pin) are configured as high-impedance general-purpose inputs with internal pullup devices disabled.

When an on-chip peripheral system is controlling a pin, data direction control bits still determine what is read from the port data registers, even though the peripheral controls the pin direction via the pin's output buffer enable. For information about controlling these pins as general-purpose I/O pins, see "Chapter 4, "Parallel Input/Output Control."

Chapter 3
Memory

3.1 MCF51EM256 Series Memory Map

The left-most map in Figure 3-1 is the generic, high level, memory map applicable to the V1 ColdFire family. Memory map areas shown for RAM and flash are a superset for the family. Lesser amounts of both will usually be included on specific devices. The memory maps for the MCF51EM256 and MCF51EM128 are shown in the center and right, respectively.

Address Range	Generic V1 ColdFire Memory Usage
0x(00)00_0000	Allocated to on-chip flash memory
0x(00)3F_FFFF	
0x(00)40_0000	Optional off-chip expansion (not available on MCF51EM devices)
0x(00)7F_FFFF	
0x(00)80_0000	Allocated to on-chip RAM memory
0x(00)9F_FFFF	
0x(00)A0_0000	Optional off-chip expansion (not available on MCF51EM devices)
0x(00)BF_FFFF	
0x(00)C0_0000	ColdFire Rapid GPIO
0x(00)C0_000F	
0x(00)C0_0010	Unimplemented
0x(FF)FF_7FFF	
0x(FF)FF_8000	Slave Peripherals
0x(FF)FF_FFFF	

Address Range	MCF51EM256 Memory Usage
0x(00)00_0000	128 KB Flash Memory Array 1
0x(00)01_FFFF	
0x(00)02_0000	128 KB Flash Memory Array 2
0x(00)03_FFFF	
0x(00)04_0000	Unimplemented
0x(00)7F_FFFF	
0x(00)80_0000	16 KB RAM memory
0x(00)80_3FFF	
0x(00)80_4000	Unimplemented
0x(00)BF_FFFF	
0x(00)C0_0000	ColdFire Rapid GPIO
0x(00)C0_000F	
0x(00)C0_0010	Unimplemented
0x(FF)FF_7FFF	
0x(FF)FF_8000	Slave Peripherals
0x(FF)FF_FFFF	

Address Range	MCF51EM128 Memory Usage
0x(00)00_0000	64 KB Flash Memory Array 1
0x(00)00_FFFF	
0x(00)01_0000	64 KB Flash Memory Array 2
0x(00)01_FFFF	
0x(00)02_0000	Unimplemented
0x(00)7F_FFFF	
0x(00)80_0000	8 KB RAM memory
0x(00)80_1FFF	
0x(00)80_2000	Unimplemented
0x(00)BF_FFFF	
0x(00)C0_0000	ColdFire Rapid GPIO
0x(00)C0_000F	
0x(00)C0_0010	Unimplemented
0x(FF)FF_7FFF	
0x(FF)FF_8000	Slave Peripherals
0x(FF)FF_FFFF	

Figure 3-1. MCF51EM256 Series Memory Maps

Regions within the memory map are subject to restrictions with regard to the types of CPU accesses allowed. These are outlined in Table 3-1. Non-supported access types terminate the bus cycle with an error (and would typically generate a system reset in response to the error termination).

Table 3-1. CPU Access Type Allowed by Region

Base Address	Region	Read			Write		
		Byte	Word	Long	Byte	Word	Long
0x(00)00_0000	Flash	x	x	x	—	—	x
0x(00)80_0000	RAM	x	x	x	x	x	x
0x(00)C0_0000	Rapid GPIO	x	x	x	x	x	x
0x(FF)FF_8000	8-bit Peripherals[1]	x	x	x	x	x	x
0x(FF)FF_E000	16-bit Peripherals[2]	—	x	x	—	x	x
0x(FF)FF_8700	LCD registers[3]	x	x	x	x	—	—

1 Allowed access types are peripheral specific. The peripheral bus bridge will serialize 16 and 32-bit accesses into multiple 8-bit accesses. When using 8-bit peripherals, care must be taken to ensure that all accesses are properly aligned and only desired 8-bit locations are accessed.

2 Allowed access types are peripheral specific. The peripheral bus bridge will serialize 32-bit accesses into multiple 16-bit accesses. When using 16-bit peripherals, care must be taken to ensure that all accesses are properly aligned and only desired 16-bit locations are accessed.

3 LCD registers require several cycles between write accesses. Only byte writes should be used to write to these registers. Finally, consecutive writes must be separated by at least one non-write cycle.

Consistent with past ColdFire devices, flash configuration data is located at 0x(00)00_0400. The slave peripherals section of the memory map is further broken down as shown in Table 3-2.

Table 3-2. High Level Peripheral Memory Map

Peripheral	Description	Instance Name	Base Address
RGPIO	Rapid General Purpose I/O	RGPIO	0x(00)C0_0000
Port I/O Module	General Purpose I/O With Set/Clear/Toggle Functionality	PTA	0x(FF)FF_8000
Port Control Module	Port I/O Control Module	PTA	0x(FF)FF_8008
Port I/O Module	General Purpose I/O With Set/Clear/Toggle Functionality	PTB	0x(FF)FF_8040
Port Control Module	Port I/O Control Module	PTB	0x(FF)FF_8048
Port I/O Module	General Purpose I/O With Set/Clear/Toggle Functionality	PTC	0x(FF)FF_8080
Port Control Module	Port I/O Control Module	PTC	0x(FF)FF_8088
Port I/O Module	General Purpose I/O With Set/Clear/Toggle Functionality	PTD	0x(FF)FF_80C0
Port Control Module	Port I/O Control Module	PTD	0x(FF)FF_80C8
Port I/O Module	General Purpose I/O With Set/Clear/Toggle Functionality	PTE	0x(FF)FF_8100
Port Control Module	Port I/O Control Module	PTE	0x(FF)FF_8108
Port I/O Module	General Purpose I/O With Set/Clear/Toggle Functionality	PTF	0x(FF)FF_8140
Port Control Module	Port I/O Control Module	PTF	0x(FF)FF_8148

Table 3-2. High Level Peripheral Memory Map (continued)

Peripheral	Description	Instance Name	Base Address
KBI	Keyboard Interrupt Module	KBI1	0x(FF)FF_8180
KBI	Keyboard Interrupt Module	KBI2	0x(FF)FF_81A0
IRQ	External Interrupt Module	IRQ	0x(FF)FF_81C0
RESERVED	Reserved Memory	NONE	0x(FF)FF_81E0
Port Mux Controls	Port Mux Controls	MC	0x(FF)FF_8200
ICS	Internal Clock Source	ICS	0x(FF)FF_8220
Clock Check & Select	Clock Check & Select	CCS	0x(FF)FF_8240
RESERVED	Reserved Memory	NONE	0x(FF)FF_8260
SIM	System Integration Module	SIM	0x(FF)FF_8280
PMC	Power Management Controller	PMC	0x(FF)FF_82A0
SCI	Serial Communications Interface	SCI1	0x(FF)FF_82C0
SCI	Serial Communications Interface	SCI2	0x(FF)FF_82E0
SCI	Serial Communications Interface	SCI3	0x(FF)FF_8300
SPI	Serial Peripheral Interface	SPI1	0x(FF)FF_8320
SPI	Serial Peripheral Interface	SPI2	0x(FF)FF_8340
SPI	Serial Peripheral Interface	SPI3	0x(FF)FF_8360
IIC	Inter-Integrated IC	IIC	0x(FF)FF_8380
RESERVED	Reserved Memory	NONE	0x(FF)FF_83A0
Voltage Reference	Programmable Voltage Reference	VREF	0x(FF)FF_83C0
ADC	16-bit Successive Approximation Analog-to-Digital Converter	ADC1	0x(FF)FF_8400
ADC	16-bit Successive Approximation Analog-to-Digital Converter	ADC2	0x(FF)FF_8440
ADC	16-bit Successive Approximation Analog-to-Digital Converter	ADC3	0x(FF)FF_8480
ADC	16-bit Successive Approximation Analog-to-Digital Converter	ADC4	0x(FF)FF_84C0
RESERVED	Reserved Memory	NONE	0x(FF)FF_8500
PRACMP	Programmable Reference Analog Comparator	PRACMP1	0x(FF)FF_8520
PRACMP	Programmable Reference Analog Comparator	PRACMP2	0x(FF)FF_8540
MTIM8	8-Bit Modulo Timer	MTIM1	0x(FF)FF_8560
MTIM8	8-Bit Modulo Timer	MTIM2	0x(FF)FF_8580
MTIM16	16-Bit Modulo Timer	MTIM3	0x(FF)FF_85A0
CRC	Cyclic Redundancy Check Generator	CRC	0x(FF)FF_85C0
2-channel TPM	2-channel Timer / PWM Module	TPM	0x(FF)FF_8600
IRTC	Independent Real Time Counter	IRTC	0x(FF)FF_8640

Table 3-2. High Level Peripheral Memory Map (continued)

Peripheral	Description	Instance Name	Base Address
RESERVED	Reserved Memory	NONE	0x(FF)FF_86A0
LCD Module	LCD Module	LCD	0x(FF)FF_8700
FTSR	Flash Wrapper	FTSR1	0x(FF)FF_8780
FTSR	Flash Wrapper	FTSR2	0x(FF)FF_87A0
PDB	Programmable Delay Block	PDB	0x(FF)FF_E000
INTC	V1 ColdFire Interrupt Controller	INTC	0x(FF)FF_FFC0

The section of memory at 0x(00)C0_0000 is assigned for use by the ColdFire rapid GPIO module. See Table 3-4 for the rapid GPIO memory map and Chapter 5, “Rapid GPIO (RGPIO),” for further details on the module.

The MCF51EM256 series microcontrollers include 8-bit and 16-bit peripheral busses. The bus bridge from the ColdFire system bus to the peripheral bus is capable of serializing 32-bit accesses into two 16-bit accesses or four 8-bit accesses. This can be used to speed access to properly aligned peripheral registers. Note, not all peripheral registers are aligned to take advantage of this feature.

CPU accesses to those parts of the memory map marked as not implemented in Table 3-2 result in an illegal address reset if CPUCR[ARD] = 0 or an address error exception if CPUCR[ARD] = 1.

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

3.2 Detailed Register Addresses and Bit Assignments

The ColdFire interrupt controller module is mapped in the peripheral space and occupies a 64-byte space at the upper end of memory. Accordingly, its address decode is defined as 0x(FF)FF_FFC0–0x(FF)FF_FFFF. This 64-byte space includes the program-visible interrupt controller registers as well as the space used for interrupt acknowledge (IACK) cycles.

There is a nonvolatile register area consisting of a block of 20 bytes in flash memory as shown in Table 3-15. Nonvolatile register locations include:

- NVPROT and NVOPT are loaded into working registers at reset
- An 8-byte backdoor comparison key that optionally allows a user to gain controlled access to secure memory

Because the nonvolatile register locations are flash memory, they must be erased and programmed like other flash memory locations. MCF51EM256 series have two sets of nonvolatile registers, one for each flash block. Refer to Section 3.6, “Flash Module Reserved Memory Locations,” for detailed information.

Table 3-3 is a summary of all user-accessible peripheral registers and control bits. Cells that are not associated with named bits are shaded. A shaded cell with a 0 indicates this unused bit always reads as a

0. Shaded cells with dashes indicate unused or reserved bit locations that could read as 1s or 0s. When writing to these bits, write a 0 unless otherwise specified.

Table 3-3. Detailed Peripheral Memory Map (Sheet 1 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(00)C0_0000	**RGPIO_DIR**	DIR[15:8] (Read/Write)							
0x(00)C0_0001		DIR[7:0] (Read/Write)							
0x(00)C0_0002	**RGPIO_DATA**	DATA[15:8] (Read/Write)							
0x(00)C0_0003		DATA[7:0] (Read/Write)							
0x(00)C0_0004	**RGPIO_ENB**	ENB[15:8] (Read/Write)							
0x(00)C0_0005		ENB[7:0] (Read/Write)							
0x(00)C0_0006	**RGPIO_CLR**	CLR[15:8] (Write only)							
		CLR[7:0] (Write only)							
0x(00)C0_0006	**RGPIO_DATA**	DATA[15:8] (Read only)							
		DATA[7:0] (Read only)							
0x(00)C0_0008	**RGPIO_DIR**	DIR[15:8] (Read only)							
0x(00)C0_0009		DIR[7:0] (Read only)							
0x(00)C0_000A	**RGPIO_SET**	SET[15:8] (Write only)							
		SET[7:0] (Write only)							
0x(00)C0_000A	**RGPIO_DATA**	DATA[15:8] (Read only)							
		DATA[7:0] (Read only)							
0x(00)C0_000C	**RGPIO_DIR**	DIR[15:8] (Read only)							
		DIR[7:0] (Read only)							
0x(00)C0_000E	**RGPIO_TOG**	TOG[15:8] (Write only)							
		TOG[17:0] (Write only)							
0x(00)C0_000E	**RGPIO_DATA**	DATA[15:8] (Read only)							
		DATA[7:0] (Read only)							
0x(00)C0_0010– 0x(FF)FF_7FFF	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8000	**PTAD**	D7	D6	D5	D4	D3	D2	D1	D0
0x(FF)FF_8001	**PTADD**	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
0x(FF)FF_8002	**PTASET**	SET7	SET6	SET5	SET4	SET3	SET2	SET1	SET0
0x(FF)FF_8003	**PTACLR**	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
0x(FF)FF_8004	**PTATOG**	TOG7	TOG6	TOG5	TOG4	TOG3	TOG2	TOG1	TOG0
0x(FF)FF_8005– 0x(FF)FF_8007	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8008	**PTAPE**	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0

Table 3-3. Detailed Peripheral Memory Map (Sheet 2 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8009	**PTASE**	SE7	SE6	SE5	SE4	SE3	SE2	SE1	SE0
0x(FF)FF_800A	**PTADS**	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
0x(FF)FF_800B	**PTAIFE**	IFE7	IFE6	IFE5	IFE4	IFE3	IFE2	IFE1	IFE0
0x(FF)FF_800C– 0x(FF)FF_803F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8040	**PTBD**	D7	D6	D5	D4	D3	D2	D1	D0
0x(FF)FF_8041	**PTBDD**	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
0x(FF)FF_8042	**PTBSET**	SET7	SET6	SET5	SET4	SET3	SET2	SET1	SET0
0x(FF)FF_8043	**PTBCLR**	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
0x(FF)FF_8044	**PTBTOG**	TOG7	TOG6	TOG5	TOG4	TOG3	TOG2	TOG1	TOG0
0x(FF)FF_8045– 0x(FF)FF_8047	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8048	**PTBPE**	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
0x(FF)FF_8049	**PTBSE**	SE7	SE6	SE5	SE4	SE3	SE2	SE1	SE0
0x(FF)FF_804A	**PTBDS**	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
0x(FF)FF_804B	**PTBIFE**	IFE7	IFE6	IFE5	IFE4	IFE3	IFE2	IFE1	IFE0
0x(FF)FF_804C– 0x(FF)FF_807F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8080	**PTCD**	D7	D6	D5	D4	D3	D2	D1	D0
0x(FF)FF_8081	**PTCDD**	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
0x(FF)FF_8082	**PTCSET**	SET7	SET6	SET5	SET4	SET3	SET2	SET1	SET0
0x(FF)FF_8083	**PTCCLR**	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
0x(FF)FF_8084	**PTCTOG**	TOG7	TOG6	TOG5	TOG4	TOG3	TOG2	TOG1	TOG0
0x(FF)FF_8085– 0x(FF)FF_8087	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8088	**PTCPE**	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
0x(FF)FF_8089	**PTCSE**	SE7	SE6	SE5	SE4	SE3	SE2	SE1	SE0
0x(FF)FF_808A	**PTCDS**	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
0x(FF)FF_808B	**PTCIFE**	IFE7	IFE6	IFE5	IFE4	IFE3	IFE2	IFE1	IFE0
0x(FF)FF_808C– 0x(FF)FF_80BF	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_80C0	**PTDD**	D7	D6	D5	D4	D3	D2	D1	D0
0x(FF)FF_80C1	**PTDDD**	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
0x(FF)FF_80C2	**PTDSET**	SET7	SET6	SET5	SET4	SET3	SET2	SET1	SET0
0x(FF)FF_80C3	**PTDCLR**	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0

Table 3-3. Detailed Peripheral Memory Map (Sheet 3 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_80C4	**PTDTOG**	TOG7	TOG6	TOG5	TOG4	TOG3	TOG2	TOG1	TOG0
0x(FF)FF_80C5– 0x(FF)FF_80C7	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_80C8	**PTDPE**	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
0x(FF)FF_80C9	**PTDSE**	SE7	SE6	SE5	SE4	SE3	SE2	SE1	SE0
0x(FF)FF_80CA	**PTDDS**	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
0x(FF)FF_80CB	**PTDIFE**	IFE7	IFE6	IFE5	IFE4	IFE3	IFE2	IFE1	IFE0
0x(FF)FF_80CC– 0x(FF)FF_80FF	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8100	**PTED**	D7	D6	D5	D4	D3	D2	D1	D0
0x(FF)FF_8101	**PTEDD**	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
0x(FF)FF_8102	**PTESET**	SET7	SET6	SET5	SET4	SET3	SET2	SET1	SET0
0x(FF)FF_8103	**PTECLR**	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
0x(FF)FF_8104	**PTETOG**	TOG7	TOG6	TOG5	TOG4	TOG3	TOG2	TOG1	TOG0
0x(FF)FF_8105– 0x(FF)FF_8107	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8108	**PTEPE**	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
0x(FF)FF_8109	**PTESE**	SE7	SE6	SE5	SE4	SE3	SE2	SE1	SE0
0x(FF)FF_810A	**PTEDS**	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
0x(FF)FF_810B	**PTEIFE**	IFE7	IFE6	IFE5	IFE4	IFE3	IFE2	IFE1	IFE0
0x(FF)FF_810C– 0x(FF)FF_813F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8140	**PTFD**	D7	D6	D5	D4	D3	D2	D1	D0
0x(FF)FF_8141	**PTFDD**	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
0x(FF)FF_8142	**PTFSET**	SET7	SET6	SET5	SET4	SET3	SET2	SET1	SET0
0x(FF)FF_8143	**PTFCLR**	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
0x(FF)FF_8144	**PTFTOG**	TOG7	TOG6	TOG5	TOG4	TOG3	TOG2	TOG1	TOG0
0x(FF)FF_8145– 0x(FF)FF_8147	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8148	**PTFPE**	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
0x(FF)FF_8149	**PTFSE**	SE7	SE6	SE5	SE4	SE3	SE2	SE1	SE0
0x(FF)FF_814A	**PTFDS**	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
0x(FF)FF_814B	**PTFIFE**	IFE7	IFE6	IFE5	IFE4	IFE3	IFE2	IFE1	IFE0
0x(FF)FF_814C– 0x(FF)FF_817F	Reserved	—	—	—	—	—	—	—	—

Table 3-3. Detailed Peripheral Memory Map (Sheet 4 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8180	**KBI1SC**	0	0	0	0	KBF	KBACK	KBIE	KBIMOD
0x(FF)FF_8181	**KBI1PE**	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
0x(FF)FF_8182	**KBI1ES**	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
0x(FF)FF_8183– 0x(FF)FF_819F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_81A0	**KBI2SC**	0	0	0	0	KBF	KBACK	KBIE	KBIMOD
0x(FF)FF_81A1	**KBI2PE**	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
0x(FF)FF_81A2	**KBI2ES**	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
0x(FF)FF_81A3– 0x(FF)FF_81BF	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_81C0	**IRQSC**	0	IRQPDD	IRQEDG	IRQPE	IRQF	IRQACK	IRQIE	IRQMOD
0x(FF)FF_81C1– 0x(FF)FF_81FF	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8200	**PTAPF1**	A7		A6		A5		A4	
0x(FF)FF_8201	**PTAPF2**	A3		A2		A1		A0	
0x(FF)FF_8202	**PTBPF1**	B7		B6		B5		B4	
0x(FF)FF_8203	**PTBPF2**	B3		B2		B1		B0	
0x(FF)FF_8204	**PTCPF1**	C7		C6		C5		C4	
0x(FF)FF_8205	**PTCPF2**	C3		C2		C1		C0	
0x(FF)FF_8206	**PTDPF1**	D7		D6		D5		D4	
0x(FF)FF_8207	**PTDPF2**	D3		D2		D1		D0	
0x(FF)FF_8208	**PTEPF1**	E7		E6		E5		E4	
0x(FF)FF_8209	**PTEPF2**	E3		E2		E1		E0	
0x(FF)FF_820A	**PTFPF1**	F7		F6		F5		F4	
0x(FF)FF_820B	**PTFPF2**	F3		F2		F1		F0	
0x(FF)FF_820C	**LCDPF1**	LCD9		LCD8		LCD7		LCD6	
0x(FF)FF_820D	**LCDPF2**	0	0	0	0	0	0	LCD35	
0x(FF)FF_820E– 0x(FF)FF_821F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8220	**ICSC1**	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
0x(FF)FF_8221	**ICSC2**	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
0x(FF)FF_8222	**ICSTRM**	TRIM							
0x(FF)FF_8223	**ICSSC**	DRST/DRS		DMX32	IREFST	CLKST		OSCINIT	FTRIM
0x(FF)FF_8224– 0x(FF)FF_823F	Reserved	—	—	—	—	—	—	—	—

Table 3-3. Detailed Peripheral Memory Map (Sheet 5 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8240	**CCSCTRL**	0	0	0	0	0	EN	TEST	SEL
0x(FF)FF_8241	**CCSTMR1**	CNT1							
0x(FF)FF_8242	**CCSTMR2**	CNT2							
0x(FF)FF_8243	**CCSTMRIR**	CNTIR							
0x(FF)FF_8244– 0x(FF)FF_827F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8280	**SRS**	POR	PIN	COP	ILOP	ILAD	0	LVD	0
0x(FF)FF_8281	**SOPT1**	0	0	STOPE	WAITE	COPT		COPCLKS	COPW
0x(FF)FF_8282	**SOPT2**	0	0	0	PMC_LVD_TRIM				
0x(FF)FF_8283	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8284	**SDIDH**	REV				ID11	ID10	ID9	ID8
0x(FF)FF_8285	**SDIDL**	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
0x(FF)FF_8286	**SCGC1**	ADC4	ADC3	ADC2	ADC1	IIC	SCI3	SCI2	SCI1
0x(FF)FF_8287	**SCGC2**	CMP2	CMP1	VREF	IRQ	LCD	SPI3	SPI2	SPI1
0x(FF)FF_8288	**SCGC3**	KBI2	KBI1	PTF	PTE	PTD	PTC	PTB	PTA
0x(FF)FF_8289	**SCGC4**	1	PM	CRC	TPM	PDB	MTIM3	MTIM2	MTIM1
0x(FF)FF_828A	**SCGC5**	—	—	—	—	—	—	FTSR2	FTSR1
0x(FF)FF_828B	**SIMCO**	—	—	—	—	CS			
0x(FF)FF_828C	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_828D	**SIMIPS1**	TPM	0	MTIM3		0	MTIM2		MTIM1
0x(FF)FF_828E	**SIMIPS2**	RX2IN	RX1IN	MTBASE2		MTBASE1		MODTX2	MODTX1
0x(FF)FF_828F– 0x(FF)FF_829F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_82A0	**SPMSC1**	LVDF	LVDACK	LVDIE	LVDRE	LVDSE	LVDE	0	BGBE
0x(FF)FF_82A1	**SPMSC2**	LPR	LPRS	LPWUI	0	PPDF	PPDACK	PPDE	PPDC
0x(FF)FF_82A2	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_82A3	**SPMSC3**	LVWF	LVWACK	LVDV	LVWV	LVWIE	0	0	0
0x(FF)FF_82A4– 0x(FF)FF_82BF	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_82C0	**SCI1BDH**	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x(FF)FF_82C1	**SCI1BDL**	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x(FF)FF_82C2	**SCI1C1**	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x(FF)FF_82C3	**SCI1C2**	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x(FF)FF_82C4	**SCI1S1**	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF

Table 3-3. Detailed Peripheral Memory Map (Sheet 6 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_82C5	**SCI1S2**	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x(FF)FF_82C6	**SCI1C3**	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x(FF)FF_82C7	**SCI1D**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_82C8– 0x(FF)FF_82DF	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_82E0	**SCI2BDH**	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x(FF)FF_82E1	**SCI2BDL**	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x(FF)FF_82E2	**SCI2C1**	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x(FF)FF_82E3	**SCI2C2**	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x(FF)FF_82E4	**SCI2S1**	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x(FF)FF_82E5	**SCI2S2**	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x(FF)FF_82E6	**SCI2C3**	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x(FF)FF_82E7	**SCI2D**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_82E8– 0x(FF)FF_82FF	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8300	**SCI3BDH**	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x(FF)FF_8301	**SCI3BDL**	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x(FF)FF_8302	**SCI3C1**	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x(FF)FF_8303	**SCI3C2**	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x(FF)FF_8304	**SCI3S1**	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x(FF)FF_8305	**SCI3S2**	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x(FF)FF_8306	**SCI3C3**	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x(FF)FF_8307	**SCI3D**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8308– 0x(FF)FF_831F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8320	**SPI1C1**	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x(FF)FF_8321	**SPI1C2**	SPMIE	SPIMODE	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
0x(FF)FF_8322	**SPI1BR**	0	SPPR2	SPPR1	SPPR0	SPR3	SPR2	SPR1	SPR0
0x(FF)FF_8323	**SPI1S**	SPRF	SPMF	SPTEF	MODF	RNFULLF	TNEAREF	TXFULLF	RFIFOEF
0x(FF)FF_8324	**SPI1DH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_8325	**SPI1DL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8326	**SPI1MH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_8327	**SPI1ML**	Bit 7	6	5	4	3	2	1	Bit 0

Table 3-3. Detailed Peripheral Memory Map (Sheet 7 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8328	**SPI1C3**	0	0	TNEAREF MARK	RNFULL MARK	INTCLR	TNEARIEN	RNFULLIEN	FIFOMODE
0x(FF)FF_8329	**SPI1CI**	TXFERR	RXFERR	TXFOF	RXFOF	TNEAREFCI	RNFULLFCI	SPTEFCI	SPRFCI
0x(FF)FF_8330– 0x(FF)FF_833F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8340	**SPI2C1**	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x(FF)FF_8341	**SPI2C2**	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
0x(FF)FF_8342	**SPI2BR**	0	SPPR2	SPPR1	SPPR0	SPR3	SPR2	SPR1	SPR0
0x(FF)FF_8343	**SPI2S**	SPRF	0	SPTEF	MODF	0	0	0	0
0x(FF)FF_8344	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8345	**SPI2D**	DATA							
0x(FF)FF_8346– 0x(FF)FF_835F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8360	**SPI3C1**	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x(FF)FF_8361	**SPI3C2**	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
0x(FF)FF_8362	**SPI3BR**	0	SPPR2	SPPR1	SPPR0	SPR3	SPR2	SPR1	SPR0
0x(FF)FF_8363	**SPI3S**	SPRF	0	SPTEF	MODF	0	0	0	0
0x(FF)FF_8364	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8365	**SPI3D**	DATA							
0x(FF)FF_8366– 0x(FF)FF_837F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8380	**IICA1**	AD7	AD6	AD5	AD4	AD3	AD2	AD1	0
0x(FF)FF_8381	**IICF**	MULT		ICR					
0x(FF)FF_8382	**IICC1**	IICEN	IICIE	MST	TX	TXAK	RSTA	0	0
0x(FF)FF_8383	**IICS**	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
0x(FF)FF_8384	**IICD**	DATA							
0x(FF)FF_8385	**IICC2**	GCAEN	ADEXT	0	0	0	AD10	AD9	AD8
0x(FF)FF_8386	**IICSMB**	FACK	ALERTEN	SIICAEN	TCKSEL	SLTF	SHTF	0	0
0x(FF)FF_8387	**IICA2**	SAD7	SAD6	SAD5	SAD4	SAD3	SAD2	SAD1	0
0x(FF)FF_8388	**IICSLTH**	SSLT15	SSLT14	SSLT13	SSLT12	SSLT11	SSLT10	SSLT9	SSLT8
0x(FF)FF_8389	**IICSLTL**	SSLT7	SSLT6	SSLT5	SSLT4	SSLT3	SSLT2	SSLT1	SSLT0
0x(FF)FF_838A	**IICFLT**	0	0	0	0	FLT3	FLT2	FLT1	FLT0
0x(FF)FF_838B– 0x(FF)FF_83BF	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_83C0	**VREFTRM**	TRM							

Table 3-3. Detailed Peripheral Memory Map (Sheet 8 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_83C1	**VREFSC**	VREFEN	0	0	0	0	VREFST	MODE	
0x(FF)FF_83C2– 0x(FF)FF_83FF	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8400	**ADC1SC1A**	COCOA	AIENA	DIFFA	ADCHA				
0x(FF)FF_8401	**ADC1SC1B**	COCOB	AIENB	DIFFB	ADCHB				
0x(FF)FF_8402	**ADC1CFG1**	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
0x(FF)FF_8403	**ADC1CFG2**	0	0	0	0	ADACKEN	ADHSC	ADLSTS	
0x(FF)FF_8404	**ADC1RHA**	D[15:8]							
0x(FF)FF_8405	**ADC1RLA**	D[7:0]							
0x(FF)FF_8406	**ADC1RHB**	D[15:8]							
0x(FF)FF_8407	**ADC1RLB**	D[7:0]							
0x(FF)FF_8408	**ADC1CV1H**	CV1[15:8]							
0x(FF)FF_8409	**ADC1CV1L**	CV1[7:0]							
0x(FF)FF_840A	**ADC1CV2H**	CV2[15:8]							
0x(FF)FF_840B	**ADC1CV2L**	CV2[[7:0]							
0x(FF)FF_840C	**ADC1SC2**	ADACT	ADTRG	ACFE	ACFGT	ACREN	0	REFSEL	
0x(FF)FF_840D	**ADC1SC3**	CAL	CAF	0	0	ADCO	ADGE	AVGS	
0x(FF)FF_840E	**ADC1OFSH**	OFS[15:8]							
0x(FF)FF_840F	**ADC1OFSL**	OFS[7:0]							
0x(FF)FF_8410	**ADC1PGH**	PG[15:8]							
0x(FF)FF_8411	**ADC1PGL**	PG[7:0]							
0x(FF)FF_8412	**ADC1MGH**	MG[15:8]							
0x(FF)FF_8413	**ADC1MGL**	MG[7:0]							
0x(FF)FF_8414	**ADC1CLPD**	0	0	CLPD[5:0]					
0x(FF)FF_8415	**ADC1CLPS**	0	0	CLPS[5:0]					
0x(FF)FF_8416	**ADC1CLP4H**	0	0	0	0	0	0	CLP4[9:8]	
0x(FF)FF_8417	**ADC1CLP4L**	CLP4[7:0]							
0x(FF)FF_8418	**ADC1CLP3H**	0	0	0	0	0	0	0	CLP3[8]
0x(FF)FF_8419	**ADC1CLP3L**	CLP3[7:0]							
0x(FF)FF_841A	**ADC1CLP2**	CLP2[7:0]							
0x(FF)FF_841B	**ADC1CLP1**	0	CLP1[6:0]						
0x(FF)FF_841C	**ADC1CLP0**	0	0	CLP0[5:0]					
0x(FF)FF_841D	Reserved	—	—	—	—	—	—	—	—

Table 3-3. Detailed Peripheral Memory Map (Sheet 9 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_841E	**ADC1CLMD**	0	0	CLMD[5:0]					
0x(FF)FF_841F	**ADC1CLMS**	0	0	CLMS[5:0]					
0x(FF)FF_8420	**ADC1CLM4H**	0	0	0	0	0	0	CLM4[9:8]	
0x(FF)FF_8421	**ADC1CLM4L**	CLM4[7:0]							
0x(FF)FF_8422	**ADC1CLM3H**	0	0	0	0	0	0	0	CLM3[8]
0x(FF)FF_8423	**ADC1CLM3L**	CLM3[7:0]							
0x(FF)FF_8424	**ADC1CLM2**	CLM2[7:0]							
0x(FF)FF_8425	**ADC1CLM1**	0	CLM1[6:0]						
0x(FF)FF_8426	**ADC1CLM0**	0	0	CLM0[5:0]					
0x(FF)FF_8427	**ADC1APCTL1**	ADPC[7:0]							
0x(FF)FF_8428	**ADC1APCTL2**	ADPC[15:8]							
0x(FF)FF_8429	**ADC1APCTL3**	ADPC[23:16]							
0x(FF)FF_842A	**ADC1APCTL4**	ADPC[31:24]							
0x(FF)FF_842B– 0x(FF)FF_843F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8440	**ADC2SC1A**	COCOA	AIENA	DIFFA	ADCHA				
0x(FF)FF_8441	**ADC2SC1B**	COCOB	AIENB	DIFFB	ADCHB				
0x(FF)FF_8442	**ADC2CFG1**	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
0x(FF)FF_8443	**ADC2CFG2**	0	0	0	0	ADACKEN	ADHSC	ADLSTS	
0x(FF)FF_8444	**ADC2RHA**	D[15:8]							
0x(FF)FF_8445	**ADC2RLA**	D[7:0]							
0x(FF)FF_8446	**ADC2RHB**	D[15:8]							
0x(FF)FF_8447	**ADC2RLB**	D[7:0]							
0x(FF)FF_8448	**ADC2CV1H**	CV1[15:8]							
0x(FF)FF_8449	**ADC2CV1L**	CV1[7:0]							
0x(FF)FF_844A	**ADC2CV2H**	CV2[15:8]							
0x(FF)FF_844B	**ADC2CV2L**	CV2[[7:0]							
0x(FF)FF_844C	**ADC2SC2**	ADACT	ADTRG	ACFE	ACFGT	ACREN	0	REFSEL	
0x(FF)FF_844D	**ADC2SC3**	CAL	CAF	0	0	ADCO	ADGE	AVGS	
0x(FF)FF_844E	**ADC2OFSH**	OFS[15:8]							
0x(FF)FF_844F	**ADC2OFSL**	OFS[7:0]							
0x(FF)FF_8450	**ADC2PGH**	PG[15:8]							
0x(FF)FF_8451	**ADC2PGL**	PG[7:0]							

Table 3-3. Detailed Peripheral Memory Map (Sheet 10 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8452	**ADC2MGH**	MG[15:8]							
0x(FF)FF_8453	**ADC2MGL**	MG[7:0]							
0x(FF)FF_8454	**ADC2CLPD**	0	0	CLPD[5:0]					
0x(FF)FF_8455	**ADC2CLPS**	0	0	CLPS[5:0]					
0x(FF)FF_8456	**ADC2CLP4H**	0	0	0	0	0	0	CLP4[9:8]	
0x(FF)FF_8457	**ADC2CLP4L**	CLP4[7:0]							
0x(FF)FF_8458	**ADC2CLP3H**	0	0	0	0	0	0	0	CLP3[8]
0x(FF)FF_8459	**ADC2CLP3L**	CLP3[7:0]							
0x(FF)FF_845A	**ADC2CLP2**	CLP2[7:0]							
0x(FF)FF_845B	**ADC2CLP1**	0	CLP1[6:0]						
0x(FF)FF_845C	**ADC2CLP0**	0	0	CLP0[5:0]					
0x(FF)FF_845D	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_845E	**ADC2CLMD**	0	0	CLMD[5:0]					
0x(FF)FF_845F	**ADC2CLMS**	0	0	CLMS[5:0]					
0x(FF)FF_8460	**ADC2CLM4H**	0	0	0	0	0	0	CLM4[9:8]	
0x(FF)FF_8461	**ADC2CLM4L**	CLM4[7:0]							
0x(FF)FF_8462	**ADC2CLM3H**	0	0	0	0	0	0	0	CLM3[8]
0x(FF)FF_8463	**ADC2CLM3L**	CLM3[7:0]							
0x(FF)FF_8464	**ADC2CLM2**	CLM2[7:0]							
0x(FF)FF_8465	**ADC2CLM1**	0	CLM1[6:0]						
0x(FF)FF_8466	**ADC2CLM0**	0	0	CLM0[5:0]					
0x(FF)FF_8467– 0x(FF)FF_847F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8480	**ADC3SC1A**	COCOA	AIENA	DIFFA	ADCHA				
0x(FF)FF_8481	**ADC3SC1B**	COCOB	AIENB	DIFFB	ADCHB				
0x(FF)FF_8482	**ADC3CFG1**	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
0x(FF)FF_8483	**ADC3CFG2**	0	0	0	0	ADACKEN	ADHSC	ADLSTS	
0x(FF)FF_8484	**ADC3RHA**	D[15:8]							
0x(FF)FF_8485	**ADC3RLA**	D[7:0]							
0x(FF)FF_8486	**ADC3RHB**	D[15:8]							
0x(FF)FF_8487	**ADC3RLB**	D[7:0]							
0x(FF)FF_8488	**ADC3CV1H**	CV1[15:8]							
0x(FF)FF_8489	**ADC3CV1L**	CV1[7:0]							

Table 3-3. Detailed Peripheral Memory Map (Sheet 11 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_848A	**ADC3CV2H**	CV2[15:8]							
0x(FF)FF_848B	**ADC3CV2L**	CV2[[7:0]							
0x(FF)FF_848C	**ADC3SC2**	ADACT	ADTRG	ACFE	ACFGT	ACREN	0	REFSEL	
0x(FF)FF_848D	**ADC3SC3**	CAL	CAF	0	0	ADCO	ADGE	AVGS	
0x(FF)FF_848E	**ADC3OFSH**	OFS[15:8]							
0x(FF)FF_848F	**ADC3OFSL**	OFS[7:0]							
0x(FF)FF_8490	**ADC3PGH**	PG[15:8]							
0x(FF)FF_8491	**ADC3PGL**	PG[7:0]							
0x(FF)FF_8492	**ADC3MGH**	MG[15:8]							
0x(FF)FF_8493	**ADC3MGL**	MG[7:0]							
0x(FF)FF_8494	**ADC3CLPD**	0	0	CLPD[5:0]					
0x(FF)FF_8495	**ADC3CLPS**	0	0	CLPS[5:0]					
0x(FF)FF_8496	**ADC3CLP4H**	0	0	0	0	0	0	CLP4[9:8]	
0x(FF)FF_8497	**ADC3CLP4L**	CLP4[7:0]							
0x(FF)FF_8498	**ADC3CLP3H**	0	0	0	0	0	0	0	CLP3[8]
0x(FF)FF_8499	**ADC3CLP3L**	CLP3[7:0]							
0x(FF)FF_849A	**ADC3CLP2**	CLP2[7:0]							
0x(FF)FF_849B	**ADC3CLP1**	0	CLP1[6:0]						
0x(FF)FF_849C	**ADC3CLP0**	0	0	CLP0[5:0]					
0x(FF)FF_849D	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_849E	**ADC3CLMD**	0	0	CLMD[5:0]					
0x(FF)FF_849F	**ADC3CLMS**	0	0	CLMS[5:0]					
0x(FF)FF_84A0	**ADC3CLM4H**	0	0	0	0	0	0	CLM4[9:8]	
0x(FF)FF_84A1	**ADC3CLM4L**	CLM4[7:0]							
0x(FF)FF_84A2	**ADC3CLM3H**	0	0	0	0	0	0	0	CLM3[8]
0x(FF)FF_84A3	**ADC3CLM3L**	CLM3[7:0]							
0x(FF)FF_84A4	**ADC3CLM2**	CLM2[7:0]							
0x(FF)FF_84A5	**ADC3CLM1**	0	CLM1[6:0]						
0x(FF)FF_84A6	**ADC3CLM0**	0	0	CLM0[5:0]					
0x(FF)FF_84A7– 0x(FF)FF_84BF	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_84C0	**ADC4SC1A**	COCOA	AIENA	DIFFA	ADCHA				
0x(FF)FF_84C1	**ADC4SC1B**	COCOB	AIENB	DIFFB	ADCHB				

Table 3-3. Detailed Peripheral Memory Map (Sheet 12 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_84C2	**ADC4CFG1**	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
0x(FF)FF_84C3	**ADC4CFG2**	0	0	0	0	ADACKEN	ADHSC	ADLSTS	
0x(FF)FF_84C4	**ADC4RHA**	D[15:8]							
0x(FF)FF_84C5	**ADC4RLA**	D[7:0]							
0x(FF)FF_84C6	**ADC4RHB**	D[15:8]							
0x(FF)FF_84C7	**ADC4RLB**	D[7:0]							
0x(FF)FF_84C8	**ADC4CV1H**	CV1[15:8]							
0x(FF)FF_84C9	**ADC4CV1L**	CV1[7:0]							
0x(FF)FF_84CA	**ADC4CV2H**	CV2[15:8]							
0x(FF)FF_84CB	**ADC4CV2L**	CV2[[7:0]							
0x(FF)FF_84CC	**ADC4SC2**	ADACT	ADTRG	ACFE	ACFGT	ACREN	0	REFSEL	
0x(FF)FF_84CD	**ADC4SC3**	CAL	CAF	0	0	ADCO	ADGE	AVGS	
0x(FF)FF_84CE	**ADC4OFSH**	OFS[15:8]							
0x(FF)FF_84CF	**ADC4OFSL**	OFS[7:0]							
0x(FF)FF_84D0	**ADC4PGH**	PG[15:8]							
0x(FF)FF_84D1	**ADC4PGL**	PG[7:0]							
0x(FF)FF_84D2	**ADC4MGH**	MG[15:8]							
0x(FF)FF_84D3	**ADC4MGL**	MG[7:0]							
0x(FF)FF_84D4	**ADC4CLPD**	0	0	CLPD[5:0]					
0x(FF)FF_84D5	**ADC4CLPS**	0	0	CLPS[5:0]					
0x(FF)FF_84D6	**ADC4CLP4H**	0	0	0	0	0	0	CLP4[9:8]	
0x(FF)FF_84D7	**ADC4CLP4L**	CLP4[7:0]							
0x(FF)FF_84D8	**ADC4CLP3H**	0	0	0	0	0	0	0	CLP3[8]
0x(FF)FF_84D9	**ADC4CLP3L**	CLP3[7:0]							
0x(FF)FF_84DA	**ADC4CLP2**	CLP2[7:0]							
0x(FF)FF_84DB	**ADC4CLP1**	0	CLP1[6:0]						
0x(FF)FF_84DC	**ADC4CLP0**	0	0	CLP0[5:0]					
0x(FF)FF_84DD	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_84DE	**ADC4CLMD**	0	0	CLMD[5:0]					
0x(FF)FF_84DF	**ADC4CLMS**	0	0	CLMS[5:0]					
0x(FF)FF_84E0	**ADC4CLM4H**	0	0	0	0	0	0	CLM4[9:8]	
0x(FF)FF_84E1	**ADC4CLM4L**	CLM4[7:0]							
0x(FF)FF_84E2	**ADC4CLM3H**	0	0	0	0	0	0	0	CLM3[8]

Table 3-3. Detailed Peripheral Memory Map (Sheet 13 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_84E3	**ADC4CLM3L**	CLM3[7:0]							
0x(FF)FF_84E4	**ADC4CLM2**	CLM2[7:0]							
0x(FF)FF_84E5	**ADC4CLM1**	0	CLM1[6:0]						
0x(FF)FF_84E6	**ADC4CLM0**	0	0	CLM0[5:0]					
0x(FF)FF_84E7–0x(FF)FF_851F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8520	**PRACMP1CS**	ACEN	ACMPF	0	ACOPE	ACMPO	ACINTS		ACIEN
0x(FF)FF_8521	**PRACMP1C0**	0	ACPSEL			0	ACNSEL		
0x(FF)FF_8522	**PRACMP1C1**	PRGEN	PRGINS	0	PRGOS4	PRGOS3	PRGOS2	PRGOS1	PRGOS0
0x(FF)FF_8523	**PRACMP1C2**	0	ACIPE6	ACIPE5	ACIPE4	ACIPE3	ACIPE2	ACIPE1	ACIPE0
0x(FF)FF_8524–0x(FF)FF_853F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8540	**PRACMP2CS**	ACEN	ACMPF	0	ACOPE	ACMPO	ACINTS		ACIEN
0x(FF)FF_8541	**PRACMP2C0**	0	ACPSEL			0	ACNSEL		
0x(FF)FF_8542	**PRACMP2C1**	PRGEN	PRGINS	0	PRGOS4	PRGOS3	PRGOS2	PRGOS1	PRGOS0
0x(FF)FF_8543	**PRACMP2C2**	0	ACIPE6	ACIPE5	ACIPE4	ACIPE3	ACIPE2	ACIPE1	ACIPE0
0x(FF)FF_8544–0x(FF)FF_855F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8560	**MTIM1SC**	TOF	TOIE	TRST	TSTP	0	0	0	0
0x(FF)FF_8561	**MTIM1CLK**	0	0	CLKS		PS			
0x(FF)FF_8562	**MTIM1CNT**	COUNT							
0x(FF)FF_8563	**MTIM1MOD**	MOD							
0x(FF)FF_8564–0x(FF)FF_857F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8580	**MTIM2SC**	TOF	TOIE	TRST	TSTP	0	0	0	0
0x(FF)FF_8581	**MTIM2CLK**	0	0	CLKS		PS			
0x(FF)FF_8582	**MTIM2CNT**	COUNT							
0x(FF)FF_8583	**MTIM2MOD**	MOD							
0x(FF)FF_8584–0x(FF)FF_859F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_85A0	**MTIM3SC**	TOF	TOIE	TRST	TSTP	0	0	0	0
0x(FF)FF_85A1	**MTIM3CLK**	0	0	CLKS		PS			
0x(FF)FF_85A2	**MTIM3CNTH**	CNTH							
0x(FF)FF_85A3	**MTIM3CNTL**	CNTL							

Table 3-3. Detailed Peripheral Memory Map (Sheet 14 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_85A4	**MTIM3MODH**	MODH							
0x(FF)FF_85A5	**MTIM3MODL**	MODL							
0x(FF)FF_85A6– 0x(FF)FF_85BF	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_85C0	**CRCH**	CRC[15:8]							
0x(FF)FF_85C1	**CRCL**	CRC[7:0]							
0x(FF)FF_85C2	**TRANSPOSE**	TRANSPOSE[7:0]							
0x(FF)FF_85C3– 0x(FF)FF_85FF	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8600	**TPMSC**	TOF	TOIE	CPWMS	CLKS		PS		
0x(FF)FF_8601	**TPMCNTH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_8602	**TPMCNTL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8603	**TPMMODH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_8604	**TPMMODL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8605	**TPMC0SC**	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x(FF)FF_8606	**TPMC0VH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_8607	**TPMC0VL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8608	**TPMC1SC**	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x(FF)FF_8609	**TPMC1VH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_860A	**TPMC1VL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_860B– 0x(FF)FF_863F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8640	**IRTC_YEARMON**	YEAR							
		Value remains in binary as year indicates offset from base year							
0x(FF)FF_8641		0	0	0	0	MONTH			
		MONTH (tens)				MONTH (units)			
0x(FF)FF_8642	**IRTC_DAYS**	0	0	0	0	0	DAY_OF_WEEK		
		0	0	0	0	0	Value remains unchanged		
0x(FF)FF_8643		0	0	0	DAYS				
		DAYS (tens)				DAYS (units)			
0x(FF)FF_8644	**IRTC_HOURMIN**	0	0	0	HOURS				
		HOURS (tens)				HOURS (units)			
0x(FF)FF_8645		0	0	MINUTES					
		MINUTES (tens)				MINUTES (ones)			

Table 3-3. Detailed Peripheral Memory Map (Sheet 15 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8646	IRTC_SECONDS	0	0	0	0	0	0	0	0
0x(FF)FF_8647		0	0	SECONDS					
		SECONDS (tens)				SECONDS (units)			
0x(FF)FF_8648	IRTC_ALM_YRMON	ALM_YEAR							
		Value remains in binary as year indicates offset from base year							
0x(FF)FF_8649		0	0	0	0	ALM_MONTH			
		ALM_MONTH (tens)				ALM_MONTH (units)			
0x(FF)FF_864A	IRTC_ALM_DAYS	0	0	0	0	0	0	0	0
0x(FF)FF_864B		0	0	0	ALM_DAYS				
		ALM_DAYS (tens)				ALM_DAYS (units)			
0x(FF)FF_864C	IRTC_ALM_HM	0	0	0	ALM_HOURS				
		ALM_HOURS (tens)				ALM_HOURS (units)			
0x(FF)FF_864D		0	0	ALM_MINUTES					
		ALM_MINUTES (tens)				ALM_MINUTES (units)			
0x(FF)FF_864E	IRTC_ALM_SEC	0	0	0	0	0	0	INC_S	DEC_S
0x(FF)FF_864F		0	0	ALM_SECONDS					
		ALM_SECONDS (tens)				ALM_SECONDS (units)			
0x(FF)FF_8650	IRTC_CTRL	OCLK	OFF_CHIP_CLK		TAMPER DETECT DURATION				SWR
0x(FF)FF_8651		BCDEN	DSTEN	0	0	ALARM sMATCH		WE1	WE0
0x(FF)FF_8652	IRTC_STATUS	0	0	0	0	0	0	0	0
0x(FF)FF_8653		0	0	CLV	WPE	OCAL	BERR	C_DON	INVAL
0x(FF)FF_8654	IRTC_ISR	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0
0x(FF)FF_8655		2Hz	1Hz	MIN	HR	DAY	ALM	STW	TMPR
0x(FF)FF_8656	IRTC_IER	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0
0x(FF)FF_8657		2Hz	1Hz	MIN	HR	DAY	ALM	STW	TMPR
0x(FF)FF_8658	IRTC_COUNT_DN	0	0	0	0	0	0	0	0
0x(FF)FF_8659		COUNTDOWN_COUNT							
		COUNTDOWN_COUNT (tens)				COUNTDOWN_COUNT (units)			
0x(FF)FF_865A–0x(FF)FF_865F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8660	IRTC_CFG_DATA	0	0	0	0	0	0	0	0
0x(FF)FF_8661		0	0	0	0	0	0	0	CFG0

Table 3-3. Detailed Peripheral Memory Map (Sheet 16 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8662	IRTC_DST_HOUR	0	0	0	DST_START_HOUR				
		DST_START_HOUR (tens)				DST_START_HOUR (units)			
0x(FF)FF_8663		0	0	0	DST_END_HOUR				
		DST_END_HOUR (tens)				DST_END_HOUR (units)			
0x(FF)FF_8664	IRTC_DST_MNTH	0	0	0	0	DST_START_MONTH			
		DST_START_MONTH (tens)				DST_START_MONTH (units)			
0x(FF)FF_8665		0	0	0	0	DST_END_MONTH			
		DST_END_MONTH (tens)				DST_END_MONTH (units)			
0x(FF)FF_8666	IRTC_DST_DAY	0	0	0	DST_START_DAY				
		DST_START_DAY (tens)				DST_START_DAY (units)			
0x(FF)FF_8667		0	0	0	DST_END_DAY				
		DST_END_DAY (tens)				DST_END_DAY (units)			
0x(FF)FF_8668	IRTC_COMPEN	COMPENSATION_INTERVAL							
0x(FF)FF_8669		COMPENSATION_VALUE							
0x(FF)FF_866A	IRTC_TTSR_MY	TIME STAMP YEAR							
		Value remains in binary as year indicates offset from base year							
0x(FF)FF_866B		0	0	0	0	TIME STAMP MONTHS			
		TIME STAMP MONTHS (tens)				TIME STAMP MONTHS (units)			
0x(FF)FF_866C	IRTC_TTSR_DAY	0	0	0	0	0	0	0	0
0x(FF)FF_866D		0	0	0	TIME STAMP DAY				
		TIME STAMP DAY (tens)				TIME STAMP DAY (units)			
0x(FF)FF_866E	IRTC_TTSR_HM	0	0	0	TIME STAMP HOURS				
		TIME STAMP HOURS (tens)				TIME STAMP HOURS (units)			
0x(FF)FF_866F		0	0	TIME STAMP MINUTES					
		TIME STAMP MINUTES (tens)				TIME STAMP MINUTES (units)			
0x(FF)FF_8670	IRTC_TTSR_SEC	0	0	0	0	0	0	STATUS	
		0	0	0	0	0	0	Value in Binary	
0x(FF)FF_8671		0	0	TIME STAMP SECONDS					
		TIME STAMP SECONDS (tens)				TIME STAMP SECONDS (units)			
0x(FF)FF_8672	IRTC_UP_CNTR_H	UP COUNTER Upper Word Byte 3							
0x(FF)FF_8673		UP COUNTER Upper Word Byte 2							
0x(FF)FF_8674	IRTC_UP_CNTR_L	UP COUNTER Upper Word Byte 1							
0x(FF)FF_8675		UP COUNTER Upper Word Byte 0							

Table 3-3. Detailed Peripheral Memory Map (Sheet 17 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8676– 0x(FF)FF_867F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8680	**IRTCRAM5**	MSB							
		LSB							
0x(FF)FF_8682	**IRTCRAM6**	MSB							
		LSB							
0x(FF)FF_8684	**IRTCRAM7**	MSB							
		LSB							
0x(FF)FF_8686	**IRTCRAM8**	MSB							
		LSB							
0x(FF)FF_8688	**IRTCRAM9**	MSB							
		LSB							
0x(FF)FF_868A	**IRTCRAM10**	MSB							
		LSB							
0x(FF)FF_868C	**IRTCRAM11**	MSB							
		LSB							
0x(FF)FF_868E	**IRTCRAM13**	MSB							
		LSB							
0x(FF)FF_8690	**IRTCRAM14**	MSB							
		LSB							
0x(FF)FF_8692	**IRTCRAM15**	MSB							
		LSB							
0x(FF)FF_8694	**IRTCRAM16**	MSB							
		LSB							
0x(FF)FF_8696	**IRTCRAM17**	MSB							
		LSB							
0x(FF)FF_8698	**IRTCRAM18**	MSB							
		LSB							
0x(FF)FF_869A	**IRTCRAM19**	MSB							
		LSB							
0x(FF)FF_869C	**IRTCRAM20**	MSB							
		LSB							
0x(FF)FF_869E	**IRTCRAM21**	MSB							

Table 3-3. Detailed Peripheral Memory Map (Sheet 18 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
		LSB							
0x(FF)FF_86A0–0x(FF)FF_86FF	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8700	**LCDC0**	LCDEN	SOURCE	LCLK2	LCLK1	LCLK0	DUTY2	DUTY1	DUTY0
0x(FF)FF_8701	**LCDC1**	LCDIEN	0	0	0	0	FCDEN	LCDWAI	LCDSTP
0x(FF)FF_8702	**LCDSUPPLY**	CPSEL	HREFSEL	LADJ1	LADJ0	0	BBYPASS	VSUPPLY1	VSUPPLY0
0x(FF)FF_8703	**LCDRVC**	RVEN	0	0	0	RVTRIM3	RVTRIM2	RVTRIM1	RVTRIM0
0x(FF)FF_8704	**LCDBCTL**	BLINK	ALT	BLANK	0	BMODE	BRATE2	BRATE1	BRATE0
0x(FF)FF_8705	**LCDS**	LCDIF	0	0	0	0	0	0	0
0x(FF)FF_8706	**LCDPEN0**	PEN7	PEN6	PEN5	PEN4	PEN3	PEN2	PEN1	PEN0
0x(FF)FF_8707	**LCDPEN1**	PEN15	PEN14	PEN13	PEN12	PEN11	PEN10	PEN9	PEN8
0x(FF)FF_8708	**LCDPEN2**	PEN23	PEN22	PEN21	PEN20	PEN19	PEN18	PEN17	PEN16
0x(FF)FF_8709	**LCDPEN3**	PEN31	PEN30	PEN29	PEN28	PEN27	PEN26	PEN25	PEN24
0x(FF)FF_870A	**LCDPEN4**	PEN39	PEN38	PEN37	PEN36	PEN35	PEN34	PEN33	PEN32
0x(FF)FF_870B	**LCDPEN5**	0	0	0	0	PEN43	PEN42	PEN41	PEN40
0x(FF)FF_870C–0x(FF)FF_870D	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_870E	**LCDBPEN0**	BPEN7	BPEN6	BPEN5	BPEN4	BPEN3	BPEN2	BPEN1	BPEN0
0x(FF)FF_870F	**LCDBPEN1**	BPEN15	BPEN14	BPEN13	BPEN12	BPEN11	BPEN10	BPEN9	BPEN8
0x(FF)FF_8710	**LCDBPEN2**	BPEN23	BPEN22	BPEN21	BPEN20	BPEN19	BPEN18	BPEN17	BPEN16
0x(FF)FF_8711	**LCDBPEN3**	BPEN31	BPEN30	BPEN29	BPEN28	BPEN27	BPEN26	BPEN25	BPEN24
0x(FF)FF_8712	**LCDBPEN4**	BPEN39	BPEN38	BPEN37	BPEN36	BPEN35	BPEN34	BPEN33	BPEN32
0x(FF)FF_8713	**LCDBPEN5**	0	0	0	0	BPEN43	BPEN42	BPEN41	BPEN40
0x(FF)FF_8714–0x(FF)FF_8715	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8716	**LCDWF0**	BPHLCD0	BPGLCD0	BPFLCD0	BPELCD0	BPDLCD0	BPCLCD0	BPBLCD0	BPALCD0
0x(FF)FF_8717	**LCDWF1**	BPHLCD1	BPGLCD1	BPFLCD1	BPELCD1	BPDLCD1	BPCLCD1	BPBLCD1	BPALCD1
0x(FF)FF_8718	**LCDWF2**	BPHLCD2	BPGLCD2	BPFLCD2	BPELCD2	BPDLCD2	BPCLCD2	BPBLCD2	BPALCD2
0x(FF)FF_8719	**LCDWF3**	BPHLCD3	BPGLCD3	BPFLCD3	BPELCD3	BPDLCD3	BPCLCD3	BPBLCD3	BPALCD3
0x(FF)FF_871A	**LCDWF4**	BPHLCD4	BPGLCD4	BPFLCD4	BPELCD4	BPDLCD4	BPCLCD4	BPBLCD4	BPALCD4
0x(FF)FF_871B	**LCDWF5**	BPHLCD5	BPGLCD5	BPFLCD5	BPELCD5	BPDLCD5	BPCLCD5	BPBLCD5	BPALCD5
0x(FF)FF_871C	**LCDWF6**	BPHLCD6	BPGLCD6	BPFLCD6	BPELCD6	BPDLCD6	BPCLCD6	BPBLCD6	BPALCD6
0x(FF)FF_871D	**LCDWF7**	BPHLCD7	BPGLCD7	BPFLCD7	BPELCD7	BPDLCD7	BPCLCD7	BPBLCD7	BPALCD7
0x(FF)FF_871E	**LCDWF8**	BPHLCD8	BPGLCD8	BPFLCD8	BPELCD8	BPDLCD8	BPCLCD8	BPBLCD8	BPALCD8

Table 3-3. Detailed Peripheral Memory Map (Sheet 19 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_871F	**LCDWF9**	BPHLCD9	BPGLCD9	BPFLCD9	BPELCD9	BPDLCD9	BPCLCD9	BPBLCD9	BPALCD9
0x(FF)FF_8720	**LCDWF10**	BPHLCD10	BPGLCD10	BPFLCD10	BPELCD10	BPDLCD10	BPCLCD10	BPBLCD10	BPALCD10
0x(FF)FF_8721	**LCDWF11**	BPHLCD11	BPGLCD11	BPFLCD11	BPELCD11	BPDLCD11	BPCLCD11	BPBLCD11	BPALCD11
0x(FF)FF_8722	**LCDWF12**	BPHLCD12	BPGLCD12	BPFLCD12	BPELCD12	BPDLCD12	BPCLCD12	BPBLCD12	BPALCD12
0x(FF)FF_8723	**LCDWF13**	BPHLCD13	BPGLCD13	BPFLCD13	BPELCD13	BPDLCD13	BPCLCD13	BPBLCD13	BPALCD13
0x(FF)FF_8724	**LCDWF14**	BPHLCD14	BPGLCD14	BPFLCD14	BPELCD14	BPDLCD14	BPCLCD14	BPBLCD14	BPALCD14
0x(FF)FF_8725	**LCDWF15**	BPHLCD15	BPGLCD15	BPFLCD15	BPELCD15	BPDLCD15	BPCLCD15	BPBLCD15	BPALCD15
0x(FF)FF_8726	**LCDWF16**	BPHLCD16	BPGLCD16	BPFLCD16	BPELCD16	BPDLCD16	BPCLCD16	BPBLCD16	BPALCD16
0x(FF)FF_8727	**LCDWF17**	BPHLCD17	BPGLCD17	BPFLCD17	BPELCD17	BPDLCD17	BPCLCD17	BPBLCD17	BPALCD17
0x(FF)FF_8728	**LCDWF18**	BPHLCD18	BPGLCD18	BPFLCD18	BPELCD18	BPDLCD18	BPCLCD18	BPBLCD18	BPALCD18
0x(FF)FF_8729	**LCDWF19**	BPHLCD19	BPGLCD19	BPFLCD19	BPELCD19	BPDLCD19	BPCLCD19	BPBLCD19	BPALCD19
0x(FF)FF_872A	**LCDWF20**	BPHLCD20	BPGLCD20	BPFLCD20	BPELCD20	BPDLCD20	BPCLCD20	BPBLCD20	BPALCD20
0x(FF)FF_872B	**LCDWF21**	BPHLCD21	BPGLCD21	BPFLCD21	BPELCD21	BPDLCD21	BPCLCD21	BPBLCD21	BPALCD21
0x(FF)FF_872C	**LCDWF22**	BPHLCD22	BPGLCD22	BPFLCD22	BPELCD22	BPDLCD22	BPCLCD22	BPBLCD22	BPALCD22
0x(FF)FF_872D	**LCDWF23**	BPHLCD23	BPGLCD23	BPFLCD23	BPELCD23	BPDLCD23	BPCLCD23	BPBLCD23	BPALCD23
0x(FF)FF_872E	**LCDWF24**	BPHLCD24	BPGLCD24	BPFLCD24	BPELCD24	BPDLCD24	BPCLCD24	BPBLCD24	BPALCD24
0x(FF)FF_872F	**LCDWF25**	BPHLCD25	BPGLCD25	BPFLCD25	BPELCD25	BPDLCD25	BPCLCD25	BPBLCD25	BPALCD25
0x(FF)FF_8730	**LCDWF26**	BPHLCD26	BPGLCD26	BPFLCD26	BPELCD26	BPDLCD26	BPCLCD26	BPBLCD26	BPALCD26
0x(FF)FF_8731	**LCDWF27**	BPHLCD27	BPGLCD27	BPFLCD27	BPELCD27	BPDLCD27	BPCLCD27	BPBLCD27	BPALCD27
0x(FF)FF_8732	**LCDWF28**	BPHLCD28	BPGLCD28	BPFLCD28	BPELCD28	BPDLCD28	BPCLCD28	BPBLCD28	BPALCD28
0x(FF)FF_8733	**LCDWF29**	BPHLCD29	BPGLCD29	BPFLCD29	BPELCD29	BPDLCD29	BPCLCD29	BPBLCD29	BPALCD29
0x(FF)FF_8734	**LCDWF30**	BPHLCD30	BPGLCD30	BPFLCD30	BPELCD30	BPDLCD30	BPCLCD30	BPBLCD30	BPALCD30
0x(FF)FF_8735	**LCDWF31**	BPHLCD31	BPGLCD31	BPFLCD31	BPELCD31	BPDLCD31	BPCLCD31	BPBLCD31	BPALCD31
0x(FF)FF_8736	**LCDWF32**	BPHLCD32	BPGLCD32	BPFLCD32	BPELCD32	BPDLCD32	BPCLCD32	BPBLCD32	BPALCD32
0x(FF)FF_8737	**LCDWF33**	BPHLCD33	BPGLCD33	BPFLCD33	BPELCD33	BPDLCD33	BPCLCD33	BPBLCD33	BPALCD33
0x(FF)FF_8738	**LCDWF34**	BPHLCD34	BPGLCD34	BPFLCD34	BPELCD34	BPDLCD34	BPCLCD34	BPBLCD34	BPALCD34
0x(FF)FF_8739	**LCDWF35**	BPHLCD35	BPGLCD35	BPFLCD35	BPELCD35	BPDLCD35	BPCLCD35	BPBLCD35	BPALCD35
0x(FF)FF_873A	**LCDWF36**	BPHLCD36	BPGLCD36	BPFLCD36	BPELCD36	BPDLCD36	BPCLCD36	BPBLCD36	BPALCD36
0x(FF)FF_873B	**LCDWF37**	BPHLCD37	BPGLCD37	BPFLCD37	BPELCD37	BPDLCD37	BPCLCD37	BPBLCD37	BPALCD37
0x(FF)FF_873C	**LCDWF38**	BPHLCD38	BPGLCD38	BPFLCD38	BPELCD38	BPDLCD38	BPCLCD38	BPBLCD38	BPALCD38
0x(FF)FF_873D	**LCDWF39**	BPHLCD39	BPGLCD39	BPFLCD39	BPELCD39	BPDLCD39	BPCLCD39	BPBLCD39	BPALCD39
0x(FF)FF_873E	**LCDWF40**	BPHLCD40	BPGLCD40	BPFLCD40	BPELCD40	BPDLCD40	BPCLCD40	BPBLCD40	BPALCD40
0x(FF)FF_873F	**LCDWF41**	BPHLCD41	BPGLCD41	BPFLCD41	BPELCD41	BPDLCD41	BPCLCD41	BPBLCD41	BPALCD41

Table 3-3. Detailed Peripheral Memory Map (Sheet 20 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8740	**LCDWF42**	BPHLCD42	BPGLCD42	BPFLCD42	BPELCD42	BPDLCD42	BPCLCD42	BPBLCD42	BPALCD42
0x(FF)FF_8741	**LCDWF43**	BPHLCD43	BPGLCD43	BPFLCD43	BPELCD43	BPDLCD43	BPCLCD43	BPBLCD43	BPALCD43
0x(FF)FF_8742– 0x(FF)FF_877F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8780	**F1CDIV**	FDIVLD	PRDIV8	FDIV					
0x(FF)FF_8781	**F1OPT**	KEYEN		0	0	0	0	SEC	
0x(FF)FF_8782	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_8783	**F1CNFG**	CBEIE	CCIE	KEYACC	0	0	0	0	0
0x(FF)FF_8784	**F1PROT**	FPS							FPOPEN
0x(FF)FF_8785	**F1STAT**	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
0x(FF)FF_8786	**F1CMD**	0	FCMD						
0x(FF)FF_8787– 0x(FF)FF_879F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_87A0	**F2CDIV**	FDIVLD	PRDIV8	FDIV					
0x(FF)FF_87A1	**F2OPT**	KEYEN		0	0	0	0	SEC	
0x(FF)FF_87A2	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_87A3	**F2CNFG**	CBEIE	CCIE	KEYACC	0	0	0	0	0
0x(FF)FF_87A4	**F2PROT**	FPS							FPOPEN
0x(FF)FF_87A5	**F2STAT**	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
0x(FF)FF_87A6	**F2CMD**	0	FCMD						
0x(FF)FF_87A7– 0x(FF)FF_CFFF	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_E000	**PDBSC**	PRESCALER			0	0	0	0	0
0x(FF)FF_E001		IF	CONT	SWTRIG	TRIGSEL			IE	EN
0x(FF)FF_E002	**PDBMOD**	MOD[15:8]							
0x(FF)FF_E003		MOD[7:0]							
0x(FF)FF_E004	**PDBCNT**	COUNT[15:8]							
0x(FF)FF_E005		COUNT[7:0]							
0x(FF)FF_E006	**PDBIDLY**	IDELAY[15:8]							
0x(FF)FF_E007		IDELAY[7:0]							
0x(FF)FF_E008	**PDBCH1CR**	ERRA	ERRB	0	0	0	0	0	0
0x(FF)FF_E009		0	0	AOS		BOS		ENA	ENB
0x(FF)FF_E00A	**PDBCH1DLYA**	DELAY[15:8]							
0x(FF)FF_E00B		DELAY[7:0]							

Table 3-3. Detailed Peripheral Memory Map (Sheet 21 of 22)

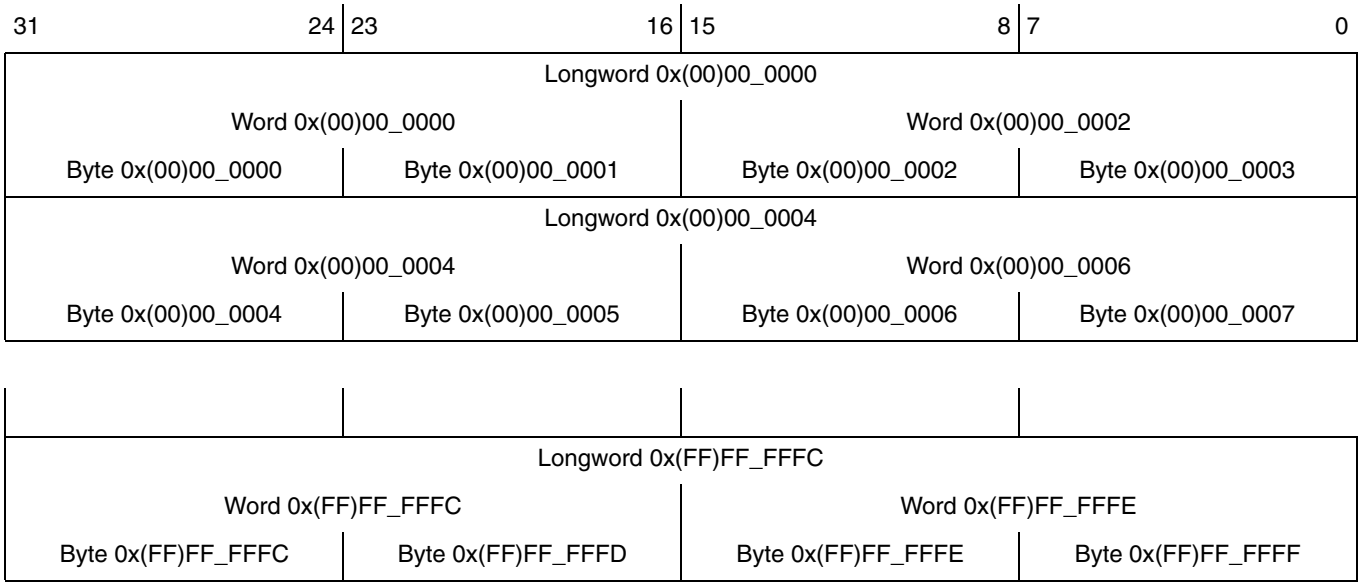
Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_E00C	**PDBCH1DLYB**	DELAY[15:8]							
0x(FF)FF_E00D		DELAY[7:0]							
0x(FF)FF_E00E	Reserved	RESERVED							
0x(FF)FF_E00F		RESERVED							
0x(FF)FF_E010	**PDBCH2CR**	ERRA	ERRB	0	0	0	0	0	0
0x(FF)FF_E011		0	0	AOS		BOS		ENA	ENB
0x(FF)FF_E012	**PDBCH2DLYA**	DELAY[15:8]							
0x(FF)FF_E013		DELAY[7:0]							
0x(FF)FF_E014	**PDBCH2DLYB**	DELAY[15:8]							
0x(FF)FF_E015		DELAY[7:0]							
0x(FF)FF_E016– 0x(FF)FF_E017	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_E018	**PDBCH3CR**	ERRA	ERRB	0	0	0	0	0	0
0x(FF)FF_E019		0	0	AOS		BOS		ENA	ENB
0x(FF)FF_E01A	**PDBCH3DLYA**	DELAY[15:8]							
0x(FF)FF_E01B		DELAY[7:0]							
0x(FF)FF_E01C	**PDBCH3DLYB**	DELAY[15:8]							
0x(FF)FF_E01D		DELAY[7:0]							
0x(FF)FF_E01E– 0x(FF)FF_E01F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_E020	**PDBCH4CR**	ERRA	ERRB	0	0	0	0	0	0
0x(FF)FF_E021		0	0	AOS		BOS		ENA	ENB
0x(FF)FF_E022	**PDBCH4DLYA**	DELAY[15:8]							
0x(FF)FF_E023		DELAY[7:0]							
0x(FF)FF_E024	**PDBCH4DLYB**	DELAY[15:8]							
0x(FF)FF_E025		DELAY[7:0]							
0x(FF)FF_E026– 0x(FF)FF_FFCF	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_FFD0	**INTC_FRC**	0	LVL1	LVL2	LVL3	LVL4	LVL5	LVL6	LVL7
0x(FF)FF_FFD8	**INTC_PL6P7**	0	0	REQN					
0x(FF)FF_FFD9	**INTC_PL6P6**	0	0	REQN					
0x(FF)FF_FFDB	**INTC_WCR**	ENB	0	0	0	0	MASK		
0x(FF)FF_FFDE	**INTC_SFRC**	0	0	SET					
0x(FF)FF_FFDF	**INTC_CFRC**	0	0	CLR					

Table 3-3. Detailed Peripheral Memory Map (Sheet 22 of 22)

Address	Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_FFE0	**INTC_SWIACK**	0	VECN						
0x(FF)FF_FFE4	**INTC_LVL1IACK**	0	VECN						
0x(FF)FF_FFE8	**INTC_LVL2IACK**	0	VECN						
0x(FF)FF_FFEC	**INTC_LVL3IACK**	0	VECN						
0x(FF)FF_FFF0	**INTC_LVL4IACK**	0	VECN						
0x(FF)FF_FFF4	**INTC_LVL5IACK**	0	VECN						
0x(FF)FF_FFF8	**INTC_LVL6IACK**	0	VECN						
0x(FF)FF_FFFC	**INTC_LVL7IACK**	0	VECN						

Recall that ColdFire has a big endian byte addressable memory architecture. The most significant byte of each address is the lowest numbered as shown in Figure 3-2. Multi-byte operands (e.g., 16-bit words and 32-bit longwords) are referenced using an address pointing to the most significant (first) byte.

Figure 3-2. ColdFire Memory Organization



3.2.1 ColdFire Rapid GPIO Memory Map

The rapid GPIO module is mapped into a 16-byte area starting at location 0xC0_0000. Its memory map is shown below in Table 3-4.

Table 3-4. Rapid GPIO Memory Map

Address	Register	Byte 0	Byte 1
0x(00)C0_0000	RGPIO_DIR	DIR[15:8] (Read/Write)	DIR[7:0] (Read/Write)
0x(00)C0_0002	RGPIO_DATA	DATA[15:8] (Read/Write)	DATA[7:0] (Read/Write)
0x(00)C0_0004	RGPIO_ENB	ENB[15:8] (Read/Write)	ENB[7:0] (Read/Write)
0x(00)C0_0006	RGPIO_CLR	CLR[15:8] (Write only)	CLR[7:0] (Write only)
	RGPIO_DATA	DATA[15:8] (Read only)	DATA[7:0] (Read only)
0x(00)C0_0008	RGPIO_DIR	DIR[15:8] (Read only)	DIR[7:0] (Read only)
0x(00)C0_000A	RGPIO_SET	SET[15:8] (Write only)	SET[7:0] (Write only)
	RGPIO_DATA	DATA[15:8] (Read only)	DATA[7:0] (Read only)
0x(00)C0_000C	RGPIO_DIR	DIR[15:8] (Read only)	DIR[7:0] (Read only)
0x(00)C0_000E	RGPIO_TOG	TOG[15:8] (Write only)	TOG[7:0] (Write only)
	RGPIO_DATA	DATA[15:8] (Read only)	DATA[7:0] (Read only)

3.2.2 ColdFire Interrupt Controller Memory Map

The V1 ColdFire interrupt controller (CF1_INTC) register map is sparsely-populated, but retains compatibility with earlier ColdFire interrupt controller definitions. The CF1_INTC occupies the upper 64 bytes of the 4 GB address space and all memory locations are accessed as 8-bit (byte) operands.

3.3 RAM

An MCF51EM256 series microcontroller includes up to 16 KB of static RAM. RAM is most efficiently accessed using the A5-relative addressing mode (address register indirect with displacement mode). Any single bit in this area can be accessed with the bit manipulation instructions (BCLR, BSET,etc.).

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

3.4 Flash Memory

The flash memory is for program storage and read-only data. In-circuit programming allows the operating program to be loaded into the flash memory after final assembly of the application product. It is possible to program the entire array through the single-wire background debug interface. Because no special voltages are needed for flash erase and programming operations, in-application programming is also possible through other software-controlled communication paths.

Flash memory is ideal for single-supply applications allowing for field reprogramming without requiring external high voltage sources for program or erase operations. The flash module includes a memory controller that executes commands to modify flash memory contents.

Array read access time is one bus cycle for bytes, aligned words, and aligned longwords. Multiple accesses are needed for misaligned words and longword operands. For flash memory, an erased bit reads 1 and a programmed bit reads 0. It is not possible to read from a flash block while any command is executing on that specific flash block.

CAUTION

A flash block address must be in the erased state before being programmed. Cumulative programming of bits within a flash block address is not allowed except for status field updates required in EEPROM emulation applications.

Flash memory on this device must be programmed 32-bits at a time with long word address aligned when the low-voltage detect flag (LVDF) in the system power management status and control 1 register (SPMSC1) is clear. If SPMSC1[LVDF] is set, the programming sequence must be modified such that odd and even bytes are written separately. This device's flash memory is organized as two 16-bit wide blocks interleaved to yield a 32-bit data path. When programming flash if LVDF is set, alternate bytes must be set to 0xFF as shown in Table 3-5. Failure to adhere to these guidelines may result in a partially programmed flash array.

Table 3-5. Alternate Bytes Setting

Addresses	Desired Value	Values Programmed
0x00 – 0x03 0x00 – 0x03	0x5555_AAAA	0x55FF_AAFF 0xFF55_FFAA
0x04 – 0x07 0x04 – 0x07	0xCCCC_CCCC	0xCCFF_CCFF 0xFFCC_FFCC

Table 3-5. Alternate Bytes Setting (continued)

Addresses	Desired Value	Values Programmed
0x08 – 0x0B 0x08 – 0x0B	0x1234_5678	0x12FF_56FF 0xFF34_FF78
0x0C – 0x0F 0x0C – 0x0F	0x9ABC_DEF0	0x9AFF_DEFF 0xFFBC_FFF0

3.4.1 Features

Features of the flash memory include:

- The MCF51EM256 series incorporate dual flash controllers, allowing the microcontroller to execute code from one flash array while programming the other.
- Flash size
  - — MCF51EM256: Two flash arrays. Each is 131,072 bytes (128 sectors of 1024 bytes each)
  - — MCF51EM128: Two flash arrays. Each is 65,536 bytes (64 sectors of 1024 bytes each)
- Automated program and erase algorithm
- Fast program and sector erase operation
- Burst program command for faster flash array program times
- Single power supply program and erase
- Command interface for fast program and erase operation
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Flexible block protection (on any 2 KB memory boundary for each flash block)
- Security feature to prevent unauthorized access to on-chip memory and resources
- Auto power-down for low-frequency read accesses

3.4.2 Dual Flash Controllers

The MCF51EM256 and MCF51EM128 devices each separate available flash into two distinct blocks. Each block is controlled by its own, independent, flash controller. Memory mapping of the individual arrays is impacted by the current state of IRTC_CFG_DATA[CFG0] as shown in Table 3-6. Upon initial power-up, IRTC_CFG_DATA[CFG0] = 0, and Array 1 is located at the bottom of the map with Array 2 above it. Setting IRTC_CFG_DATA[CFG0] = 1 reverses the locations of the two arrays in the memory map. The V1 ColdFire core will normally boot from 0x(00)00_0000. The procedure to swap flash arrays must be located in RAM, and must include the following steps:

- Disable flash speculation by the V1 ColdFire CPU by setting CPUCR[FSD] to 1
- Disable all interrupts
- Toggle IRTC_CFG_DATA[CFG0]
- Re-enable interrupts
- If desired, re-enable flash speculation by the V1 ColdFire CPU by re-setting CPUCR[FSD] to 0
- Jump back (using an indirect jump) to main application code residing in the flash memory

Note that swapping flash arrays as discussed above does NOT change the location of the flash controllers in the memory map. Registers in FTSR1 and FTSR2 are fixed in the memory map. Only the flash arrays are impacted by changing IRTC_CFG_DATA[CFG0].

Table 3-6. Flash Array Base Address

IRTC_CFG_DATA[CFG0]	Array	MCF51EM256	MCF51EM128
0 (initial POR value)	FTSR1	0x(00)00_0000	0x(00)00_0000
	FTSR2	0x(00)02_0000	0x(00)01_0000
1	FTSR1		
	FTSR2	0x(00)00_0000	0x(00)00_0000

3.4.3 Register Descriptions

The flash controller (AKA FTSR) contains a set of 16 control and status registers. Detailed descriptions of each register bit are provided in the following sections.

3.4.3.1 Flash Clock Divider Register (FxCDIV)

The FxCDIV register controls the length of timed events in program and erase algorithms executed by the flash memory controller. All bits in the FxCDIV register are readable and writable with restrictions as determined by the value of FDIVLD when writing to the FxCDIV register.

	7	6	5	4	3	2	1	0
R W	FDIVLD	PRDIV8	FDIV					
Reset	0	0	0	0	0	0	0	0

Figure 3-3. Flash Clock Divider Register (FxCDIV)

Table 3-7. FxCDIV Field Descriptions

Field	Description
7 FDIVLD	**Clock Divider Load Control**. When writing to the FxCDIV register for the first time after a reset, the value of the FDIVLD bit written controls the future ability to write to the FxCDIV register: 0 Writing a 0 to FDIVLD locks the FxCDIV register contents; all future writes to FxCDIV are ignored. 1 Writing a 1 to FDIVLD keeps the FxCDIV register writable; next write to FxCDIV is allowed. When reading the FxCDIV register, the value of the FDIVLD bit read indicates the following: 0 FxCDIV register has not been written to since the last reset. 1 FxCDIV register has been written to since the last reset.
6 PRDIV8	**Enable Prescalar by 8.** 0 The bus clock is directly fed into the clock divider. 1 The bus clock is divided by 8 before feeding into the clock divider.
5–0 FDIV	**Clock Divider Bits**. The combination of PRDIV8 and FDIV[5:0] must divide the bus clock down to a frequency of 150 kHz–200 kHz. The minimum divide ratio is 2 (PRDIV8 = 0, FDIV = 0x01) and the maximum divide ratio is 512 (PRDIV8 = 1, FDIV = 0x3F).

3.4.3.2 Flash Options Register (FxOPT and NVOPT)

The FxOPT register holds all bits associated with the security of the MCU and flash module. All bits in the FxOPT register are readable but are not writable.

The FxOPT register is loaded from the flash configuration field during the reset sequence, indicated by F in Figure 3-4.

The security feature in the flash module is described in Section 3.5, “Security”.

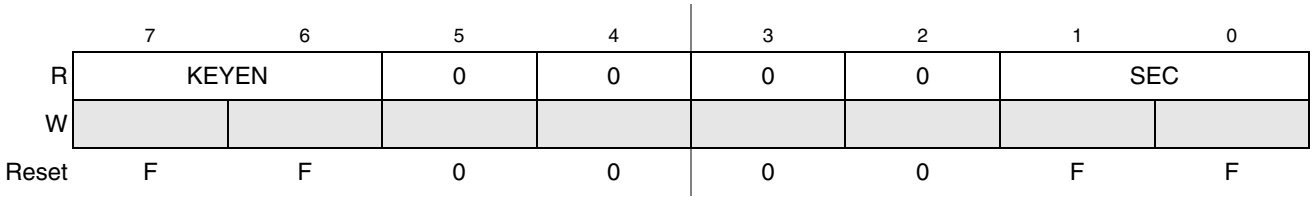


Figure 3-4. Flash Options Register (FxOPT)

Table 3-8. FxOPT Field Descriptions

Field	Description
7–6 KEYEN	**Backdoor Key Security Enable Bits**. The KEYEN[1:0] bits define the enabling of backdoor key access to the flash module. 00 Disabled 01 Disabled (Preferred KEYEN state to disable Backdoor Key Access) 10 Enabled 11 Disabled
5–2	Reserved, should be cleared.
1–0 SEC	**Flash Security Bits**. The SEC[1:0] bits define the security state of the MCU. If the flash module is unsecured using backdoor key access, the SEC[1:0] bits are forced to the unsecured state. 00 Unsecured 01 Unsecured 10 Secured 11 Unsecured

3.4.3.3 Flash Configuration Register (FxCNFG)

The FxCNFG register gates the security backdoor writes.

KEYACC, CBEIE and CCIE are readable and writable while all remaining bits read 0 and are not writable. KEYACC is writable only if KEYEN is set to the enabled state (see Section 3.4.3.2, “Flash Options Register (FxOPT and NVOPT)”).

NOTE

Flash array reads are allowed while KEYACC is set.

	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

Figure 3-5. Flash Configuration Register (FxCNFG)

Table 3-9. FxCNFG 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 3.4.3.5, “Flash Status Register (FxSTAT)”) 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 3.4.3.5, “Flash Status Register (FxSTAT)”) is set.
5 KEYACC	Enable Security Key Writing 0 Writes to the flash block are interpreted as the start of a command write sequence. 1 Writes to the flash block are interpreted as keys to open the backdoor.
4–0	Reserved,

3.4.3.4 Flash Protection Register (FxPROT and NVPROT)

The FxPROT register defines which flash sectors are protected against program or erase operations. FxPROT bits are readable and writable as long as the size of the protected flash memory is being increased. Any write to FxPROT that attempts to decrease the size of the protected flash memory is ignored.

NOTE

MCF51EM256 series have two flash blocks. Each block protection is configured by its respective FxPROT and NVPROT registers. The flash block swap feature is described in Section 3.6, “Flash Module Reserved Memory Locations.” The protection follows the flash block instead of the memory addresses.

During the reset sequence, the FxPROT register is loaded from the flash protection byte in the flash configuration field, indicated by F in Figure 3-6. To change the flash protection loaded during the reset sequence, the flash sector containing the flash configuration field must be unprotected. Then, the flash protection byte must be reprogrammed.

Trying to alter data in any protected area in the flash memory results in a protection violation error and FxSTAT[FPVIOL] is set. The mass erase of the flash array is not possible if any of the flash sectors contained in the flash array are protected.

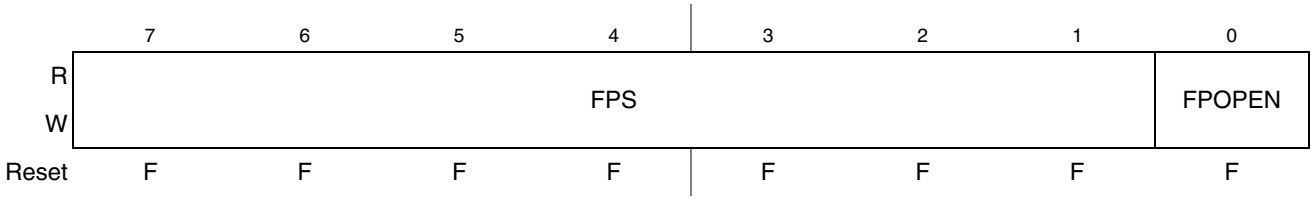


Figure 3-6. Flash Protection Register (FxPROT)

Table 3-10. FxPROT Field Descriptions

Field	Description
7–1 FPS	**Flash Protection Size**. With FPOPEN set, the FPS bits determine the size of the protected flash address range as shown in Table 3-11.
0 FPOPEN	Flash Protection Open 0 Flash array fully protected. 1 Flash array protected address range determined by FPS bits.

Table 3-11. Flash Protection Address Range

FPS	FPOPEN	Protected Address Range Relative to Flash Array Base	Protected Size
—	0	0x0_0000–0x1_FFFF	128 KB
0x00–0x3F	1	0x0_0000–0x1_FFFF	128 KB
0x40		0x0_0000–0x1_F7FF	126 KB
0x41		0x0_0000–0x1_EFFF	124 KB
0x42		0x0_0000–0x1_E7FF	122 KB
0x43		0x0_0000–0x1_DFFF	120 KB
0x44		0x0_0000–0x1_D7FF	118 KB
0x45		0x0_0000–0x1_CFFF	116 KB
0x46		0x0_0000-0x1_C7FF	114 KB

Table 3-11. Flash Protection Address Range (continued)

FPS	FPOPEN	Protected Address Range Relative to Flash Array Base	Protected Size
0x47	1	0x0_0000–0x1_BFFF	112 KB
...		...	...
0x5B		0x0_0000–0x1_1FFF	72 KB
0x5C		0x0_0000–0x1_17FF	70 KB
0x5D		0x0_0000–0x1_0FFF	68 KB
0x5E		0x0_0000–0x1_07FF	66 KB
0x5F		0x0_0000–0x0_FFFF	64 KB
0x60		0x0_0000–0x0_F7FF	62 KB
0x61		0x0_0000–0x0_EFFF	60 KB
0x62		0x0_0000–0x0_E7FF	58 KB
0x63		0x0_0000–0x0_DFFF	56 KB
...		...	...
0x77		0x0_0000–0x0_3FFF	16 KB
0x78		0x0_0000–0x0_37FF	14 KB
0x79		0x0_0000–0x0_2FFF	12 KB
0x7A		0x0_0000–0x0_27FF	10 KB
0x7B		0x0_0000–0x0_1FFF	8 KB
0x7C		0x0_0000–0x0_17FF	6 KB
0x7D		0x0_0000–0x0_0FFF	4 KB
0x7E		0x0_0000–0x0_07FF	2 KB
0x7F		No Protection	0 KB

3.4.3.5 Flash Status Register (FxSTAT)

The FxSTAT register defines the operational status of the flash module. FCBEF, FPVIOL and FACCERR are readable and writable. FBLANK and FCCF are readable but not writable. The remaining bits read 0 and are not writable.

	7	6	5	4	3	2	1	0
R	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
W	w1c[1]		w1c	w1c				
Reset	1	1	0	0	0	0	0	0

[1] wic stands for writing 1 to clear.

Figure 3-7. Flash Status Register (FxSTAT)

Table 3-12. FxSTAT Field Descriptions

Field	Description
7 FCBEF	**Command Buffer Empty Flag.** The FCBEF flag indicates that the command buffer is empty so that a new command write sequence can be started when performing burst programming. Writing a 0 to the FCBEF flag has no effect on FCBEF. Writing a 0 to FCBEF after writing an aligned address to the flash array memory, but before FCBEF is cleared, aborts a command write sequence and causes the FACCERR flag to be set. Writing a 0 to FCBEF outside of a command write sequence does not set the FACCERR flag. Writing a 1 to this bit clears it. 0 Command buffers are full. 1 Command buffers are ready to accept a new command.
6 FCCF	**Command Complete Flag**. The FCCF flag indicates that there are no more commands pending. The FCCF flag is cleared when FCBEF is cleared and sets automatically upon completion of all active and pending commands. The FCCF flag does not set when an active program command completes and a pending burst program command is fetched from the command buffer. Writing to the FCCF flag has no effect on FCCF. 0 Command in progress. 1 All commands are completed.
5 FPVIOL	**Protection Violation Flag**. The FPVIOL 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 FPVIOL flag has no effect on FPVIOL. Writing a 1 to this bit clears it. While FPVIOL 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 FACCERR	**Access Error Flag**. The FACCERR flag indicates an illegal access has occurred to the flash memory caused by either a violation of the command write sequence, issuing an illegal flash command (see Section 3.4.3.6, “Flash Command Register (FxCMD)”), or the execution of a CPU STOP instruction while a command is executing (FCCF = 0). Writing a 0 to the FACCERR flag has no effect on FACCERR. Writing a 1 to this bit clears it. While FACCERR is set, it is not possible to launch a command or start a command write sequence. 0 No access error detected. 1 Access error has occurred.
3	Reserved, must be cleared.
2 FBLANK	**Flag Indicating the Erase Verify Operation Status**. When the FCCF flag is set after completion of an erase verify command, the FBLANK flag indicates the result of the erase verify operation. The FBLANK flag is cleared by the flash module when FCBEF is cleared as part of a new valid command write sequence. Writing to the FBLANK flag has no effect on FBLANK. 0 Flash block verified as not erased. 1 Flash block verified as erased.
1–0	Reserved, should be cleared.

3.4.3.6 Flash Command Register (FxCMD)

The FxCMD register is the flash command register. All FxCMD bits are readable and writable during a command write sequence while bit 7 reads 0 and is not writable.

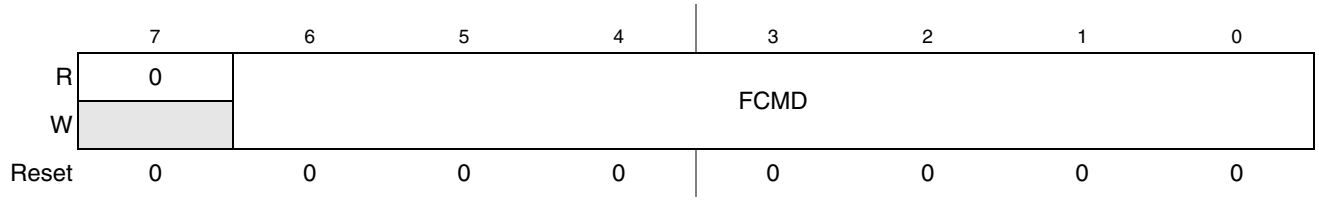


Figure 3-8. Flash Command Register (FxCMD)

Table 3-13. FxCMD Field Descriptions

Field	Description
7	Reserved, must be cleared.
6–0 FCMD	**Flash Command.** Valid flash commands are shown below. Writing any command other than those listed sets the FACCERR flag in the FxSTAT register. 0x05 Erase verify 0x20 Program 0x25 Burst program 0x40 Sector erase 0x41 Mass erase

3.4.4 Flash Command Operations

Flash command operations are used to execute program, erase, and erase verify algorithms described in this section. The program and erase algorithms are controlled by the flash memory controller whose time base, FCLK, is derived from the bus clock via a programmable divider.

The next sections describe:

1. How to write the FxCDIV register to set FCLK
2. Command write sequences to program, erase, and erase verify operations on the flash memory
3. Valid flash commands
4. Effects resulting from illegal flash command write sequences or aborting flash operations

3.4.4.1 Writing the FxCDIV Register

Prior to issuing any flash command after a reset, the user is required to write the FxCDIV register to divide the bus clock down to within the 150 kHz to 200 kHz range.

If we define:

- FCLK as the clock of the flash timing control block
- INT(x) as taking the integer part of x (e.g. INT(4.323) = 4)

then the FxCDIV[PRDIV8, FDIV] bits must be set as described in Figure 3-9.

For example, if the bus clock frequency is 25 MHz, FxCDIV[FDIV] must be set to 0x0F (001111) and the FxCDIV[PRDIV8] bit set to 1. The resulting FCLK frequency is then 195 kHz. In this case, the flash program and erase algorithm timings are increased over the optimum target by:

$$(200 - 195) \div 200 = 3\% \qquad \textit{Eqn. 3-1}$$

CAUTION

Program and erase command execution time will increase proportionally with the period of FCLK. Programming or erasing the flash memory with FCLK < 150 kHz must be avoided. Setting FxCDIV to a value such that FCLK < 150 kHz can destroy the flash memory due to overstress. Setting FxCDIV to a value such that FCLK > 200 kHz can result in incomplete programming or erasure of the flash memory cells.

If the FxCDIV register is written, the FDIVLD bit is set automatically. If the FDIVLD bit is 0, the FxCDIV register has not been written since the last reset. If the FxCDIV register has not been written to, the flash command loaded during a command write sequence will not execute and FxSTAT[FACCERR] will be set.

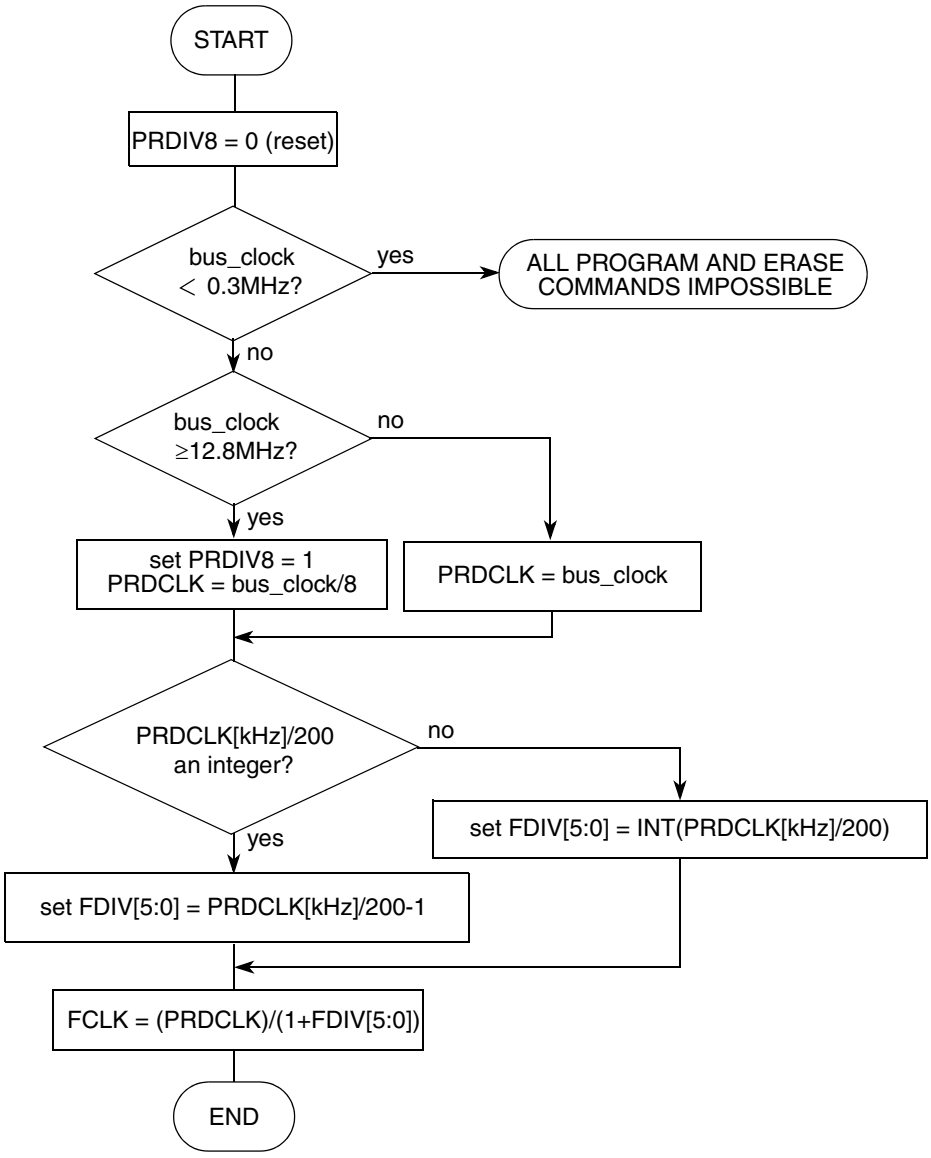


Figure 3-9. Determination Procedure for PRDIV8 and FDIV Bits

3.4.4.2 Command Write Sequence

The flash command controller is used to supervise the command write sequence to execute program, erase, and erase verify algorithms.

Before starting a command write sequence, the FACCERR and FPVIOL flags in the FxSTAT register must be clear and the FCBEF flag must be set (see Section 3.4.3.5).

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 array memory.
2. Write a valid command to the FxCMD register.
3. Clear the FCBEF flag in the FxSTAT register by writing a 1 to FCBEF to launch the command.

Once a command is launched, the completion of the command operation is indicated by the setting of the FCCF flag in the FxSTAT register. The FCCF flag will set upon completion of all active and buffered burst program commands.

3.4.5 Flash Commands

Table 3-14 summarizes the valid flash commands along with the effects of the commands on the flash block.

Table 3-14. Flash Command Description

FCMD	NVM Command	Function on Flash Memory
0x05	Erase Verify	Verify all memory bytes in the flash array memory are erased. If the flash array memory is erased, the FBLANK flag in the FxSTAT register will set upon command completion.
0x20	Program	Program an address in the flash array.
0x25	Burst Program	Program an address in the flash array with the internal address incrementing after the program operation.
0x40	Sector Erase	Erase all memory bytes in a sector of the flash array.
0x41	Mass Erase	Erase all memory bytes in the flash array. A mass erase of the full flash array is only possible when no protection is enabled prior to launching the command.

CAUTION

A flash block address must be in the erased state before being programmed. Cumulative programming of bits within a flash block address is not allowed except for status field updates required in EEPROM emulation applications.

3.4.5.1 Erase Verification Command

The erase verification operation will verify that the entire flash array memory is erased.

An example flow to execute the erase verification operation is shown in Figure 3-10. The erase verification command write sequence is as follows:

1. Write to an aligned flash block address to start the command write sequence for the erase verification command. The address and data written will be ignored.
2. Write the erase verification command, 0x05, to the FxCMD register.
3. Clear the FCBEF flag in the FxSTAT register by writing a 1 to FCBEF to launch the erase verification command.

After launching the erase verification command, the FCCF flag in the FxSTAT register will set after the operation has completed. The number of bus cycles required to execute the erase verification operation is equal to the number of addresses in the flash array memory plus several bus cycles as measured from the time the FCBEF flag is cleared until the FCCF flag is set. Upon completion of the erase verification operation, the FBLANK flag in the FxSTAT register will be set if all addresses in the flash array memory are verified to be erased. If any address in the flash array memory is not erased, the erase verification operation will terminate and the FBLANK flag in the FxSTAT register will remain clear.

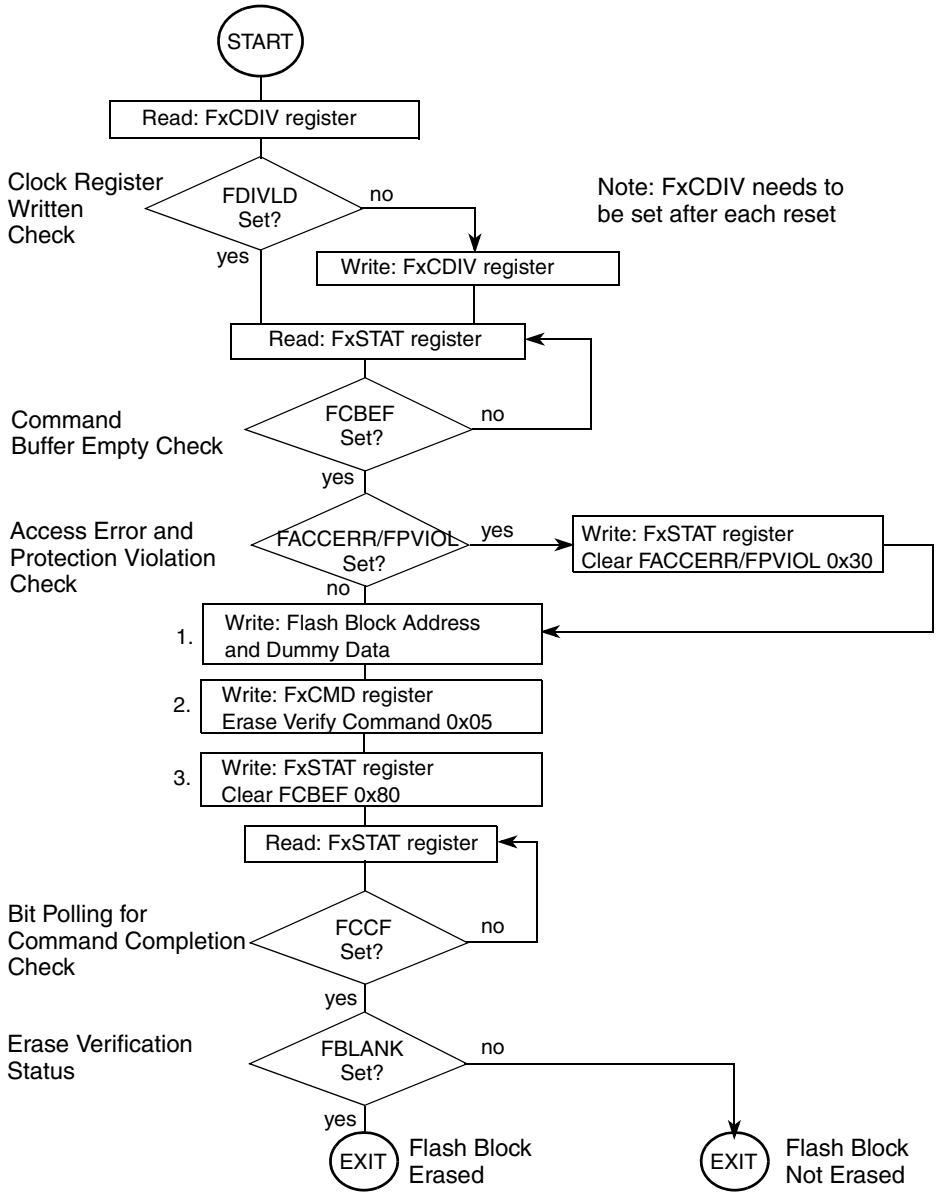


Figure 3-10. Example Erase Verify Command Flow

3.4.5.2 Program Command

The program operation will program a previously erased address in the flash memory using an embedded algorithm.

An example flow to execute the program operation is shown in Figure 3-11. The program command write sequence is as follows:

1. Write to an aligned flash 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 FxCMD register.

3. Clear the FCBEF flag in the FxSTAT register by writing a 1 to FCBEF to launch the program command.

If an address to be programmed is in a protected area of the flash block, the FPVIOL flag in the FxSTAT register will set and the program command will not launch. Once the program command has successfully launched, the FCCF flag in the FxSTAT register will set after the program operation has completed.

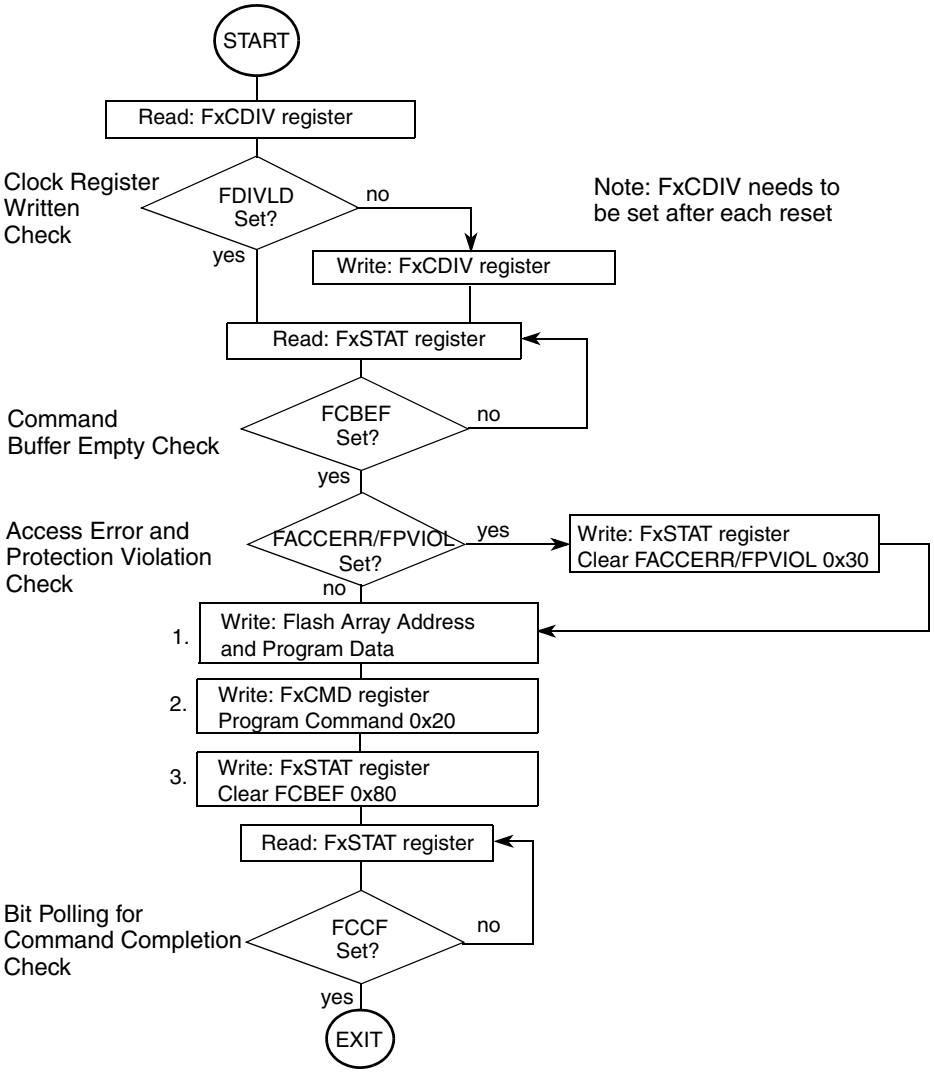


Figure 3-11. Example Program Command Flow

3.4.5.3 Burst Program Command

The burst program operation will program previously erased data in the flash memory using an embedded algorithm.

While burst programming, two internal data registers operate as a buffer and a register (2-stage FIFO) so that a second burst programming command along with the necessary data can be stored to the buffers while the first burst programming command is still in progress. This pipelined operation allows a time optimization when programming more than one consecutive address on a specific row in the flash array as the high voltage generation can be kept active in between two programming commands.

An example flow to execute the burst program operation is shown in Figure 3-12. The burst program command write sequence is as follows:

1. Write to an aligned flash block address to start the command write sequence for the burst program command. The data written will be programmed to the address written.
2. Write the program burst command, 0x25, to the FxCMD register.
3. Clear the FCBEF flag in the FxSTAT register by writing a 1 to FCBEF to launch the program burst command.
4. After the FCBEF flag in the FxSTAT register returns to a 1, repeat steps 1 through 3. The address written is ignored but is incremented internally.

The burst program procedure can be used to program the entire flash memory even while crossing row boundaries within the flash array. If data to be burst programmed falls within a protected area of the flash array, the FPVIOL flag in the FxSTAT register will set and the burst program command will not launch. Once the burst program command has successfully launched, the FCCF flag in the FxSTAT register will set after the burst program operation has completed unless a new burst program command write sequence has been buffered. By executing a new burst program command write sequence on sequential addresses after the FCBEF flag in the FxSTAT register has been set, greater than 50% faster programming time for the entire flash array can be effectively achieved when compared to using the basic program command.

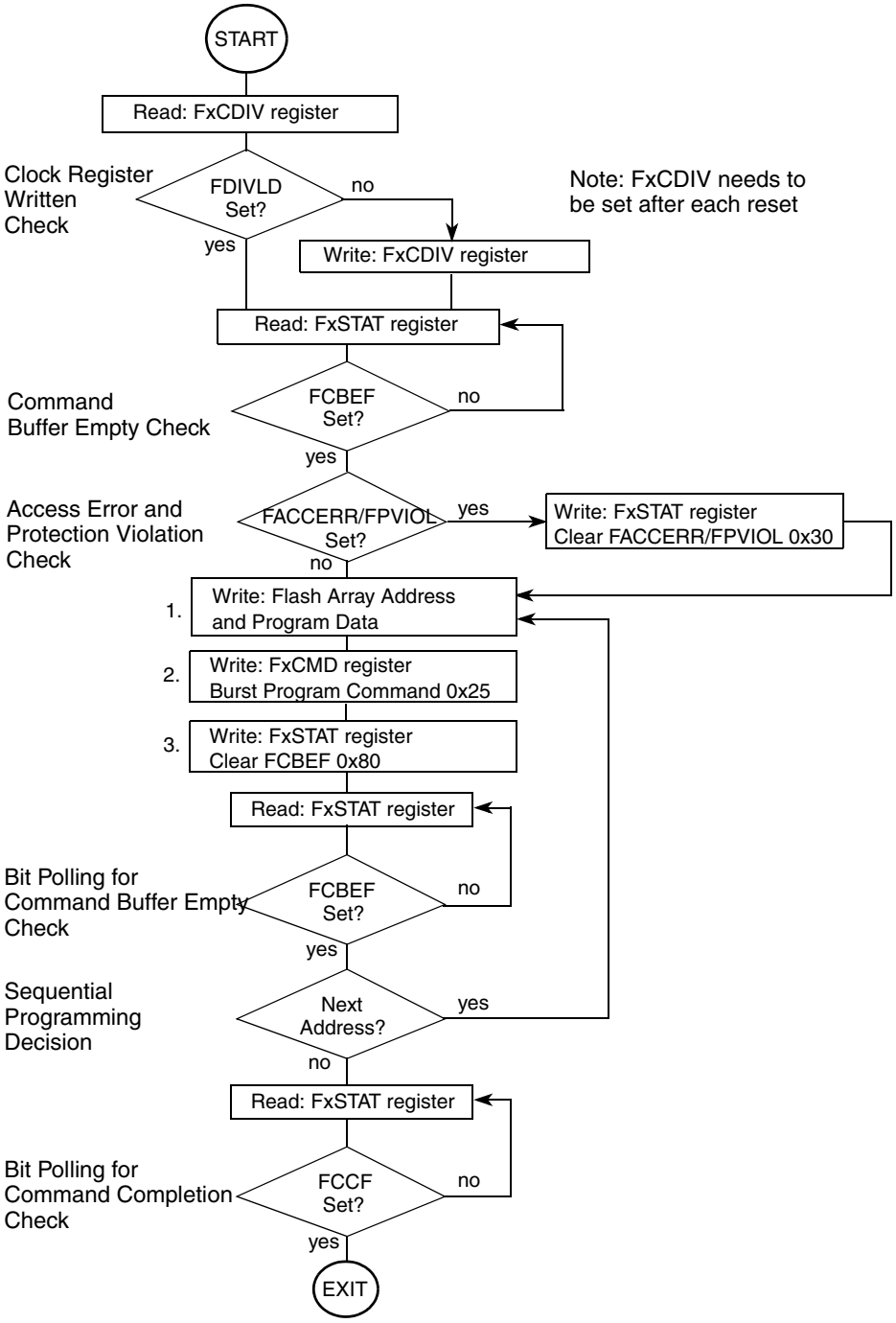


Figure 3-12. Example Burst Program Command Flow

3.4.5.4 Sector Erase Command

The sector erase operation will erase all addresses in a 1 KB sector of flash memory using an embedded algorithm.

An example flow to execute the sector erase operation is shown in Figure 3-13. The sector erase command write sequence is as follows:

1. Write to an aligned flash block address to start the command write sequence for the sector erase command. The flash address written determines the sector to be erased while the data written is ignored.
2. Write the sector erase command, 0x40, to the FxCMD register.
3. Clear the FCBEF flag in the FxSTAT register by writing a 1 to FCBEF to launch the sector erase command.

If a flash sector to be erased is in a protected area of the flash block, the FPVIOL flag in the FxSTAT register will set and the sector erase command will not launch. Once the sector erase command has successfully launched, the FCCF flag in the FxSTAT register will set after the sector erase operation has completed.

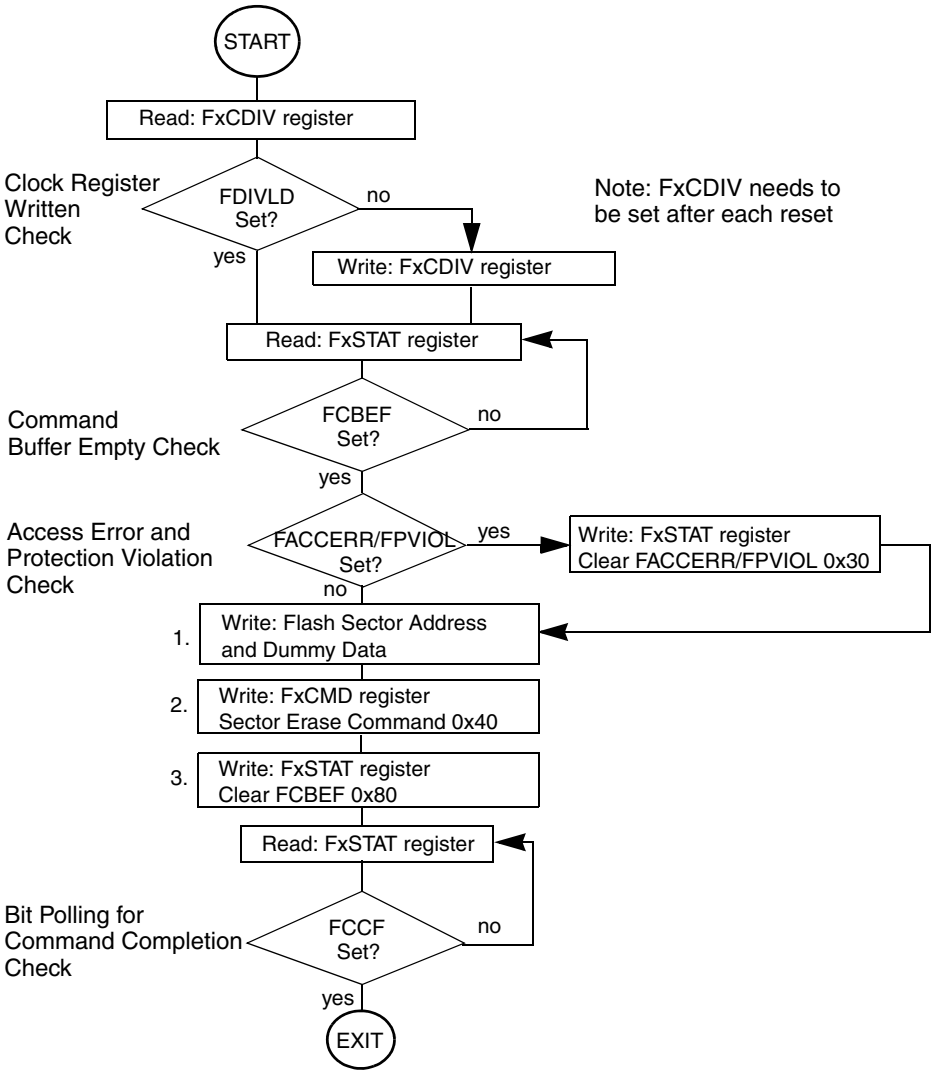


Figure 3-13. Example Sector Erase Command Flow

3.4.5.5 Mass Erase Command

The mass erase operation will erase the entire flash array memory using an embedded algorithm.

An example flow to execute the mass erase operation is shown in Figure 3-14. The mass erase command write sequence is as follows:

1. Write to an aligned flash block 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 FxCMD register.
3. Clear the FCBEF flag in the FxSTAT register by writing a 1 to FCBEF to launch the mass erase command.

If the flash array memory to be mass erased contains any protected area, the FPVIOL flag in the FxSTAT register will set and the mass erase command will not launch. Once the mass erase command has

successfully launched, the FCCF flag in the FxSTAT register will set after the mass erase operation has completed.

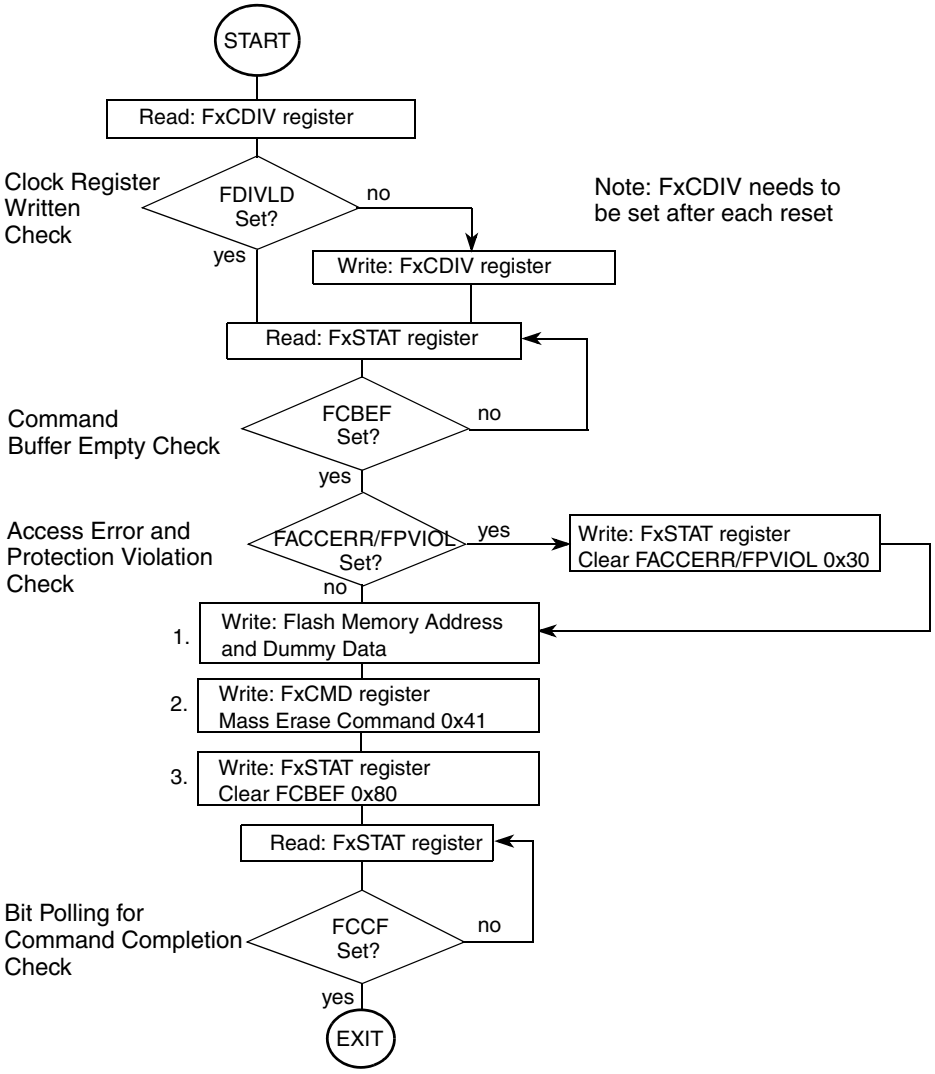


Figure 3-14. Example Mass Erase Command Flow

3.4.6 Illegal Flash Operations

3.4.6.1 Flash Access Violations

The FACCERR 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 FxCDIV register.
2. Writing a byte, word, or misaligned longword to a valid flash address.
3. Writing to any flash register other than FxCMD after writing to a flash address.
4. Writing to a second flash address in the same command write sequence.
5. Writing an invalid command to the FxCMD register unless the address written was in a protected area of the flash array.
6. Writing a command other than burst program while FCBEF is set and FCCF is clear.
7. When security is enabled, writing a command other than erase verify or mass erase to the FxCMD register when the write originates from a non-secure memory location or from the background debug mode.
8. Writing to a flash address after writing to the FxCMD register.
9. Writing to any flash register other than FxSTAT (to clear FCBEF) after writing to the FxCMD register.
10. Writing a 0 to the FCBEF flag in the FxSTAT register to abort a command write sequence.

The FACCERR flag will also be set if the MCU enters stop mode while any command is active (FCCF=0). The operation is aborted immediately and, if burst programming, any pending burst program command is purged (see Section 3.4.7.2, “Stop Mode”).

The FACCERR flag will not be set if any flash register is read during a valid command write sequence.

If the flash memory is read during execution of an algorithm (FCCF = 0), the read operation will return invalid data and the FACCERR flag will not be set.

If the FACCERR flag is set in the FxSTAT register, the user must clear the FACCERR flag before starting another command write sequence (see Section 3.4.3.5, “Flash Status Register (FxSTAT)”).

3.4.6.2 Flash Protection Violations

The FPVIOL flag will be set after the command is written to the FxCMD 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 flash array.
2. Writing the sector erase command if the address written in the command write sequence was in a protected area of the flash array.
3. Writing the mass erase command while any flash protection is enabled.

4. Writing an invalid command if the address written in the command write sequence was in a protected area of the flash array.

If the FPVIOL flag is set in the FxSTAT register, the user must clear the FPVIOL flag before starting another command write sequence (see Section 3.4.3.5, “Flash Status Register (FxSTAT)”).

3.4.7 Operating Modes

3.4.7.1 WAIT Mode

If a command is active (FCCF = 0) when the MCU enters wait mode, the active command and any buffered command will be completed.

3.4.7.2 Stop Mode

If a command is active (FCCF = 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 FCCF and FACCERR flags will be set. If active, the high voltage circuitry to the flash array will immediately be switched off when entering stop mode. Upon exit from stop mode, the FCBEF flag is set and any buffered command will not be launched. The FACCERR flag must be cleared before starting a command write sequence (see Section 3.4.4.2, “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.

Active commands will continue when the MCU enters wait mode. Use of the STOP instruction when SOPT1[WAITE]=1 is acceptable.

3.4.7.3 Background Debug Mode

In background debug mode, the FxPROT register is writable without restrictions. If the MCU is unsecured, then all flash commands listed in Table 3-14 can be executed. If the MCU is secured, only the mass erase and erase verify commands can be executed.

3.5 Security

The MCF51EM256 series microcontrollers include circuitry to prevent unauthorized access to the contents of flash and RAM memory. When security is engaged, BDM access is restricted to the upper byte of the ColdFire CSR, XCSR, and CSR2 registers. RAM, flash memory, peripheral registers and most of the CPU register set are not available via BDM. Programs executing from internal memory have normal access to all microcontroller memory locations and resources.

The MCF51EM256 series devices include two independent flash blocks in support of the robust update feature for on-chip flash. Each flash block has its own set of two security bits as described below. Security must be clear ON BOTH flash blocks in order for the device to be unsecured. This allows the device to

remain secure, even when updating one of the two flash blocks during a code upgrade. The remainder of this section discusses security from the perspective of a single flash block. Routines for clearing flash security must be applied to both blocks before the device can be unsecured.

Security is engaged or disengaged based on the state of two nonvolatile register bits (SEC01, SEC00) in the FxOPT register. During reset, the contents of the nonvolatile location, NVOPT, are copied from flash into the working FxOPT register in high-page register space. A user engages security by programming the NVOPT location which can be done at the same time the flash memory is programmed. The 1:0 stage engages the security and other three combinations disengage security.

Upon exiting reset, the XCSR[25] bit in the ColdFire CPU is initialized to one if the device is secured, zero otherwise.

A user can choose to allow or disallow a security unlocking mechanism through an 8-byte backdoor security key. The security key can be written by the CPU executing from internal memory. It cannot be entered without the cooperation of a secure user program. The procedure for this is detailed in Section 3.5.1, “Unsecuring the MCU using Backdoor Key Access.”

Development tools will unsecure devices via an alternate BDM-based methodology shown in Figure 3-15. Because both $\overline{\text{RESET}}$ and BKGD pins can be reprogrammed via software, a power-on-reset is required to be absolutely certain of obtaining control of the device via BDM, which is a required prerequisite for clearing security. Other methods (outlined in red in Figure 3-15) can also be used, but may not work under all circumstances.

3.5.1 Unsecuring the MCU using Backdoor Key Access

The MCU may be unsecured by using the backdoor key access feature that requires knowledge of the contents of the backdoor keys (see Section 3.6). If the KEYEN[1:0] bits are in the enabled state (see Section 3.4.3.2) 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 backdoor keys are written to the correct addresses in the correct order and the data matches the backdoor keys stored in the flash memory, the MCU is unsecured. The data must be written to the backdoor keys sequentially. Values 0x0000_0000 and 0xFFFF_FFFF are not permitted as backdoor keys. While the KEYACC bit is set, reads of the flash memory return valid 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 3.4.3.2), the MCU can be unsecured by the backdoor key access sequence described below:

1. Set FxCNFG[KEYACC].
2. Execute three NOP instructions to provide time for the backdoor state machine to load the starting address and number of keys required into the flash state machine.
3. Sequentially write the correct longwords to the flash address(es) containing the backdoor keys.
4. 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.

5. If all data written match the backdoor keys, the MCU is unsecured and the SEC[1:0] bits in the NVOPT register are forced to an unsecured state.

The backdoor key access sequence is monitored by an internal security state machine. An illegal operation during the backdoor key access sequence causes the security state machine to lock, leaving the MCU in the secured state. A reset of the MCU causes the security state machine to exit the lock state and allows a new backdoor key access sequence to be attempted. The following operations during the backdoor key access sequence lock the security state machine:

1. If any of the keys written does not match the backdoor keys programmed in the flash array.
2. If the keys are written in the wrong sequence.
3. If any of the keys written are all 0's or all 1's.
4. If the KEYACC bit does not remain set while the keys are written.
5. If any of the keys are written on successive MCU clock cycles.
6. Executing a STOP instruction before all keys have been written.

After the backdoor keys have been correctly matched, the MCU is unsecured. After the MCU is unsecured, the flash security byte can be programmed to the unsecure state, if desired.

In the unsecure state, you have full control of the contents of the backdoor keys by programming the associated addresses in the flash configuration field (see Section 3.6, "Flash Module Reserved Memory Locations").

The security as defined in the flash security byte is not changed by using the backdoor key access sequence to unsecure. The stored backdoor keys 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. The backdoor key access sequence has no effect on the program and erase protections defined in the flash protection register (FxPROT).

It is not possible to unsecure the MCU by using the backdoor key access sequence in background debug mode (BDM) because the MCU does not allow flash array writes in BDM to the flash module.

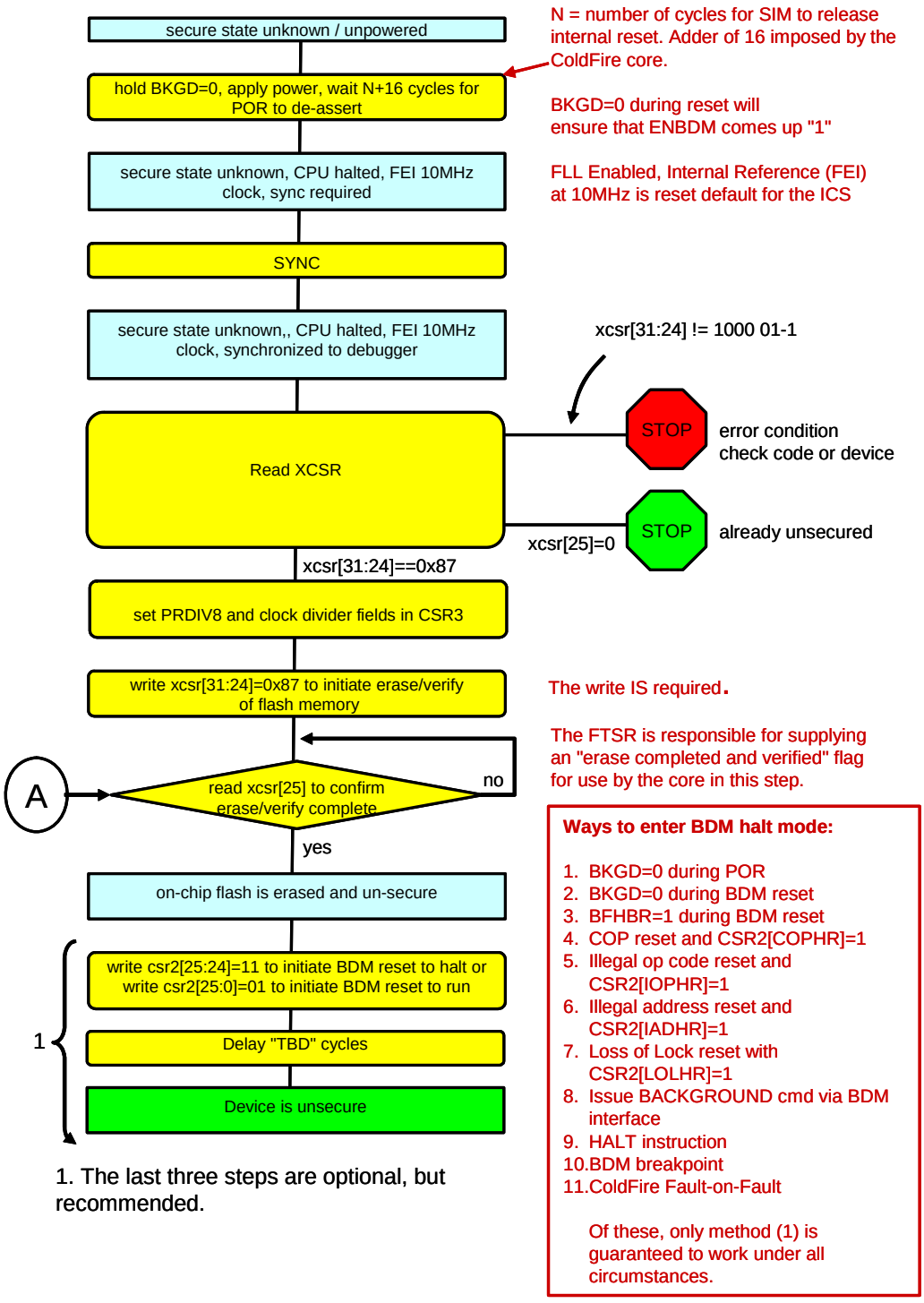


Figure 3-15. Procedure for Clearing Security on MCF51EM256 Series MCUs via the BDM Port 1

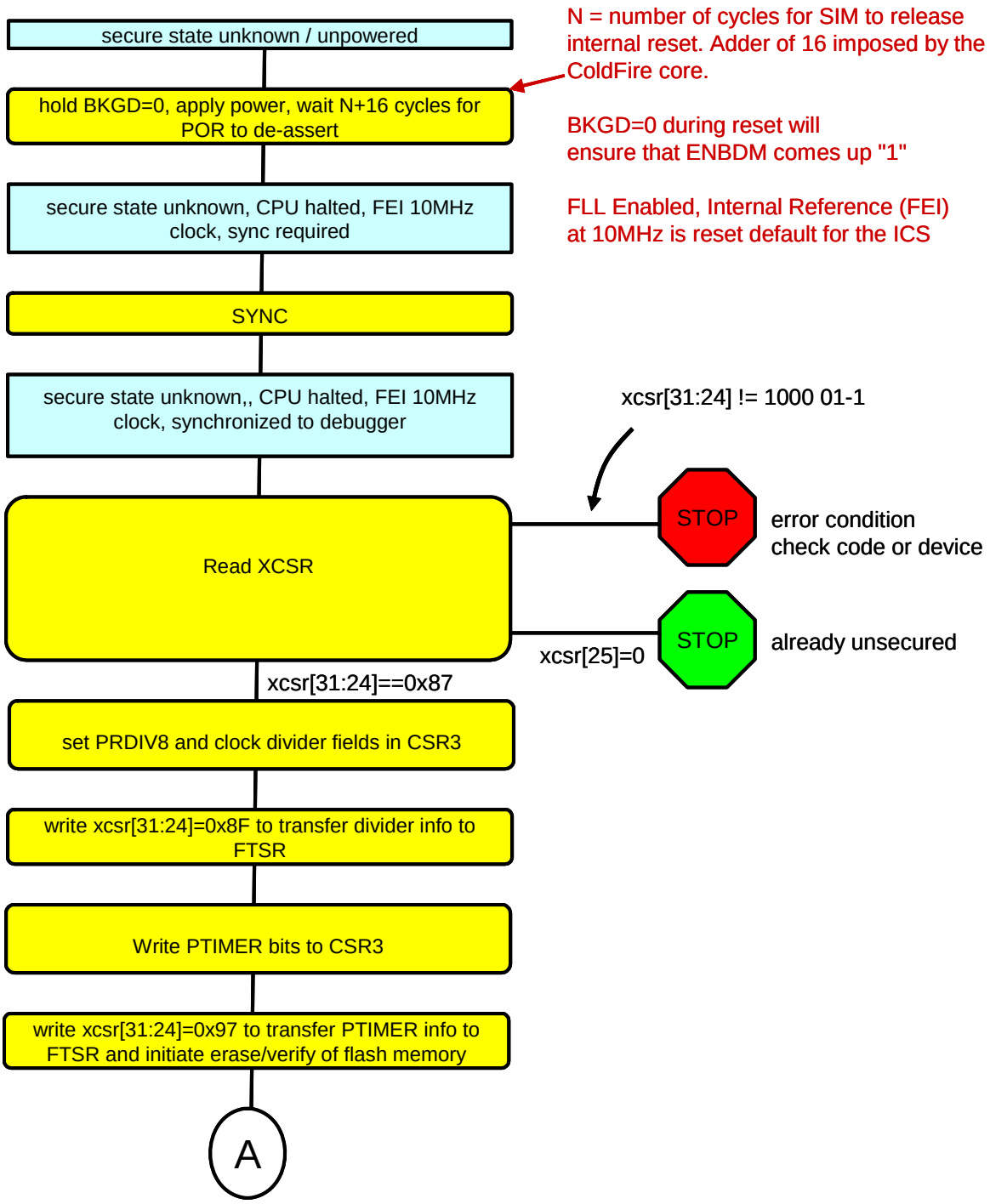


Figure 3-16. Procedure for Clearing Security on MCF51EM256 Series MCUs via the BDM Port 2

3.6 Flash Module Reserved Memory Locations

Several reserved flash memory locations, shown in Table 3-15, are used for storing values used by corresponding peripheral registers. These registers include an 8-byte backdoor key which can be used to

gain access to secure memory resources. During reset events, the contents of the flash protection (NVPROT) byte and flash nonvolatile (NVOPT) byte in the reserved flash memory are transferred into the corresponding FxPROT and FxOPT registers in the high-page register area to control security and block protection options.

Table 3-15 reflects the fact that ColdFire uses big endian addressing. See the ColdFire documentation for further details.

Table 3-15. Reserved Flash Locations[1]

Address	MSB (0x0)	(0x1)	(0x2)	LSB (0x3)
Base+0x03FC	Reserved		FTRIM (bit 0)	TRIM
Base+0x0400	Backdoor comparison key bytes zero thru three			
	byte0	byte1	byte2	byte3
Base+0x0404	Backdoor comparison key bytes four thru seven			
	byte4	byte5	byte6	byte7
Base+0x0408	Reserved			
Base+0x040C	Reserved	NVPROT	Reserved	NVOPT

1 MCF51EM256 series have two sets of nonvolatile registers described in Table 3-15. One for each flash block. The base address of the blocks can be found in Table 3-6. At reset, each nonvolatile register set is loaded in their respective flash registers. When the flash blocks swap, the nonvolatile flash registers also swap following the array.

Chapter 4
Parallel Input/Output Control

4.1 MCF51EM256 Series Functions Overview

NOTE

The configuration of the GPIOs as slew rate control, drive strenght, pullup and input filter affects all modules that share these pins as digital I/O functions.

4.1.1 Summary

This section explains software controls related to parallel input/output (I/O) and pin control. The MCF51EM256 series MCUs have up to six parallel I/O ports which include a total of 48 I/O pins, including one output-only pin. See Chapter 2, “Pins and Connections,” for more information about pin assignments and external hardware considerations of these pins. The MCF51EM256 series MCUs have multiplexed functions in several pins and there is a set of registers to control the multiplex functions in those pins, see Section 4.7, “Pin Mux Controls” for more information.

Table 4-1 summarizes capabilities for the MCF51EM256 series MCUs on a per port basis.

Table 4-1. Functionality on a Per Port Basis

Port Name	Width	GPIO Implementaton[1]	Keyboard Interface	Pin Control[2]
PTA	8	SCT & RGPIO[7:0]	—	PE, SE, DS, IFE
PTB	8	SCT & RGPIO[15:8]	PTB[7:6,3:0] = KBI1[5:0]	PE, SE, DS, IFE
PTC	8	SCT	PTC[1:0] = KBI1[7:6]	PE, SE, DS, IFE
PTD	8	SCT	KBI2[7:0]	PE, SE, DS, IFE
PTE	8	SCT	—	PE, SE, DS, IFE
PTF	8	SCT	—	PE, SE, DS, IFE

1 Possible values are “Standard”, “SCT” and RGPIO. SCT GPIO has all functions associated with Standard GPIO, with the addition of SET, CLEAR and TOGGLE functions. SCT functions are not present on all V1 ColdFire devices. RGPIO standards for “Rapid GPIO”, and is a high data rate implementation bundled with the V1 ColdFire.

2 PE = Pullup Enable, SE = Slew Rate Enable, DS = Drive Strength Control, IFE = Input Filter Enable.

4.1.2 Ports Implemented via Rapid GPIO

In addition to standard GPIO functionality, Ports A and B are implemented using “Rapid GPIO” functions which are integrated as part of the ColdFire core itself to improve edge resolution on those pins. RGPIO

provides functionality similar to standard GPIO equipped with SET/CLEAR/TOGGLE functionality, but at CPU (rather than peripheral bus clock) rates. Standard GPIO are present on these pins for consistancy and code compatibility issues, but RGPIO will normally respond faster.

Port A is associated with RGPIO[7:0]. Table 4-2 shows RGPIO pin mapping to the port I/O pins.

Table 4-2. Port A to RGPIO Mapping

Port pin	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
RGPIO pin	RGPIO7	RGPIO6	RGPIO5	RGPIO4	RGPIO3	RGPIO2	RGPIO1	RGPIO0

Port B is associated with RGPIO[15:8]. Table 4-3 shows RGPIO pin mapping to the port I/O pins.

Table 4-3. Port B to RGPIO Mapping

Port pin	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
RGPIO pin	RGPIO15	RGPIO14	RGPIO13	RGPIO12	RGPIO11	RGPIO10	RGPIO9	RGPIO8

See Chapter 5, “Rapid GPIO (RGPIO),” for additional details.

4.1.3 Keyboard Functions

Several of the Port B and C pins are associated with keyboard interrupt capability. Table 4-4 and Table 4-5 show KBI pin mapping to the port I/O pins.

Table 4-4. KBI1 to GPIO Port Mapping

Port pin	PTC1	PTC0	PTB7	PTB6	PTB3	PTB2	PTB1	PTB0
KBI pin	KBI1P7	KBI1P6	KBI1P5	KBI1P4	KBI1P3	KBI1P2	KBI1P1	KBI1P0

Table 4-5. KBI2 to GPIO Port Mapping

Port pin	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
KBI pin	KBI2P7	KBI2P6	KBI2P5	KBI2P4	KBI2P3	KBI2P2	KBI2P1	KBI2P0

4.1.4 Port Mux Control

Many port pins are shared with on-chip peripherals such as timer systems, communication systems, or keyboard interrupts as shown in Figure 1-1. The user may select which function “owns” a pin via the Port Mux Control (MC) registers. These are defined in detail in Section 4.7, “Pin Mux Controls.”

4.1.5 Special Cases

4.1.5.1 Pullup Resistors

After reset, the shared peripheral functions are normally disabled and the pins are configured as inputs (PT*x*DD[n] = 0). The pin control functions for each pin are configured as follows: slew rate control

disabled (PT*x*SE[n] = 0), low drive strength selected (PT*x*DS[n] = 0), and internal pullups disabled (PT*x*PE[n] = 0). Two exceptions to this are the RESETB and BKGD/MS pins, which have pullups enabled at reset.

NOTE

Not all general-purpose I/O pins are available on all packages. To avoid extra current drain from floating input pins, the user's reset initialization routine in the application program must either enable on-chip pullup devices or change the direction of unconnected pins to outputs so the pins do not float.

4.1.5.2 RESETB

The RESETB is an open drain device, and cannot be programmed to an active high.

4.1.5.3 High Drive Outputs

PTB3 and PTB1 pins have twice the output drive capability of other digital pins, and can source 15 to 20 mA. This allows you to interface SCI1 and SCI2 TX pins (which can mapped to these ports) directly to opto-isolators.

4.1.5.4 Open Drain Outputs

PTE4, PTE5 and PTE6 are open drain devices. These pins do not include active pullup capability (although they do have input pullups when used as input functions). The intention here is to allow the device to interface (via off-chip pullup resistors) to 5 V logic. PTE4 can be programmed to MISO3 or MOSI3. PTE5 can be programmed to SCLK3. PTE6 can be programmed as $\overline{SS3}$ or TX2.

4.2 Pin Controls

This section shows the superset of pin control functions which may be present on V1 ColdFire devices. Some devices may not include all of these controls. See the summary table earlier in the chapter for specific capabilities for your device.

4.2.1 Pin Controls Overview

A set of registers (shown in Figure 4-1) control pullups, slew rate, drive strength and input filter enables for the pins. They may also be used in conjunction with the peripheral functions on these pins. These registers are associated with the parallel I/O ports and Rapid GPIO (RGPIO) ports, but operate independently of both.

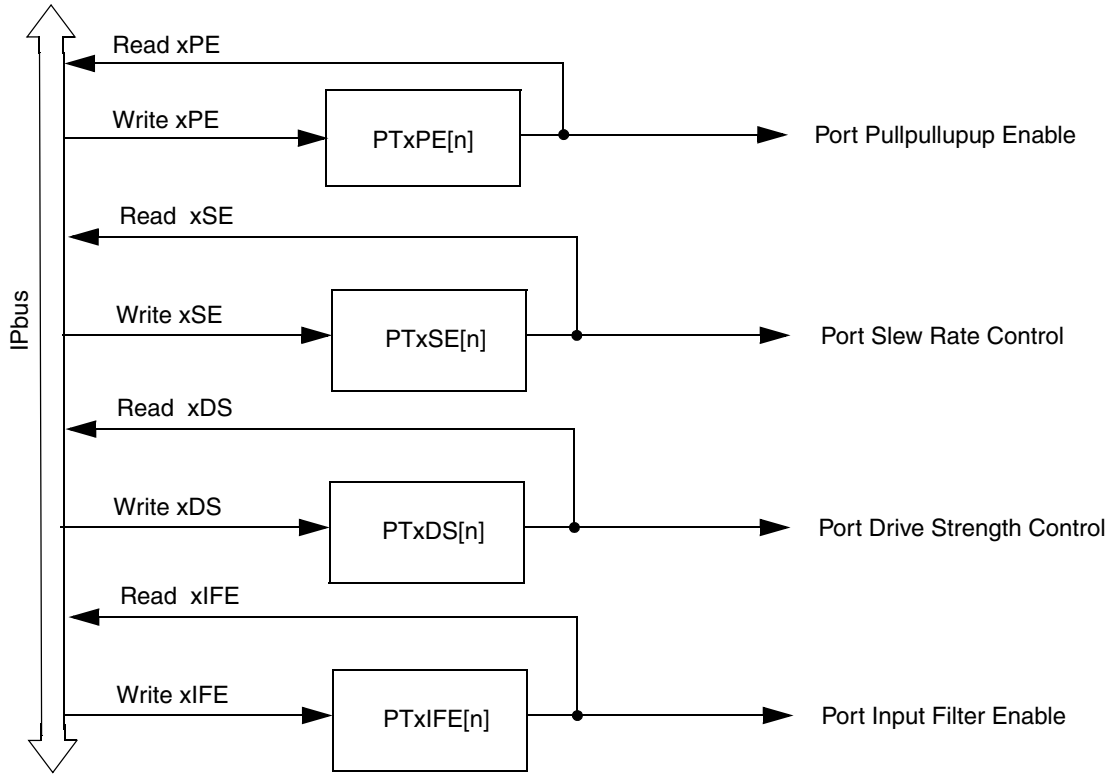


Figure 4-1. Pin Control Logic Block Diagram

4.2.2 Pin Controls Programming Model

These registers control the pullups, slew rate, drive strength, and input filter for all the pins and may be used for the peripheral functions (FEC, Mini-FlexBus, etc.) on these pins.

NOTE

The full complement of pin controls may not be present on all Freescale devices. See the summary table earlier in the chapter to determine which of these are present on your device.

Table 4-6. Register Set Summary

Register	Description	Access
PTxPE	Port x Pull Enable Register	read/write
PTxSE	Port x Slew Rate Enable Register	read/write
PTxDS	Port x Drive Strength Selection Register	read/write
PTxIFE	Port x Input Filter Enable Register	read/write

Refer to tables in Chapter 3, "Memory," for the absolute address assignments for all registers. This section refers to registers and control bits only by their names.

NOTE

A Freescale Semiconductor-provided equate or header file normally is used to translate these names into the appropriate absolute addresses.

4.2.2.1 Port x Pull Enable Register (PTxPE)

An internal pullup device can be enabled for each port pin by setting the corresponding bit in the pullup enable register (PT*x*PE[n]). The pullup device is disabled if the pin is either:

- Configured as an output by the parallel I/O control logic
- Configured as a shared peripheral function
- Controlled by an analog function.
- At reset, except for RESETB and BKGD/MS.

	7	6	5	4	3	2	1	0
R W	PTxPE7	PTxPE6	PTxPE5	PTxPE4	PTxPE3	PTxPE2	PTxPE1	PTxPE0
Reset:	0	0	0	0	0	0	0	0

Figure 4-2. Internal Pull Enable for Port x Register (PTxPE)

Table 4-7. PTxPE Field Descriptions

Field	Description
7–0 PTxPE*n*	**Internal Pull Enable for Port x Bits** — Each of these control bits determines if the internal pullup device is enabled for the associated PTx pin. For Port x pins that are configured as outputs, these bits have no effect and the internal pull devices are disabled. 0 Internal pullup device disabled for Port x bit *n*. 1 Internal pullup device enabled for Port x bit *n*.

4.2.2.2 Port x Slew Rate Enable Register (PTxSE)

Slew rate control can be enabled for each port pin by setting the corresponding bit in the slew rate control register (PT*x*SE[n]). When enabled, slew control limits the rate at which an output can transition in order to reduce EMC emissions. Slew rate control has no effect on pins that are configured as inputs.

NOTE

PTE6, PTE5 and PTE4 do not support the slew rate control, and the register bits PTESE6, PTESE5 and PTESE4 have no effect on these pins.

	7	6	5	4	3	2	1	0
R W	PTxSE7	PTxSE6	PTxSE5	PTxSE4	PTxSE3	PTxSE2	PTxSE1	PTxSE0
Reset:	0	0	0	0	0	0	0	0

Figure 4-3. Slew Rate Enable for Port x Register (PTxSE)

Table 4-8. PTxSE Field Descriptions

Field	Description
7–0 PTxSE*n*	**Output Slew Rate Enable for Port x Bits** — Each of these control bits determines if the output slew rate control is enabled for the associated PTx pin. For Port x pins configured as inputs, these bits have no effect. 0 Output slew rate control disabled for Port x bit *n*. 1 Output slew rate control enabled for Port x bit *n*.

4.2.2.3 Port x Drive Strength Selection Register (PTxDS)

An output pin can be selected to have high output drive strength by setting the corresponding bit in the drive strength select register (PT*x*DS[n]). When high drive is selected, a pin is capable of sourcing and sinking greater current. Even though every I/O pin can be selected as high drive, ensure that the total current source and sink limits for the MCU are not exceeded. Drive strength selection is intended to affect the DC behavior of I/O pins. However, the AC behavior is also affected. High drive allows a pin to drive a greater load with the same switching speed as a low drive enabled pin into a smaller load. Because of this, the EMC emissions may be affected by enabling pins as high drive.

NOTE

PTE6, PTE5 and PTE4 do not support the drive strength control, and the register bits PTEDS6, PTEDS5 and PTEDS4 have no effect on these pins.

	7	6	5	4	3	2	1	0
R W	PTxDS7	PTxDS6	PTxDS5	PTxDS4	PTxDS3	PTxDS2	PTxDS1	PTxDS0
Reset:	0	0	0	0	0	0	0	0

Figure 4-4. Drive Strength Selection for Port x Register (PTxDS)

Table 4-9. PTxDS Field Descriptions

Field	Description
7–0 PTxDS*n*	**Output Drive Strength Selection for Port x Bits** — Each of these control bits selects between low and high output drive for the associated PTx pin. For Port x pins configured as inputs, these bits have no effect. 0 Low output drive strength selected for Port x bit *n*. 1 High output drive strength selected for Port x bit *n*.

4.2.2.4 Port x Input Filter Enable Register (PTxIFE)

The pad cells on this device incorporate optional low pass filters on the digital input functions. These are enabled by setting the appropriate bit in the input filter enable register (PT*x*IFE[n]). When set, a low pass filter (10 MHz to 30 MHz bandwidth) is enabled in the logic input path. When cleared, the filter is bypassed.

	7	6	5	4	3	2	1	0
R W	PTxIFE7	PTxIFE6	PTxIFE5	PTxIFE4	PTxIFE3	PTxIFE2	PTxIFE1	PTxIFE0
Reset:	0	0	0	0	0	0	0	0

Figure 4-5. Port x Input Filter Enable Register (PTxIFE)

Table 4-10. PTxIFE Field Descriptions

Field	Description
7–0 PTxIFE*n*	**Port x Input Filter Enable** — Input low-pass filter enable control bits for PTx pins. 0 Input filter disabled 1 Input filter enabled

4.3 Standard GPIO and GPIO Equipped with Set/Clear/Toggle

4.3.1 GPIO Overview

Reading and writing of parallel I/Os are performed through the port data registers. The direction, either input or output, is controlled through the port data direction registers. The parallel I/O port function for an individual pin is illustrated in the block diagram shown in Figure 4-6.

The data direction control bit (PT*x*DD[n]) determines whether the output buffer for the associated pin is enabled, and also controls the source for port data register reads. The input buffer for the associated pin is always enabled unless the pin is enabled as an analog function or is an output-only pin.

GPIO equipped with set/clear/toggle support single CPU instruction commands for set, clear and toggle of a GPIO output level. Set/clear/toggle functions are not present on all V1 ColdFire I/O ports. See the summary table earlier in this chapter for capabilities of your specific device.

- Writing a one to a bit of any PTxSET register sets that output function to logic one.
- Writing a one to the PTxCLR register sets the output function to zero.
- Writing a one to the PTxTOG register toggles the output value.

When a shared digital function is enabled for a pin, the output buffer is controlled by the shared function. However, the data direction register bit continues to control the source for reads of the port data register.

When a shared analog function is enabled for a pin, the input and output buffers are disabled. A value of 0 is read for any port data bit where the bit is an input (PT*x*DD[n] = 0) and the input buffer is disabled.

Writing to the port data register before changing the direction of a port pin to an output ensures that the pin is not driven momentarily with an old data value in the port data register.

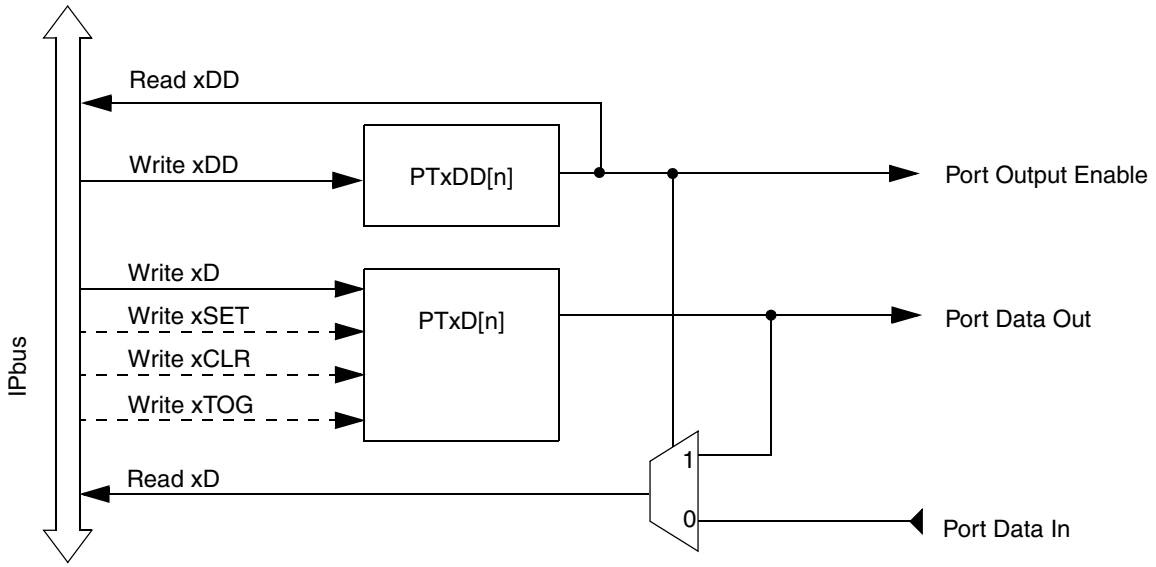


Figure 4-6. GPIO Bit Block Diagram

Pin mux controls leave GPIO input buffers enabled for all digital functions, regardless of mux control selection. This gives you the ability to poll a port to decode a key pressed after a keyboard interrupt is fired. The same function can be used for IRQ pin level detection and debounce software.

4.3.2 GPIO Programming Model

Table 4-11. Register Set Summary

Register	Description	Access
PTxD	PORTx Data Register	read/write
PTxDD	PORTx Data Direction Register	read/write
PTxSET	PORTx SET Register	write
PTxCLR	PORTx CLEAR Register	write
PTxTOG	PORTx TOGGLE Register	write

Refer to tables in Chapter 3, "Memory," for the absolute address assignments for all registers. This section refers to registers and control bits only by their names.

NOTE

A Freescale Semiconductor-provided equate or header file normally is used to translate these names into the appropriate absolute addresses.

4.3.2.1 Port x Data Register (PTxD)

The data register of each port allows software to interact with the pins of the chip. Each bit of each data register controls one pin on the chip. When the port bit is configured as an input (PTxDD[n]=0), a read of

the port returns the logic value of the external pin input for that bit location. When the port bit is configured as an output (PTxDD[n]=1), a read of the port returns the state of the internal data register bit.

Writing to the data register changes the values of all internal data register bits to the value being written regardless of the associated DD bit.

	7	6	5	4	3	2	1	0
R W	PTxD7	PTxD6	PTxD5	PTxD4	PTxD3	PTxD2	PTxD1	PTxD0
Reset:	0	0	0	0	0	0	0	0

Figure 4-7. Port x Data Register (PTxD)

Table 4-12. PTxD Field Descriptions

Field	Description
7–0 PTxD*n*	**Port x Data Register Bits** — For Port x pins that are inputs, reads return the logic level on the pin. For Port x pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For Port x pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTxD to all zeroes. Since reset configures all port pins as high-impedence inputs, these zeroes are not driven onto the pins.

4.3.2.2 Port x Data Direction Register (PTxDD)

The PTxDD registers control the direction of the port x pins.

	7	6	5	4	3	2	1	0
R W	PTxDD7	PTxDD6	PTxDD5	PTxDD4	PTxDD3	PTxDD2	PTxDD1	PTxDD0
Reset:	0	0	0	0	0	0	0	0

Figure 4-8. Port x Data Direction Register (PTxDD)

Table 4-13. PTxDD Field Descriptions

Field	Description
7–0 PTxDD*n*	**Data Direction for Port x Bits** — These read/write bits control the direction of Port x pins and what is read for PTxD reads. 0 Input (output driver disabled) and PTxD reads return the pin value. 1 Output driver enabled for Port x bit *n* and PTxD reads return the contents of PTxD*n*.

4.3.2.3 Port x Set Register (PTxSET)

Setting a PTxSET bit, sets the corresponding bit of the port data register, PTxD. If a PTxSET bit is cleared the corresponding bit in the port data register is unchanged. Writes to PTxSET only cause the PTxD to change. Subsequent writes to PTxD, PTxCLR, or PTxTOG have their own effect in changing the PTxD register.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	PTxSET7	PTxSET6	PTxSET5	PTxSET4	PTxSET3	PTxSET2	PTxSET1	PTxSET0
Reset:	0	0	0	0	0	0	0	0

Figure 4-9. Port x SET Register (PTxSET)

Table 4-14. PTxSET Field Descriptions

Field	Description
7–0 PTxSET*n*	**SET Port x Bits** — These write only bits set a GPIO output. 0 Do not change the state of the GPIO pin 1 Set PTx*n* to logic one

4.3.2.4 Port x Clear Register (PTxCLR)

Setting the PTxCLR bit, clears the corresponding bit of the port data register, PTxD. Clearing a PTxCLR bit has no effect on the PTxD register.Writes to the PTxCLR only cause the PTxD to change. Subsequent writes to PTxD, PTxSET, or PTxTOG have their own effect in changing the PTxD register.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	PTxCLR7	PTxCLR6	PTxCLR5	PTxCLR4	PTxCLR3	PTxCLR2	PTxCLR1	PTxCLR0
Reset:	0	0	0	0	0	0	0	0

Figure 4-10. Port x CLEAR Register (PTxCLR)

Table 4-15. PTxCLR Field Descriptions

Field	Description
7–0 PTxCLR*n*	**CLR Port x Bits** — These write only bits clear a GPIO output. 0 Do not change the state of the GPIO pin 1 Set PTx*n* to logic zero

4.3.2.5 Port x Toggle Register (PTxTOG)

Setting a PTxTOG bit inverts the corresponding bit of the port data register, PTxD. Clearing a PTxTOG bit has no effect on the port data register. Writes to the PTxTOG only cause the PTxD to change. Subsequent writes to PTxD, PTxSET, or PTxCLR have their own effect in changing the PTxD register.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	PTxTOG7	PTxTOG6	PTxTOG5	PTxTOG4	PTxTOG3	PTxTOG2	PTxTOG1	PTxTOG0
Reset:	0	0	0	0	0	0	0	0

Figure 4-11. Port x Toggle Register (PTxTOG)

Table 4-16. PTxTOG Field Descriptions

Field	Description
7–0 PTxTOG*n*	**TOG Port x Bits** — These write only bits toggle a GPIO output. 0 Do not change the state of the GPIO pin 1 Toggle PTxD

4.4 V1 ColdFire Rapid GPIO Functionality

The V1 ColdFire core can perform higher speed I/O via its local bus, which does not have latency penalties associated with the on-chip peripheral bus bridge. The Rapid GPIO module contains separate set/clear/data registers based at address 0x(00)C0_0000. This functionality can be programmed to take priority on ports A and B.

This functionality is further defined in Chapter 5, “Rapid GPIO (RGPIO).”

4.5 Keyboard Interrupts

The block diagram for each keyboard interrupt logic is shown Figure 4-12.

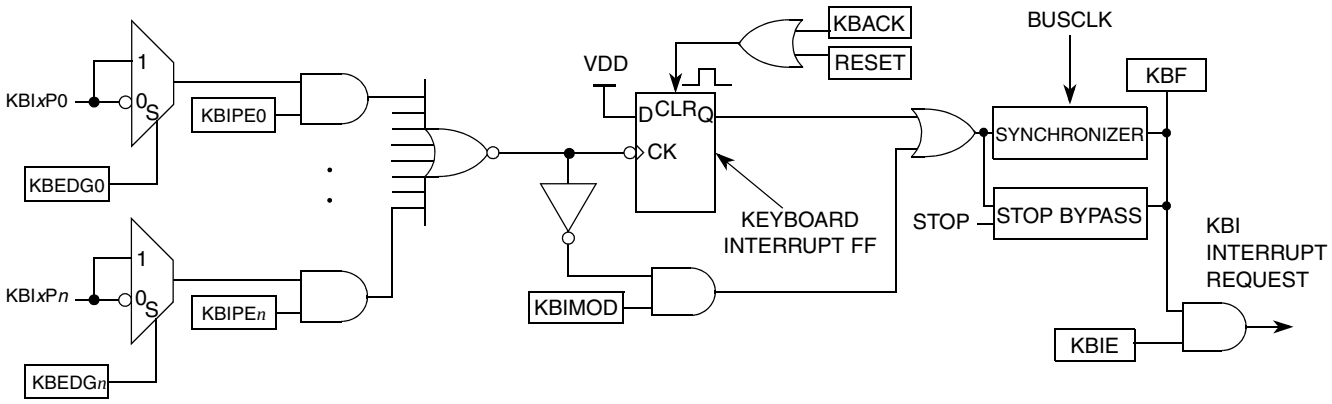


Figure 4-12. Port Interrupt Block Diagram

Writing to KBI*x*PE[KBIPEn] independently enables or disables each port pin. Each port can be configured as edge-sensitive or edge- and level-sensitive based on the KBI*x*SC[KBIMOD] bit. Edge sensitivity can be software programmed to be falling or rising; the level can be either low or high. The polarity of the edge-sensitivity or edge- and level-sensitivity is selected using the KBI*x*ES[KBEDGn].

Synchronous logic is used to detect edges. Prior to detecting an edge, enabled port inputs must be at the deasserted logic level. A falling edge is detected when an enabled port input signal is seen as a logic 1 (the deasserted level) during one bus cycle and then a logic 0 (the asserted level) during the next cycle. A rising edge is detected when the input signal is seen as a logic 0 during one bus cycle and then a logic 1 during the next cycle.

4.5.1 Keyboard Functional Considerations

4.5.1.1 Edge Only Sensitivity

A valid edge on an enabled port pin sets KBI*x*SC[KBF]. If KBI*x*SC[KBIE] is set, an interrupt request is generated to the CPU. Write a 1 to KBI*x*SC[KBACK] to clear KBF.

4.5.1.2 Edge and Level Sensitivity

A valid edge or level on an enabled port pin sets the KBI*x*SC[KBF] bit. If KBI*x*SC[KBIE] is set, an interrupt request is generated to the CPU. Write a 1 to KBI*x*SC[KBACK] to clear KBF, provided all enabled port inputs are at their deasserted levels. KBF remains set if any enabled port pin is asserted while attempting to clear by writing a 1 to KBACK.

4.5.1.3 Pullup/Pulldown Resistors

The keyboard interrupt pins can be configured to use an internal pullup/pulldown resistor using the associated I/O port pullup enable register. If an internal resistor is enabled, the KBI*x*ES register is used to select whether the resistor is a pullup (KBEDG[n] = 0) or a pulldown (KBEDG[n] = 1).

4.5.1.4 Keyboard Interrupt Initialization

When an interrupt pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request during pin interrupt initialization, do the following:

1. Mask interrupts by clearing KBI*x*SC[KBIE].
2. Select the pin polarity by setting the appropriate KBI*x*ES[n] bits.
3. If using internal pullup/pulldown device, configure the associated pull enable bits in PT*x*PE.
4. Enable the interrupt pins by setting the appropriate KBI*x*PE[n] bits.
5. Write 1 to KBI*x*SC[KBACK] to clear any false interrupts.
6. Set KBI*x*SC[KBIE] to enable interrupts.

4.5.2 Keyboard Programming Model

Table 4-17. Register Set Summary

Register	Description	Access
KBIxSC	Keyboard Interrupt Status & Control Register	read/write
KBIxPE	Keyboard Interrupt Pin Select Register	read/write
KBIxES	KBIx Interrupt Edge Select Register	read/write

Refer to tables in Chapter 3, “Memory,” for the absolute address assignments for all registers. This section refers to registers and control bits only by their names.

NOTE

A Freescale Semiconductor-provided equate or header file normally is used to translate these names into the appropriate absolute addresses.

4.5.2.1 KBIx Interrupt Status and Control Register (KBIxSC)

	7	6	5	4	3	2	1	0
R	0	0	0	0	KBF	0	KBIE	KBIMOD
W						KBACK		
Reset:	0	0	0	0	0	0	0	0

Figure 4-13. KBIx Interrupt Status and Control Register (KBIxSC)

Table 4-18. KBIxSC Field Descriptions

Field	Description
7–4	Reserved, must be cleared.
3 KBF	**KBIx Interrupt Flag** — KBF indicates when a KBIx interrupt is detected. Writes have no effect on KBF. 0 No KBIx interrupt detected. 1 KBIx interrupt detected.
2 KBACK	**KBIx Interrupt Acknowledge** — Writing a 1 to KBACK is part of the flag clearing mechanism. KBACK always reads as 0.
1 KBIE	**KBIx Interrupt Enable** — KBIE determines whether a KBIx interrupt is requested. 0 KBIx interrupt request not enabled. 1 KBIx interrupt request enabled.
0 KBIMOD	**KBIx Detection Mode** — KBIMOD (along with the KBIxES bits) controls the detection mode of the KBIx interrupt pins. 0 KBIx pins detect edges only. 1 KBIx pins detect edges and levels.

4.5.2.2 KBIx Interrupt Pin Select Register (KBIxPE)

	7	6	5	4	3	2	1	0
R W	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
Reset:	0	0	0	0	0	0	0	0

Figure 4-14. KBIx Interrupt Pin Select Register (KBIxPE)

Table 4-19. KBIxPE Field Descriptions

Field	Description
7–0 KBIPE*n*	**KBIx Interrupt Pin Selects** — Each of the KBIPE*n* bits enable the corresponding KBIx interrupt pin. 0 Pin not enabled as interrupt. 1 Pin enabled as interrupt.

4.5.2.3 KBIx Interrupt Edge Select Register (KBIxES)

	7	6	5	4	3	2	1	0
R W	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
Reset:	0	0	0	0	0	0	0	0

Figure 4-15. KBIx Edge Select Register (KBIxES)

Table 4-20. KBIxES Field Descriptions

Field	Description
7–0 KBEDG*n*	**KBIx Edge Selects** — Each of the KBEDG*n* bits serves a dual purpose by selecting the polarity of the active interrupt edge as well as selecting a pullup or pulldown device if enabled. 0 A pullup device is connected to the associated pin and detects falling edge/low level for interrupt generation. 1 A pulldown device is connected to the associated pin and detects rising edge/high level for interrupt generation.

4.6 Pin Behavior in Stop Modes

Pin behavior following execution of a STOP instruction depends on the stop mode that is entered. An explanation of pin behavior for the various stop modes follows:

- Stop2 mode is a partial power-down mode, whereby I/O latches are maintained in their state as before the STOP instruction was executed (port states are lost and need to be restored upon exiting stop2). CPU register status and the state of I/O registers should be saved in RAM before the executing the STOP instruction to place the MCU in stop2 mode.
  After recovery from stop2 mode, before accessing any I/O, examine the state of the SPMSC2[PPDF] bit.
  - If the PPDF bit is cleared, I/O must be initialized as if a power-on-reset had occurred.
  - If the PPDF bit is set, I/O register states should be restored from the values saved in RAM before the STOP instruction was executed and peripherals may require initialization or restoration to their pre-stop condition.

  Then write a 1 to the SPMSC2[PPDACK] bit. Access to I/O is now permitted again in the user application program.
- In stop3 and stop4 modes, all I/O is maintained because internal logic circuity stays powered. After recovery, normal I/O function is available to the user.

4.7 Pin Mux Controls

Package pins on the MCF51EM256 series can be programmed for up to four different functions using the Pin Mux Control Registers. Controls are organized by GPIO Port. Each GPIO port has two mux control registers, comprised 2 bits per package pin. All pin mux registers default to value 0x00 at reset. Alternate values are defined in the remainder of this section. Please note that the encoding for each function matches the column number in which that function occurs in Table 2-2. That is, default functions are assigned value 0x00, ALT1 functions 0x01 and so forth.

The RGPIO_ENB register is used to select between GPIO and RGPIO functions on pins that have both.

Pin mux controls will leave GPIO input buffers enabled for all digital functions, regardless of mux control selection. This gives the user the ability to poll a port to decode a key pressed after a keyboard interrupt is asserted. The same function can be used for IRQ pin level detection and debounce software.

Table 4-21. Pin Mux Control Registers

Address	Register	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8200	PTAPF1	A7		A6		A5		A4	
0x(FF)FF_8201	PTAPF2	A3		A2		A1		A0	
0x(FF)FF_8202	PTBPF1	B7		B6		B5		B4	
0x(FF)FF_8203	PTBPF2	B3		B2		B1		B0	
0x(FF)FF_8204	PTCPF1	C7		C6		C5		C4	
0x(FF)FF_8205	PTCPF2	C3		C2		C1		C0	
0x(FF)FF_8206	PTDPF1	D7		D6		D5		D4	
0x(FF)FF_8207	PTDPF2	D3		D2		D1		D0	
0x(FF)FF_8208	PTEPF1	E7		E6		E5		E4	
0x(FF)FF_8209	PTEPF2	E3		E2		E1		E0	
0x(FF)FF_820A	PTFPF1	F7		F6		F5		F4	
0x(FF)FF_820B	PTFPF2	F3		F2		F1		F0	
0x(FF)FF_820C	LCDPF1	LCD9		LCD8		LCD7		LCD6	
0x(FF)FF_820D	LCDPF2	0	0	0	0	0	0	LCD35	

NOTE

Configuring the mux control registers to the ADC pins disables the digital function of the pin to ensure proper ADC conversion.

4.7.1 Port A Pin Function Register 1 (PTAPF1)

	7	6	5	4	3	2	1	0
R W	A7		A6		A5		A4	
Reset	0	0	0	0	0	0	0	0

Figure 4-16. Port A Pin Function Register 1 (PTAPF1)

Table 4-22. PTAPF1 Field Descriptions

Field	Description
7–6 A7	**Port A7 Pin Mux Controls.** 00 PTA7/RGPIO7 01 TPMCLK 10 PRACMP1P2 11 AD13
5–4 A6	**Port A6 Pin Mux Controls.** 00 PTA6/RGPIO6 01 TPMCH1 10 Reserved 11 AD12
3–2 A5	**Port A5 Pin Mux Controls.** 00 PTA5/RGPIO5 01 TPMCH0 10 Reserved 11 AD11
1–0 A4	**Port A4 Pin Mux Controls.** 00 PTA4/RGPIO4 01 $\overline{SS1}$ 10 Reserved 11 Reserved

4.7.2 Port A Pin Function Register 2 (PTAPF2)

	7	6	5	4	3	2	1	0
R W	A3		A2		A1		A0	
Reset	0	0	0	0	0	0	0	0

Figure 4-17. Port A Pin Function Register 2 (PTAPF2)

Table 4-23. PTAPF2 Field Descriptions

Field	Description
7–6 A3	**Port A3 Pin Mux Controls.** 00 PTA3/RGPIO3 01 SCLK1 10 Reserved 11 Reserved
5–4 A2	**Port A2 Pin Mux Controls.** 00 PTA2/RGPIO2 01 MISO1 10 Reserved 11 AD10
3–2 A1	**Port A1 Pin Mux Controls.** 00 PTA1/RGPIO1 01 MOSI1 10 Reserved 11 Reserved
1–0 A0	**Port A0 Pin Mux Controls.** 00 PTA0/RGPIO0 01 IRQ 10 CLKOUT 11 Reserved

4.7.3 Port B Pin Function Register 1 (PTBPF1)

	7	6	5	4	3	2	1	0
R W	B7		B6		B5		B4	
Reset	0	0	0	0	0	0	0	0

Figure 4-18. Port B Pin Function Register 1 (PTBPF1)

Table 4-24. PTBPF1 Field Descriptions

Field	Description
7–6 B7	**Port B7 Pin Mux Controls.** 00 PTB7/RGPIO15 01 KBI1P5 10 TMRCLK2 11 AD15
5–4 B6	**Port B6 Pin Mux Controls.** 00 PTB6/RGPIO14 01 KBI1P4 10 TMRCLK1 11 AD14
3–2 B5	**Port B5 Pin Mux Controls.** 00 PTB5/RGPIO13 01 SDA 10 PRACMP2P3 11 Reserved
1–0 B4	**Port B4 Pin Mux Controls.** 00 PTB4/RGPIO12 01 SCL 10 PRACMP2P2 11 Reserved

4.7.4 Port B Pin Function Register 2 (PTBPF2)

	7	6	5	4	3	2	1	0
R W	B3		B2		B1		B0	
Reset	0	0	0	0	0	0	0	0

Figure 4-19. Port B Pin Function Register 2 (PTBPF2)

Table 4-25. PTBPF2 Field Descriptions

Field	Description
7–6 B3	**Port B3 Pin Mux Controls.** 00 PTB3/RGPIO11 01 KBI1P3 10 PRACMP2P1 11 TX1
5–4 B2	**Port B2 Pin Mux Controls.** 00 PTB2/RGPIO10 01 KBI1P2 10 PRACMP1P0 11 RX1
3–2 B1	**Port B1 Pin Mux Controls.** 00 PTB1/RGPIO9 01 KBI1P1 10 $\overline{SS3}$ 11 TX2
1–0 B0	**Port B0 Pin Mux Controls.** 00 PTB0/RGPIO8 01 KBI1P0 10 PRACMP2P0 11 RX2

4.7.5 Port C Pin Function Register 1 (PTCPF1)

	7	6	5	4	3	2	1	0
R W	C7		C6		C5		C4	
Reset	0	0	0	0	0	0	0	0

Figure 4-20. Port C Pin Function Register 1 (PTCPF1)

Table 4-26. PTCPF1 Field Descriptions

Field	Description
7–6 C7	**Port C7 Pin Mux Controls.** 00 PTC7 01 LCD4 10 Reserved 11 Reserved
5–4 C6	**Port C6 Pin Mux Controls.** 00 PTC6 01 LCD3 10 Reserved 11 PRACMP2P4
3–2 C5	**Port C5 Pin Mux Controls.** 00 PTC5 01 LCD2 10 Reserved 11 PRACMP1P4
1–0 C4	**Port C4 Pin Mux Controls.** 00 PTC4 01 LCD1 10 PRACMP2O 11 Reserved

4.7.6 Port C Pin Function Register 2 (PTCPF2)

	7	6	5	4	3	2	1	0
R W	C3		C2		C1		C0	
Reset	0	0	0	0	0	0	0	0

Figure 4-21. Port C Pin Function Register 2 (PTCPF2)

Table 4-27. PTCPF2 Field Descriptions

Field	Description
7–6 C3	**Port C3 Pin Mux Controls.** 00 PTC3 01 LCD0 10 PRACMP1O 11 Reserved
5–4 C2	**Port C2 Pin Mux Controls.** 00 BKGD/MS 01 PTC2 10 Reserved 11 Reserved
3–2 C1	**Port C1 Pin Mux Controls.** 00 PTC1 01 KBI1P7 10 XTAL2 11 TX3
1–0 C0	**Port C0 Pin Mux Controls.** 00 PTC0 01 KBI1P6 10 EXTAL2 11 RX3

4.7.7 Port D Pin Function Register 1 (PTDPF1)

	7	6	5	4	3	2	1	0
R W	D7		D6		D5		D4	
Reset	0	0	0	0	0	0	0	0

Figure 4-22. Port D Pin Function Register 1 (PTDPF1)

Table 4-28. PTDPF1 Field Descriptions

Field	Description
7–6 D7	**Port D7 Pin Mux Controls.** 00 PTD7 01 LCD34 10 KBI2P7 11 Reserved
5–4 D6	**Port D6 Pin Mux Controls.** 00 PTD6 01 LCD33 10 KBI2P6 11 Reserved
3–2 D5	**Port D5 Pin Mux Controls.** 00 PTD5 01 LCD32 10 KBI2P5 11 TMRCLK2
1–0 D4	**Port D4 Pin Mux Controls.** 00 PTD4 01 LCD31 10 KBI2P4 11 TMRCLK1

4.7.8 Port D Pin Function Register 2 (PTDPF2)

	7	6	5	4	3	2	1	0
R W	D3		D2		D1		D0	
Reset	0	0	0	0	0	0	0	0

Figure 4-23. Port D Pin Function Register 2 (PTDPF2)

Table 4-29. PTDPF2 Field Descriptions

Field	Description
7–6 D3	**Port D3 Pin Mux Controls.** 00 PTD3 01 LCD30 10 KBI2P3 11 Reserved
5–4 D2	**Port D2 Pin Mux Controls.** 00 PTD2 01 LCD29 10 KBI2P2 11 Reserved
3–2 D1	**Port D1 Pin Mux Controls.** 00 PTD1 01 LCD28 10 KBI2P1 11 Reserved
1–0 D0	**Port D0 Pin Mux Controls.** 00 PTD0 01 LCD27 10 KBI2P0 11 Reserved

4.7.9 Port E Pin Function Register 1 (PTEPF1)

	7	6	5	4	3	2	1	0
R W	E7		E6		E5		E4	
Reset	0	0	0	0	0	0	0	0

Figure 4-24. Port E Pin Function Register 1 (PTEPF1)

Table 4-30. PTEPF1 Field Descriptions

Field	Description
7–6 E7	**Port E7 Pin Mux Controls.** 00 PTE7 01 LCD5 10 Reserved 11 Reserved
5–4 E6	**Port E6 Pin Mux Controls.** 00 PTE6 01 $\overline{SS3}$ 10 $\overline{SS3}$ 11 TX2
3–2 E5	**Port E5 Pin Mux Controls.** 00 PTE5 01 SCLK3 10 SCLK3 11 Reserved
1–0 E4	**Port E4 Pin Mux Controls.** 00 PTE4 01 MISO3 10 MOSI3 11 Reserved

4.7.10 Port E Pin Function Register 2(PTEPF2)

	7	6	5	4	3	2	1	0
R W	E3		E2		E1		E0	
Reset	0	0	0	0	0	0	0	0

Figure 4-25. Port E Pin Function Register 2 (PTEPF2)

Table 4-31. PTEPF2 Field Descriptions

Field	Description
7–6 E3	**Port E3 Pin Mux Controls.** 00 PTE3 01 MOSI3 10 MISO3 11 Reserved
5–4 E2	**Port E2 Pin Mux Controls.** 00 PTE2 01 Reserved 10 PRACMP1P1 11 Reserved
3–2 E1	**Port E1 Pin Mux Controls.** 00 PTE1 01 Reserved 10 PRACMP1P3 11 AD9
1–0 E0	**Port E0 Pin Mux Controls.** 00 PTE0 01 Reserved 10 PRACMP1O 11 AD8

4.7.11 Port F Pin Function Register 1 (PTFPF1)

	7	6	5	4	3	2	1	0
R W	F7		F6		F5		F4	
Reset	0	0	0	0	0	0	0	0

Figure 4-26. Port F Pin Function Register 1 (PTFPF1)

Table 4-32. PTFPF1 Field Descriptions

Field	Description
7–6 F7	**Port F7 Pin Mux Controls.** 00 PTF7 01 LCD43 10 Reserved 11 AD19
5–4 F6	**Port F6 Pin Mux Controls.** 00 PTF6 01 LCD42 10 Reserved 11 AD18
3–2 F5	**Port F5 Pin Mux Controls.** 00 PTF5 01 LCD41 10 Reserved 11 AD17
1–0 F4	**Port F4 Pin Mux Controls.** 00 PTF4 01 Reserved 10 LCD40 11 AD16

4.7.12 Port F Pin Function Register 2 (PTFPF2)

	7	6	5	4	3	2	1	0
R W	F3		F2		F1		F0	
Reset	0	0	0	0	0	0	0	0

Figure 4-27. Port F Pin Function Register 2 (PTFPF2)

Table 4-33. PTFPF2 Field Descriptions

Field	Description
7–6 F3	**Port F3 Pin Mux Controls.** 00 PTF3 01 LCD39 10 Reserved 11 TX3
5–4 F2	**Port F2 Pin Mux Controls.** 00 PTF2 01 LCD38 10 Reserved 11 RX3
3–2 F1	**Port F1 Pin Mux Controls.** 00 PTF1 01 LCD37 10 Reserved 11 EXTRIG
1–0 F0	**Port F0 Pin Mux Controls.** 00 PTF0 01 LCD36 10 Reserved 11 Reserved

4.7.13 LCD Pin Function Register 1 (LCDPF1)

	7	6	5	4	3	2	1	0
R W	LCD9		LCD8		LCD7		LCD6	
Reset	0	0	0	0	0	0	0	0

Figure 4-28. LCD Pin Function Register 1 (LCDPF1)

Table 4-34. LCDPF1 Field Descriptions

Field	Description
7–6 LCD9	**LCD9 Pin Mux Controls.** 00 LCD9 01 $\overline{SS2}$ 10 Reserved 11 Reserved
5–4 LCD8	**LCD8 Pin Mux Controls.** 00 LCD8 01 SCLK2 10 Reserved 11 Reserved
3–2 LCD7	**LCD7 Pin Mux Controls.** 00 LCD7 01 MISO2 10 Reserved 11 Reserved
1–0 LCD6	**LCD6 Pin Mux Controls.** 00 LCD6 01 MOSI2 10 Reserved 11 Reserved

4.7.14 LCD Pin Function Register 2 (LCDPF2)

	7	6	5	4	3	2	1	0
R W	0		0		0		LCD35	
Reset	0	0	0	0	0	0	0	0

Figure 4-29. LCD Pin Function Register 2 (LCDPF2)

Table 4-35. LCDPF2 Field Descriptions

Field	Description
1–0 LCD35	**Port LCD35 Pin Mux Controls.** 00 LCD35 01 CLKOUT 10 Reserved 11 Reserved

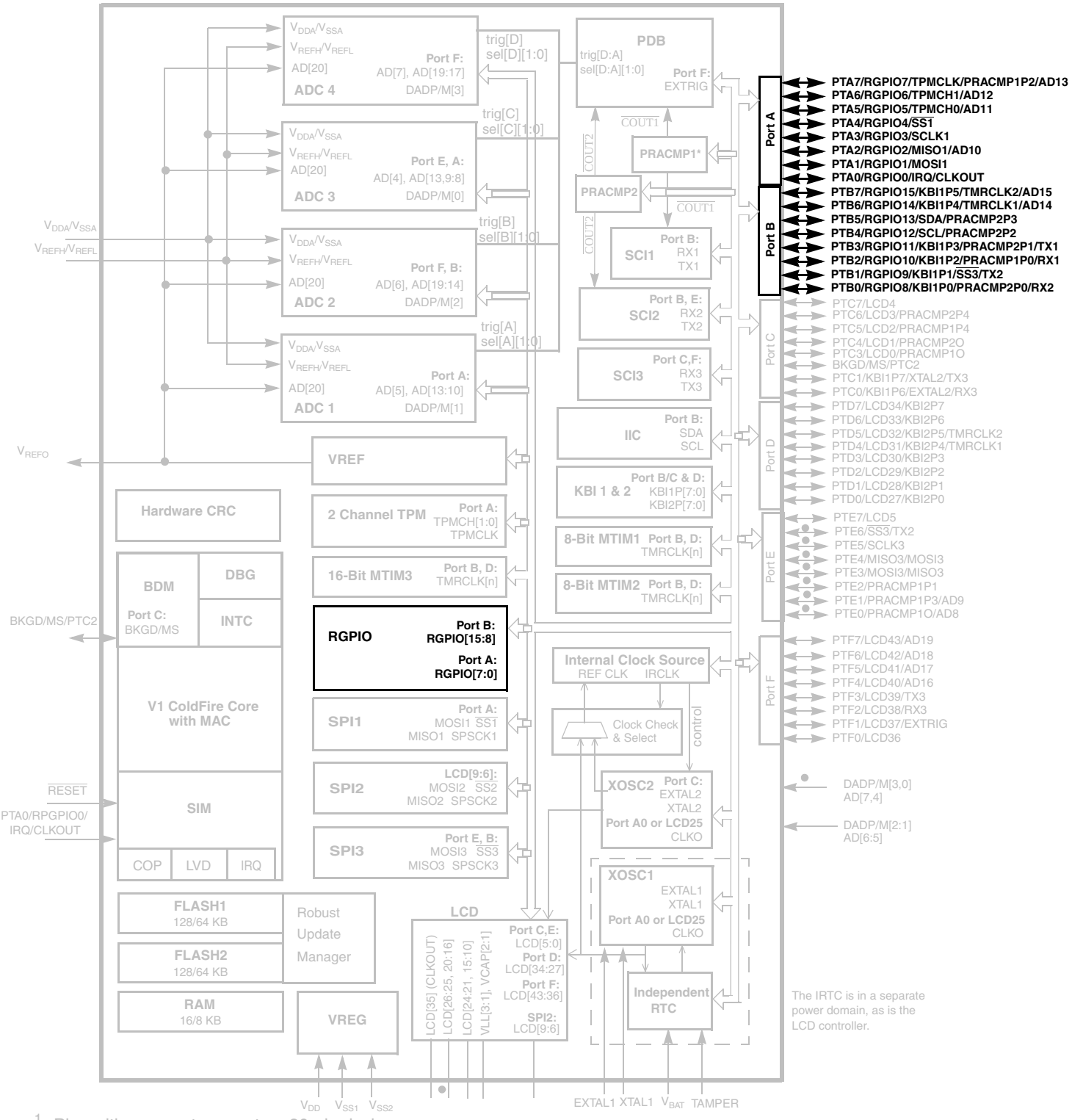
Chapter 5
Rapid GPIO (RGPIO)

5.1 Introduction

The Rapid GPIO (RGPIO) module provides a 16-bit general-purpose I/O module directly connected to the processor's high-speed 32-bit local bus. This connection plus support for single-cycle, zero wait-state data transfers allows the RGPIO module to provide improved pin performance when compared to more traditional GPIO modules located on the internal slave peripheral bus.

Many of the pins associated with a device may be used for several different functions. Their primary functions are to provide external interfaces to access off-chip resources. When not used for their primary function, many of the pins may be used as general-purpose digital I/O (GPIO) pins. The definition of the exact pin functions and the affected signals is specific to each device. Every GPIO port, including the RGPIO module, has registers that configure, monitor, and control the port pins.

Figure 5-1 shows the MCF51EM256/128 block diagram with the RGPIO highlighted.



1 Pins with • are not present on 80-pin devices.

2 PRACMP1 has two less available inputs on the 80-pin devices.

Figure 5-1. MCF51EM256/128 Block Diagram Highlighting RGPIO and Pins

5.1.1 Overview

The RGPIO module provides 16-bits of high-speed GPIO functionality, mapped to the processor's bus. The key features of this module include:

- 16 bits of high-speed GPIO functionality connected to the processor's local 32-bit bus
- Memory-mapped device connected to the ColdFire core's local bus
  - Support for all access sizes: byte, word, and longword
  - All reads and writes complete in a single data phase cycle for zero wait-state response
- Data bits can be accessed directly or via alternate addresses to provide set, clear, and toggle functions
  - Alternate addresses allow set, clear, toggle functions using simple store operations without the need for read-modify-write references
- Unique data direction and pin enable control registers
- Package pin toggle rates typically 1.5–3.5x faster than comparable pin mapped onto peripheral bus

A simplified block diagram of the RGPIO module is shown in Figure 5-2. The details of the pin muxing and pad logic are device-specific.

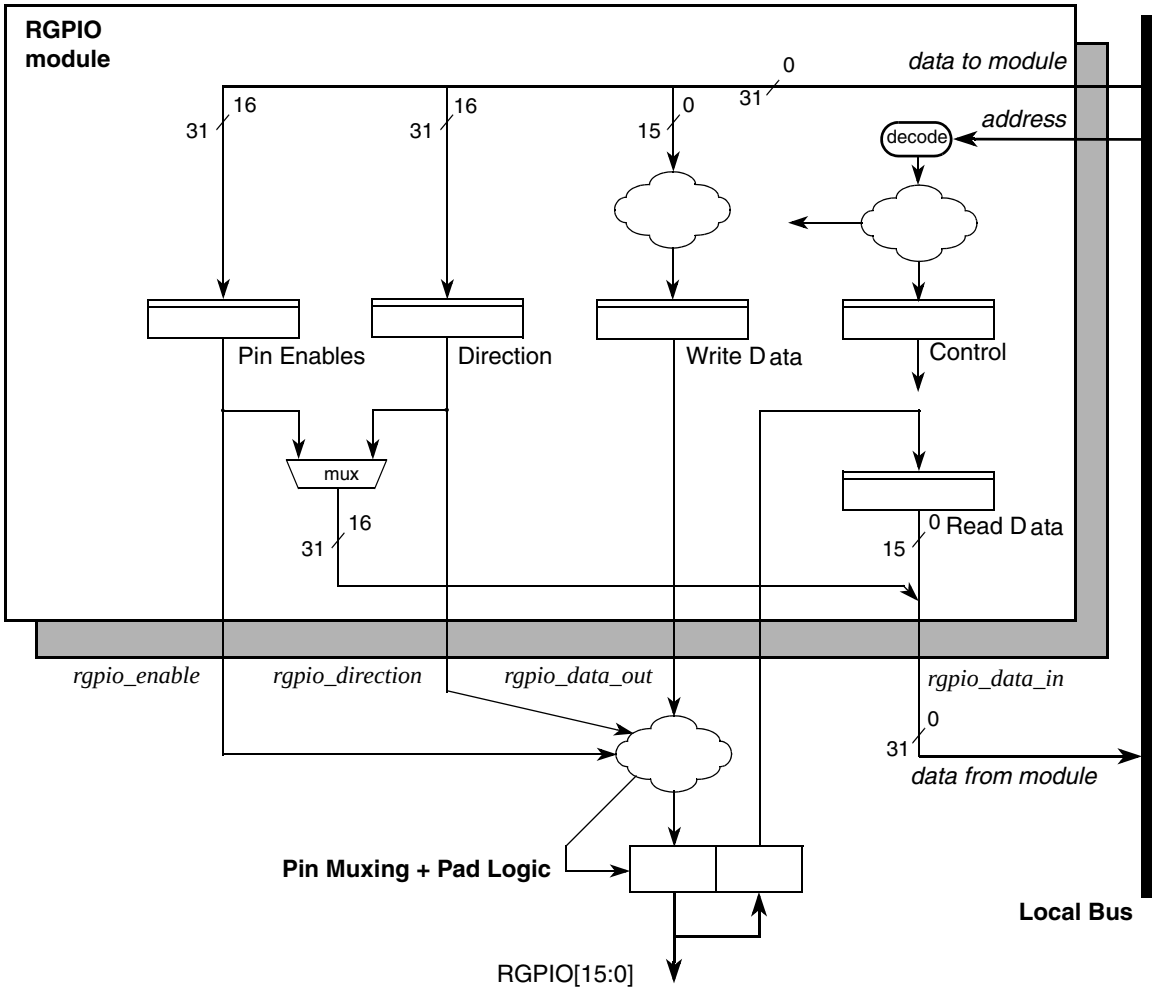


Figure 5-2. RGPIO Block Diagram

5.1.2 Features

The major features of the RGPIO module providing 16 bits of high-speed general-purpose input/output are:

- Small memory-mapped device connected to the processor's local bus
  - — All memory references complete in a single cycle to provide zero wait-state responses
  - — Located in processor's high-speed clock domain
- Simple programming model
  - — Four 16-bit registers, mapped as three program-visible locations
    - – Register for pin enables
    - – Register for controlling the pin data direction
    - – Register for storing output pin data
    - – Register for reading current pin state

  - The two data registers (read, write) are mapped to a single program-visible location
- Alternate addresses to perform data set, clear, and toggle functions using simple writes
- Separate read and write programming model views enable simplified driver software
  - Support for any access size (byte, word, or longword)

5.1.3 Modes of Operation

The RGPIO module does not support any special modes of operation. As a memory-mapped device located on the processor's high-speed local bus, it responds based strictly on memory address and does not consider the operating mode (supervisor, user) of its references.

5.2 External Signal Description

5.2.1 Overview

As shown in Figure 5-2, the RGPIO module's interface to external logic is indirect via the device pin-muxing and pad logic. For a list of the associated RGPIO input/output signals, see Table 5-1.

Table 5-1. RGPIO Module External I/O Signals

Signal Name	Type	Description
RGPIO[15:0]	I/O	RGPIO Data Input/Output

5.2.2 Detailed Signal Descriptions

Table 5-2 provides descriptions of the RGPIO module's input and output signals.

Table 5-2. RGPIO Detailed Signal Descriptions

Signal	I/O	Description	
RGPIO[15:0]	I/O	Data Input/Output. When configured as an input, the state of this signal is reflected in the read data register. When configured as an output, this signal is the output of the write data register.	
		State Meaning	Asserted— Input: Indicates the RGPIO pin was sampled as a logic high at the time of the read. Output: Indicates a properly-enabled RGPIO output pin is to be driven high. Negated— Input: Indicates the RGPIO pin was sampled as a logic low at the time of the read. Output: Indicates a properly-enabled RGPIO output pin is to be driven low.
		Timing	Assertion/Negation— Input: Anytime. The input signal is sampled at the rising-edge of the processor's high-speed clock on the data phase cycle of a read transfer of this register. Output: Occurs at the rising-edge of the processor's high-speed clock on the data phase cycle of a write transfer to this register. This output is asynchronously cleared by system reset.

5.3 Memory Map/Register Definition

The RGPIO module provides a compact 16-byte programming model based at a system memory address of 0x(00)C0_0000 (noted as RGPIO_BASE throughout the chapter). As previously noted, the programming model views are different between reads and writes as this enables simplified software for manipulation of the RGPIO pins. Additionally, the programming model can be referenced using any operand size access (byte, word, longword). Performance is typically maximized using 32-bit accesses.

NOTE

Writes to the two-byte fields at RGPIO_BASE + 0x8 and RGPIO_BASE + 0xC are allowed, but do not affect any program-visible register within the RGPIO module.

Table 5-3. RGPIO *Write* Memory Map

Offset Address	Register	Width (bits)	Access	Reset Value	Section/Page
0x00	RGPIO Data Direction Register (RGPIO_DIR)	16	W	0x0000	5.3.1/5-6
0x02	RGPIO Write Data Register (RGPIO_DATA)	16	W	0x0000	5.3.2/5-7
0x04	RGPIO Pin Enable Register (RGPIO_ENB)	16	W	0x0000	5.3.3/5-8
0x06	RGPIO Write Data Clear Register (RGPIO_CLR)	16	W	N/A	5.3.4/5-8
0x0A	RGPIO Write Data Set Register (RGPIO_SET)	16	W	N/A	5.3.5/5-9
0x0E	RGPIO Write Data Toggle Register (RGPIO_TOG)	16	W	N/A	5.3.6/5-9

Table 5-4. RGPIO *Read* Memory Map

Offset Address	Register	Width (bits)	Access	Reset Value	Section/Page
0x00	RGPIO data direction register (RGPIO_DIR)	16	R	0x0000	5.3.1/5-6
0x02	RGPIO write data register (RGPIO_DATA)	16	R	0x0000	5.3.2/5-7
0x04	RGPIO pin enable register (RGPIO_ENB)	16	R	0x0000	5.3.3/5-8
0x06	RGPIO write data register (RGPIO_DATA)	16	R	0x0000	5.3.2/5-7
0x08	RGPIO data direction register (RGPIO_DIR)	16	R	0x0000	5.3.1/5-6
0x0A	RGPIO write data register (RGPIO_DATA)	16	R	0x0000	5.3.2/5-7
0x0C	RGPIO data direction register (RGPIO_DIR)	16	R	0x0000	5.3.1/5-6
0x0E	RGPIO write data register (RGPIO_DATA)	16	R	0x0000	5.3.2/5-7

5.3.1 RGPIO Data Direction (RGPIO_DIR)

The read/write RGPIO_DIR register defines whether a properly-enabled RGPIO pin is configured as an input or output:

- Setting any bit in RGPIO_DIR configures a properly-enabled RGPIO port pin as an output
- Clearing any bit in RGPIO_DIR configures a properly-enabled RGPIO port pin as an input

At reset, all bits in the RGPIO_DIR are cleared.

Offset: RGPIO_Base + 0x0 (RGPIO_DIR) Access: Read/write
RGPIO_Base + 0x8 Read-only
RGPIO_Base + 0xC Read-only

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

Figure 5-3. RGPIO Data Direction Register (RGPIO_DIR)

Table 5-5. RGPIO_DIR Field Descriptions

Field	Description
15–0 DIR	RGPIO data direction. 0 A properly-enabled RGPIO pin is configured as an input 1 A properly-enabled RGPIO pin is configured as an output

5.3.2 RGPIO Data (RGPIO_DATA)

The RGPIO_DATA register specifies the write data for a properly-enabled RGPIO output pin or the sampled read data value for a properly-enabled input pin. An attempted read of the RGPIO_DATA register returns undefined data for disabled pins, since the data value is dependent on the device-level pin muxing and pad implementation. The RGPIO_DATA register is read/write. At reset, all bits in the RGPIO_DATA registers are cleared.

To set bits in a RGPIO_DATA register, directly set the RGPIO_DATA bits or set the corresponding bits in the RGPIO_SET register. To clear bits in the RGPIO_DATA register, directly clear the RGPIO_DATA bits, or clear the corresponding bits in the RGPIO_CLR register. Setting a bit in the RGPIO_TOG register inverts (toggles) the state of the corresponding bit in the RGPIO_DATA register.

Offset: RGPIO_Base + 0x2 (RGPIO_DATA) Access: Read/write
RGPIO_Base + 0x6 Read/Indirect Write
RGPIO_Base + 0xA Read/Indirect Write
RGPIO_Base + 0xE Read/Indirect Write

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

Figure 5-4. RGPIO Data Register (RGPIO_DATA)

Table 5-6. RGPIO_DATA Field Descriptions

Field	Description
15–0 DATA	RGPIO data. 0 A properly-enabled RGPIO output pin is driven with a logic 0, or a properly-enabled RGPIO input pin was read as a logic 0 1 A properly-enabled RGPIO output pin is driven with a logic 1, or a properly-enabled RGPIO input pin was read as a logic 1

5.3.3 RGPIO Pin Enable (RGPIO_ENB)

The RGPIO_ENB register configures the corresponding package pin as a RGPIO pin instead of the normal GPIO pin mapped onto the peripheral bus.

The RGPIO_ENB register is read/write. At reset, all bits in the RGPIO_ENB are cleared, disabling the RGPIO functionality.

Offset: RGPIO_Base + 0x4 (RGPIO_ENB) Access: Read/write

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

Figure 5-5. RGPIO Enable Register (RGPIO_ENB)

Table 5-7. RGPIO_ENB Field Descriptions

Field	Description
15–0 ENB	RGPIO enable. 0 The corresponding package pin is configured for use as a normal GPIO pin, not a RGPIO 1 The corresponding package pin is configured for use as a RGPIO pin

5.3.4 RGPIO Clear Data (RGPIO_CLR)

The RGPIO_CLR register provides a mechanism to clear specific bits in the RGPIO_DATA by performing a simple write. Clearing a bit in RGPIO_CLR clears the corresponding bit in the RGPIO_DATA register. Setting it has no effect. The RGPIO_CLR register is write-only; reads of this address return the RGPIO_DATA register.

Offset: RGPIO_Base + 0x6 (RGPIO_CLR) Access: Write-only

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

Figure 5-6. RGPIO Clear Data Register (RGPIO_CLR)

Table 5-8. RGPIO_CLR Field Descriptions

Field	Description
15–0 CLR	RGPIO clear data. 0 Clears the corresponding bit in the RGPIO_DATA register 1 No effect

5.3.5 RGPIO Set Data (RGPIO_SET)

The RGPIO_SET register provides a mechanism to set specific bits in the RGPIO_DATA register by performing a simple write. Setting a bit in RGPIO_SET asserts the corresponding bit in the RGPIO_DATA register. Clearing it has no effect. The RGPIO_SET register is write-only; reads of this address return the RGPIO_DATA register.

Offset: RGPIO_Base + 0xA (RGPIO_SET) Access: Write-only

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

Figure 5-7. RGPIO Set Data Register (RGPIO_SET)

Table 5-9. RGPIO_SET Field Descriptions

Field	Description
15–0 SET	RGPIO set data. 0 No effect 1 Sets the corresponding bit in the RGPIO_DATA register

5.3.6 RGPIO Toggle Data (RGPIO_TOG)

The RGPIO_TOG register provides a mechanism to invert (toggle) specific bits in the RGPIO_DATA register by performing a simple write. Setting a bit in RGPIO_TOG inverts the corresponding bit in the RGPIO_DATA register. Clearing it has no effect. The RGPIO_TOG register is write-only; reads of this address return the RGPIO_DATA register.

Offset: RGPIO_Base + 0xE (RGPIO_TOG) Access: Write-only

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

Figure 5-8. RGPIO Toggle Data Register (RGPIO_TOG)

Table 5-10. RGPIO_TOG Field Descriptions

Field	Description
15–0 TOG	RGPIO toggle data. 0 No effect 1 Inverts the corresponding bit in RGPIO_DATA

5.4 Functional Description

The RGPIO module is a relatively-simple design with its behavior controlled by the program-visible registers defined within its programming model.

The RGPIO module is connected to the processor's local two-stage pipelined bus with the stages of the ColdFire core's operand execution pipeline (OEP) mapped directly onto the bus. This structure allows the processor access to the RGPIO module for single-cycle pipelined reads and writes with a zero wait-state response (as viewed in the system bus data phase stage).

5.5 Initialization Information

The reset state of the RGPIO module disables the entire 16-bit data port. Prior to using the RGPIO port, software typically:

- Enables the appropriate pins in RGPIO_ENB
- Configures the pin direction in RGPIO_DIR
- Defines the contents of the data register (RGPIO_DATA)

5.6 Application Information

This section examines the relative performance of the RGPIO output pins for two simple applications

- The processor executes a loop to toggle an output pin for a specific number of cycles, producing a square-wave output
- The processor transmits a 16-bit message using a three-pin SPI-like interface with a serial clock, serial chip select, and serial data bit.

In both applications, the relative speed of the GPIO output is presented as a function of the location of the output bit (RGPIO versus peripheral bus GPIO).

5.6.1 Application 1: Simple Square-Wave Generation

In this example, several different instruction loops are executed, each generating a square-wave output with a 50% duty cycle. For this analysis, the executed code is mapped into the processor's RAM. This configuration is selected to remove any jitter from the output square wave caused by the limitations defined by the two-cycle flash memory accesses and restrictions on the initiation of a flash access. The following instruction loops were studied:

- BCHG_LOOP — In this loop, a bit change instruction was executed using the GPIO data byte as the operand. This instruction performs a read-modify-write operation and inverts the addressed bit.

A pulse counter is decremented until the appropriate number of square-wave pulses have been generated.

- SET+CLR_LOOP — For this construct, two store instructions are executed: one to set the GPIO data pin and another to clear it. Single-cycle NOP instructions (the tpf opcode) are included to maintain the 50% duty cycle of the generated square wave. The pulse counter is decremented until the appropriate number of square-wave pulse have been generated.

The square-wave output frequency was measured and the relative performance results are presented in Table 5-11. The relative performance is stated as a fraction of the processor's operating frequency, defined as *f* MHz. The performance of the BCHG loop operating on a GPIO output is selected as the reference.

Table 5-11. Square-Wave Output Performance

Loop	Peripheral Bus-mapped GPIO			RGPIO		
	Sq-Wave Frequency	Frequency @ CPU *f* = 50 MHz	Relative Speed	Sq-Wave Frequency	Frequency @ CPU *f* = 50 MHz	Relative Speed
bchg	(1/24) × *f* MHz	2.083 MHz	1.00x	(1/14) × *f* MHz	3.571 MHz	1.71x
set+clr (+toggle)	(1/12) × *f* MHz	4.167 MHz	2.00x	(1/8) × *f* MHz	6.250 MHz	3.00x

NOTE

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

5.6.2 Application 2: 16-bit Message Transmission using SPI Protocol

In this second example, a 16-bit message is transmitted using three programmable output pins. The output pins include a serial clock, an active-high chip select, and the serial data bit. The software is configured to sample the serial data bit at the rising-edge of the clock with the data sent in a most-significant to least-significant bit order. The resulting 3-bit output is shown in Figure 5-9.

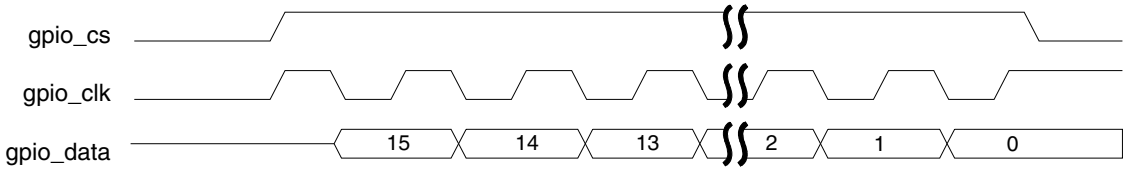


Figure 5-9. GPIO SPI Example Timing Diagram

For this example, the processing of the SPI message is considerably more complex than the generation of a simple square wave of the previous example. The code snippet used to extract the data bit from the message and build the required GPIO data register writes is shown in Figure 5-10.

```
# subtest: send a 16-bit message via a SPI interface using a RGPIO

                    # the SPI protocol uses a 3-bit value: clock, chip-select, data
                    # the data is centered around the rising-edge of the clock

                            align   16
                    send_16b_spi_message_rgpio:
```

```
00510: 4fef fff4              lea     -12(%sp),%sp        # allocate stack space
00514: 48d7 008c              movm.l  &0x8c,(%sp)         # save d2,d3,d7
00518: 3439 0080 0582         mov.w   RAM_BASE+message2,%d2   # get 16-bit message
0051e: 760f                   movq.l  &15,%d3             # static shift count
00520: 7e10                   movq.l  &16,%d7             # message bit length
00522: 207c 00c0 0003         mov.l   &RGPIO_DATA+1,%a0   # pointer to low-order data byte
00528: 203c 0000 ffff         mov.l   &0xffff,%d0         # data value for _ENB and _DIR regs
0052e: 3140 fffd              mov.w   %d0,-3(%a0)         # set RGPIO_DIR register
00532: 3140 0001              mov.w   %d0,1(%a0)          # set RGPIO_ENB register

00536: 223c 0001 0000         mov.l   &0x10000,%d1        # d1[17:16] = {clk, cs}
0053c: 2001                   mov.l   %d1,%d0             # copy into temp reg
0053e: e6a8                   lsr.l   %d3,%d0             # align in d0[2:0]
00540: 5880                   addq.l  &4,%d0              # set clk = 1
00542: 1080                   mov.b   %d0,(%a0)           # initialize data
00544: 6002                   bra.b   L%1
                              align   4

                      L%1:
00548: 3202                   mov.w   %d2,%d1             # d1[17:15] = {clk, cs, data}
0054a: 2001                   mov.l   %d1,%d0             # copy into temp reg
0054c: e6a8                   lsr.l   %d3,%d0             # align in d0[2:0]
0054e: 1080                   mov.b   %d0,(%a0)           # transmit data with clk = 0
00550: 5880                   addq.l  &4,%d0              # force clk = 1
00552: e38a                   lsl.l   &1,%d2              # d2[15] = new message data bit
00554: 51fc                   tpf                         # preserve 50% duty cycle
00556: 51fc                   tpf
00558: 51fc                   tpf
0055a: 51fc                   tpf
0055c: 1080                   mov.b   %d0,(%a0)           # transmit data with clk = 1
0055e: 5387                   subq.l  &1,%d7              # decrement loop counter
00560: 66e6                   bne.b   L%1

00562: c0bc 0000 fff5         and.l   &0xfff5,%d0         # negate chip-select
00568: 1080                   mov.b   %d0,(%a0)           # update gpio

0056a: 4cd7 008c              movm.l  (%sp),&0x8c         # restore d2,d3,d7
0056e: 4fef 000c              lea     12(%sp),%sp         # deallocate stack space
00572: 4e75                   rts
```

Figure 5-10. GPIO SPI Code Example

The resulting SPI performance, as measured in the effective Mbps transmission rate for the 16-bit message, is shown in Table 5-12.

Table 5-12. Emulated SPI Performance using GPIO Outputs

Peripheral Bus-mapped GPIO		RGPIO	
SPI Speed @ CPU *f* = 50 MHz	Relative Speed	SPI Speed @ CPU *f* = 50 MHz	Relative Speed
2.063 Mbps	1.00x	3.809 Mbps	1.29x

Chapter 6
Modes of Operation

6.1 Introduction

The operating modes of the MCF51EM256 series are described in this chapter. Entry into each mode, exit from each mode, and functionality while in each of the modes are described.

The overall system mode is generally a function of a number of separate, but inter-related variables: debug mode, security mode, power mode, and clock mode. Clock modes were discussed in Section 1.3.5, "ICS Modes of Operation". This chapter explores the other dimensions of the system operating mode.

6.2 Features

- Debug mode for code development. For V1 ColdFire devices, such as MCF51EM256 series, debug mode is mutually exclusive with use of secure mode (next item).
- Secure mode — BDC access to CPU resources is extremely restricted. It is possible to tell that the device has been secured, and to clear security, which involves mass erasing the on-chip flash memory. No other CPU access is allowed. Secure mode can be used in conjunction with each of the power modes below.
- Run mode — CPU clocks can be run at full speed and the internal supply is fully regulated.
- LPrun mode — CPU and peripheral clocks are restricted to 250 kHz CPU clock and 125 kHz bus clock maximum and the internal supply is in loose regulation.
- Wait mode — CPU shuts down to conserve power; peripheral clocks are running and full regulation is maintained.
- LPwait mode — CPU shuts down to conserve power; peripheral clocks are running at reduced speed (125 kHz maximum) and the internal voltage regulator is running in loose regulation mode.
- Stop modes — System (CPU and peripheral) clocks are stopped.
  - Stop4 — All internal circuits are powered (full regulation mode) and internal clock sources still at max frequency for fastest recovery.
  - Stop3 — All internal circuits are loosely regulated and clocks sources are at minimal values (125 kHz maximum), providing a good compromise between power utilization and speed of recovery.
  - Stop2 — Partial power-down of internal circuits; RAM content is retained. The lowest power mode for this device. Upon wakeup from stop2, the MCU will go through a reset sequence, even if the source of the wakeup was an interrupt.

On the MCF51EM256 series, wait, stop2, stop3, and stop4 are all entered via the CPU STOP instruction. See Table 6-1, Figure 6-2, and subsequent sections of this chapter for details.

6.3 Overview

The ColdFire CPU has two primary user modes of operation, run and stop. (The CPU also supports a halt mode that is used strictly for debug operations.) The STOP instruction is used to invoke stop and wait modes for this family of devices.

The Systems Option Register 1 (SOPT1) contains two bits which control operation of the STOP instruction. If SOPT1[WAITE] is set when STOP is executed, the wait mode is entered. Otherwise, if SOPT1[STOPE] is set, the CPU enters one of the stop modes. It is illegal to execute a STOP instruction if neither STOPE or WAITE are set. This results in reset assertion if the Instruction-related Reset Disable bit in the CPU Control Register (CPUCR[IRD]) is cleared or an illegal instruction exception if CPUCR[IRD] is set.

The MCF51EM256 series devices augment stop, wait, and run in a number of ways. The power management controller (PMC) can run the device in fully-regulated mode, standby mode, and partial power-down mode. Standby (loose regulation) or partial power-down can be programmed to occur naturally as a result of a STOP instruction. Additionally, standby mode can be explicitly invoked via the LPR (low-power) bit in the PMC System Power Management Status & Control Register 2 (SPMSC2[LPR]). Use of standby is limited to bus frequencies less than 125 kHz; and neither standby nor partial power-down are allowed when XCSR[ENBDM] bit is set to enable debugging in stop and wait modes.

During partial power-down mode, the regulator is in standby mode and much of the digital logic on the chip is switched off. These interactions can be seen schematically in Figure 6-1. This figure is for conceptual purposes only. It does not reflect any sequence or time dependencies between the PMC and other parts of the device, nor does it represent any actual design partitioning.

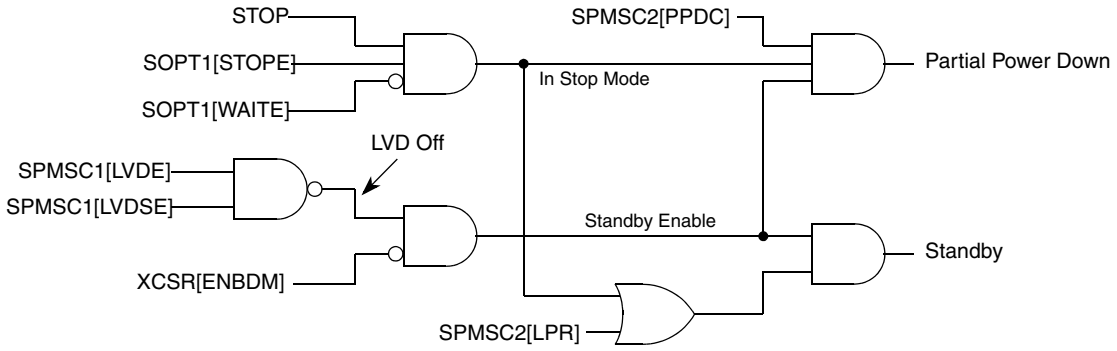


Figure 6-1. MCF51EM256 Series Power Modes — Conceptual Drawing

It is illegal for the software to have SPMSC2[PPDC] and SPMSC2[LPR] asserted concurrently. This restriction arises because the sequence of events from normal to low-power modes involves use of both bits. After entering a low-power mode, it is not possible to switch to another low-power mode.

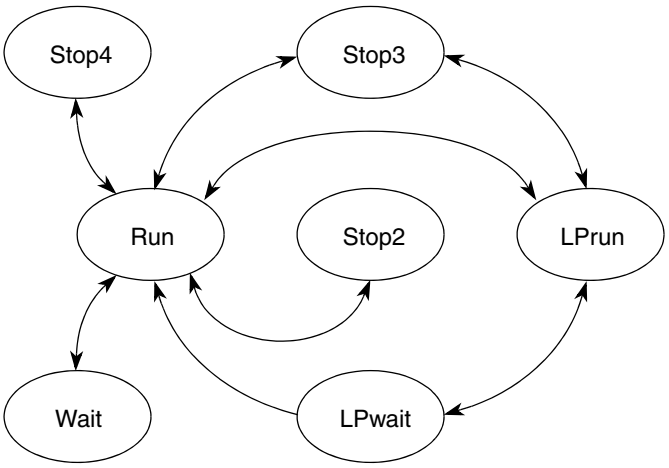
Table 6-1. CPU / Power Mode Selections

Mode of Operation	SOPT1 SIM		XCSR BDC	SPMSC1 PMC		SPMSC2 PMC		CPU and Peripheral Clocks	Effects on Sub-System	
	STOPE	WAITE	ENBDM[1]	LVDE	LVDSE	LPR	PPDC		BDC Clock	Switched Power
Run mode — processor and peripherals clocked normally.	x	x	x	x	x	0	x	On. ICS in any mode	On **Note:** When not needed, the BDC clock can be gated off at the discretion of the processor. The clock is available within a few cycles of demand by the processor, normally when a negative edge is detected on BKGD. The BDC command associated with that negative edge may not take affect.	On
			x	1	1	x	x			
			1	x	x	x	x			
LPrun mode with low voltage detect disabled — processor and peripherals clocked at low frequency[2]. Low voltage detects are not active.	x	x	0	0	x	1	0	Low frequency required. ICS in FBELP mode.		Loose Reg
			0	1	0					
Wait mode — processor clock nominally inactive, but peripherals are clocked.	x	1	x	x	x	0	x	Peripheral clocks on CPU clock on if XCSR[ENBDM]=1		On
			x	1	1	x	x			
			1	x	x	x	x	On		
LPwait mode — processor clock is inactive, peripherals are clocked at low frequency and the PMC is loosely regulating. Low voltage detects are not active.	x	1	0	0	x	1	0	CPU clock is off. Peripheral clocks at low speed. ICS in BLPE.		Loose Reg
				1	0					
Stop modes disabled — Illegal opcode reset if STOP instruction executed and CPUCR[IRD] is cleared, else illegal instruction exception is generated.	0	0	Function of BKGD/ MS at reset	⇒1	⇒1	⇒0	⇒0	⇒On	Function of BKGD/MS at reset	⇒On
Stop4 — Either low-power modes have not been requested, or low voltage detects are enabled or XCSR[ENBDM] = 1.	1	0	x	1	1	0	0	Peripheral clocks off. CPU clock on if XCSR[ENBDM]=1	BDC clock enabled only if XCSR[ENBDM]=1 prior to entering stop.	On
			x	1	1	1	0			
			x	1	1	0	1			
			1	x	x	x	x	CPU clock on. Periph clocks off.		
Stop3 — Low voltage detect in stop is not enabled. Clocks must be at low frequency and are gated. The regulator is in loose regulation.	1	0	0	x	0	x[3]	0	Low freq required. ICS in BLPE mode. CPU and peripheral clocks are gated off.	Off	Loose Reg
				0	x					
Stop2 — Low voltage detects are not active. If BDC is enabled, stop4 is invoked rather than stop2.	1	0	0	x	0	0	1	N/A	N/A	Off
				0	x					

1 ENBDM is located in the upper byte of the XCSR register which is write-accessible only through BDC commands, see Section 26.3.2, “Extended Configuration/Status Register (XCSR)”.

2 250 kHz maximum CPU frequency in LPrun; 125 kHz maximum peripheral clock frequency.

3 Assuming that neither low voltage interrupts nor BDM are enabled, stop3 can be entered from run simply by executing a STOP command. It can also be entered from LPrun via stop.



Mode	Regulator State
Run	Full On
Wait	Full On
Stop4	Full On
LPrun	Standby
LPwait	Standby
Stop3	Standby
Stop2	Partial Power Off

Figure 6-2. Allowable Power Mode Transitions for Mission Mode MCF51EM256 Series

Figure 6-2 illustrates mission mode state transitions allowed between the legal states shown in Table 6-1. $\overline{\text{RESET}}$ must be asserted low, or the IRTC must issue a wakeup signal, in order to exit stop2. Only interrupt assertion is necessary to exit the other stop and wait modes.

Figure 6-3 takes the same set of states and transitions shown in Figure 6-2 and adds the BDM halt mode for development purposes. If BDM is enabled, the chip automatically shifts LP modes into their fully regulated equivalents. If software or debugger sets SPMSC2[LPR] while BDM is enabled, SPMSC2[LPRS] reflects the fact that the regulator is not in standby. Similarly, SPMSC2[PPDF] does not indicate a recovery from stop2 if XCSR[ENBDM] forced stop4 to occur in its place.[1]

Stated another way, if XCSR[ENBDM] has been set via the BDM interface, then the power management controller keeps (or puts) the regulator in full regulation despite other settings on the contrary. The states shown in Figure 6-3 then map as follows:

- LPrun ⇒ Run
- LPwait ⇒ Wait
- Stop3 ⇒ Stop4
- Stop2 ⇒ Stop4

From a software perspective (and disregarding PMC status bits), the system remains in the appropriate low-power state, and can be debugged as such.

See Section 6.7, “Wait Modes,” for a description of the various ways to enter halt mode.

1. This can have subtle impacts on recovery from stop. The IRQ input can wake the device from stop4 if it has been enabled for that purpose. A low on the RESETB pin wakes the device from stop2 (there is an asynchronous path to the power management controller in that state).

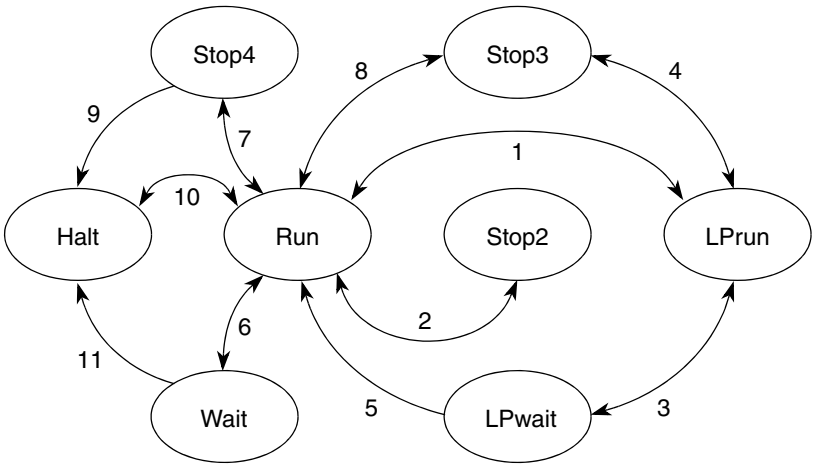


Figure 6-3. All Allowable Power Mode Transitions for MCF51EM256 Series

Table 6-2 defines triggers for various state transitions shown in Figure 6-2.

Table 6-2. Triggers for State Transition

Transition #	From	To	Trigger
1	Run	LPrun	Configure settings shown in Table 6-1, switch LPR = 1 last
	LPrun	Run	Clear SPMSC2[LPR]
			Interrupt when SPMSC2[LPWUI] = 1
			Negative transition on enabled BKGD/MS pin.
2	Run	Stop2	Pre-configure settings shown in Table 6-1, execute STOP instruction
	Stop2	Run	Assert zero on $\overline{\text{RESET}}$[1] or IRTC timeout. Reload environment from RAM. IRQ assertion
3	LPrun	LPwait	Pre-configure settings shown in Table 6-1, execute STOP instruction
	LPwait	LPrun	Interrupt when SPMSC2[LPWUI]=0
4	LPrun	Stop3	Execute STOP instruction
	Stop3	LPrun	Interrupt when SPMSC2[LPWUI]=0
5	LPwait	Run	Interrupt when SPMSC2[LPWUI]=1
	Run	LPwait	Not supported.
6	Run	Wait	Pre-configure settings shown in Table 6-1, execute STOP instruction
	Wait	Run	Interrupt
7	Run	Stop4	Pre-configure settings shown in Table 6-1, execute STOP instruction
	Stop4	Run	Interrupt

Table 6-2. Triggers for State Transition (continued)

Transition #	From	To	Trigger
8	Stop3	Run	Interrupt when SPMSC2[LPWUI]=1
	Run	Stop3	Pre-configure settings shown in Table 6-1, execute STOP instruction. The LP bit does not have to be set prior to entering stop3 in this fashion. But low voltage interrupts and BDM must be disabled.
9	Stop4	Halt	When a BACKGROUND command is received through the BKGD/MS pin (XCSR[ENBDM] must equal one).
	Halt	Stop4	Not supported.
10	Halt	Run	GO instruction issued via BDM
	Run	Halt	When a BACKGROUND command is received through the BKGD/MS pin OR When a HALT instruction is executed OR When encountering a BDM breakpoint
11	Wait	Halt	When a BACKGROUND command is received through the BKGD/MS pin (XCSR[ENBDM] must equal one).
	Halt	Wait	Not supported.

[1] An analog connection from this pin to the on-chip regulator wakes up the regulator, which then initiates a power-on-reset sequence.

Individual power states are discussed in more detail in the following sections.

6.4 Debug Mode

Debug mode functions are managed through the background debug controller (BDC) in the Version 1 ColdFire core. The BDC provides the means for analyzing MCU operation during software development.

The debug interface is used to program a bootloader or user application program into the flash program memory before the MCU is operated in run mode for the first time. When the MCF51EM256 series are shipped from the Freescale Semiconductor factory, the flash program memory is erased by default unless specifically noted, so there is no program that could be executed in run mode until the flash memory is initially programmed. The debug interface can also be used to erase and reprogram the flash memory after it has been previously programmed.

See Chapter 26, “Version 1 ColdFire Debug (CF1_DEBUG),” for more details regarding the debug interface.

6.5 Secure Mode

While the MCU is in secure mode, there are severe restrictions on which debug commands can be used. In this mode, only the upper byte of the core’s XCSR, CSR2, and CSR3 registers can be accessed. See Chapter 26, “Version 1 ColdFire Debug (CF1_DEBUG),” for details.

6.6 Run Modes

6.6.1 Run Mode

Run mode is the normal operating mode for the MCF51EM256 series. This mode is selected when the BKGD/MS pin is high at the rising edge of the internal reset signal. Upon exiting reset, the CPU fetches the supervisor SR and initial PC from locations 0x(00)00_0000 and 0x(00)00_0004 in the memory map and executes code starting at the newly set value of the PC.

6.6.2 Low-Power Run Mode (LPrun)

In the low-power run mode, the on-chip voltage regulator is put into its standby (or loose regulation) state. In this state, the power consumption is reduced to a minimum that allows CPU functionality. Power consumption is reduced the most by disabling the clocks to all unused peripherals by clearing the corresponding bits in the SCGC1-5 registers[1].

Before entering this mode, the following conditions must be met:

- FBELP[2] is the selected clock mode for the ICS. See Section 11.4.1, “Operational Modes,” for more details.
- ICSC2[HGO] is cleared.
- The bus frequency is less than 125 kHz.
- The ADC must be in low-power mode (ADCCFG1[ADLPC] = 1) or disabled.
- Low-voltage detect must be disabled. The LVDE and/or LVDSE bit in SPMSC1 register must be cleared.
- Flash programming/erasing is not allowed

After these conditions are met, low-power run mode can be entered by setting SPMSC2[LPR].

To re-enter standard run mode, clear the LPR bit. SPMSC2[LPRS] is a read-only status bit that can be used to determine if the regulator is in full-regulation mode or not. When LPRS is cleared, the regulator is in full-regulation mode and the MCU can run at full speed in any clock mode.

Assuming that SOPT1[BKGDPE] is set to enable BKGD/MS, the device also switches from LPrun to run mode when it detects a negative transition on the BKGD/MS pin.

Low-power run mode also provides the option to return to full regulation if any interrupt occurs. This is done by setting SPMSC2[LPWUI]. The ICS FLLs can then be set for full speed immediately in the interrupt service routine.

6.6.2.1 BDM in Low-Power Run Mode

Low-power run mode cannot be entered when the MCU is in active background debug mode.

If a device is in low-power run mode, a falling edge on the BKGD/MS pin exits low-power run/wait mode, clears the LPRS bit in SPMSC2, and returns the device to normal run mode. The LPR bit remains set and

1. System clock gating control registers 1, 2, 3, 4 & 5
2. FLL bypassed external low-power

low power run mode cannot be entered again until after a system reset. Note that BKGD/MS is multiplexed with PTC2. The pin must be configured as BKGD/MS for this operation to occur.

6.7 Wait Modes

6.7.1 Wait Mode

Wait mode is entered by executing a STOP instruction after configuring the device as per Table 6-1. Upon execution of the STOP instruction, the CPU enters a low-power state in which it is not clocked.

The V1 ColdFire core does not differentiate between stop and wait modes. Both are stop from the core's perspective. The difference between the two is at the device level. In stop mode, most peripheral clocks are shut down. In wait mode, they continue to run.

XCSR[ENBDM] must be set prior to entering wait mode if the device is required to respond to BDM commands once in wait.

When an interrupt request occurs, the CPU exits wait mode and resumes with exception processing, beginning with the stacking operations leading to the interrupt service routine.

6.7.2 Low-Power Wait Mode (LPwait)

Low-power wait mode is entered by executing a STOP instruction while the MCU is in low-power run mode and configured per Table 6-1. In the low-power wait mode, the on-chip voltage regulator remains in its standby state as in the low-power run mode. In this state, the power consumption is reduced to a minimum that allows most modules to maintain funtionality. Power consumption is reduced the most by disabling the clocks to all unused peripherals by clearing the corresponding bits in the SCGCx registers.

Low-power run mode restrictions also apply to low-power wait mode.

If SPMSC2[LPWUI] is set when the STOP instruction is executed, the voltage regulator returns to full regulation when wait mode is exited. The ICS FLLs can be set for full speed immediately in the interrupt service routine.

If SPMSC2[LPWUI] is cleared when the STOP instruction is executed, the device returns to low-power run mode.

Any reset exits low-power wait mode, clears SPMSC2[LPR], and returns the device to normal run mode.

6.7.2.1 BDM in Low-Power Wait Mode

If a device is in low-power wait mode, a falling edge on the BKGD/MS pin exits low-power run/wait mode, clears the LPRS bit in SPMSC2, and returns the device to normal run mode. The LPR bit remains set and low power run mode cannot be entered again until after a system reset. Again, note that BKGD/MS is multiplexed with PTC2. The pin must be configured as BKGD/MS for this operation to occur.

6.8 Stop Modes

One of three stop modes is entered upon execution of a STOP instruction when SOPT1[STOPE] is set. The SOPT1[WAITE] bit must be clear, else wait mode is entered. In stop3 mode, the bus and CPU clocks are halted. If XCSR[ENBDM] is set prior to entering stop4, only the peripheral clocks are halted. The ICS module can be configured to leave the reference clocks running. See Chapter 11, "Internal Clock Source (ICS)," for more information.

NOTE

If neither the WAITE nor STOPE bit is set when the CPU executes a STOP instruction, the MCU does not enter either of the stop modes. Instead, the MCU initiates an illegal opcode reset if CPUCR[IRD] is cleared or an illegal instruction exception if CPUCR[IRD] is set.

The stop modes are selected by setting the appropriate bits in the system power management status and control 2 (SPMSC2) register. Table 6-1 shows all of the control bits that affect mode selection under various conditions. The selected mode is entered following the execution of a STOP instruction.

Most background commands are not available in stop mode. The memory-access-with-status commands do not allow memory access, but they report an error indicating that the MCU is in either stop or wait mode. The BACKGROUND command can be used to wake the MCU from stop4 and enter halt mode if XCSR[ENBDM] was set prior to entering stop. After entering halt mode, all background commands are available.

6.8.1 Stop2 Mode

Stop2 mode is entered by executing a STOP instruction under the conditions as shown in Table 6-1.

Most of the internal circuitry of the MCU is powered off in stop2 with the exception of the RAM and the IRTC. Upon entering stop2, all I/O pin control signals are latched so that the pins retain their states during stop2.

Exit from stop2 is performed by driving the wake-up pin ($\overline{\text{RESET}}$ and PTA0/RGPIO0/IRQ/CLKOUT) on the MCU to zero.

NOTE

PTA0/RGPIO0/IRQ/CLKOUT functions as an active-low wakeup input when the MCU is in stop2, as long as the pin is configured as an input before entering stop2. The pullup on this pin is not automatically enabled in stop2. To enable the internal pullup, set PTAPE[PTAPE0].

In addition, the independent real-time counter (IRTC) can wake the MCU from stop2, if enabled.

Upon wakeup from stop2 mode, the MCU starts up as from a power-on reset (POR):

- All module control and status registers are reset, with the exception of the power management controller (SPMSC1/2/3), IRTC, LCD, and debug trace buffer. Refer to the individual module chapters for more information on which other registers are unaffected by wake-up from stop2 mode.

- The LVD reset function is enabled and the MCU remains in the reset state if V_{DD} is below the LVD trip point (low trip point selected due to POR).
- The CPU initiates reset exception processing by fetching the vectors at 0x(00)00_0000 and 0x(00)00_0004.

In addition to the above, upon waking up from stop2, SPMSC2[PPDF] is set. This flag is used to direct user code to go to a stop2 recovery routine. PPDF remains set and the I/O pin states remain latched until a 1 is written to SPMSC2[PPDACK].

Wakeup from stop2 can be initiated with an IRTC interrupt. Unlike most other modules on the chip, the IRTC is not reset as a result of exiting stop2. This implies that the IRTC interrupt is asserted (although masked) upon exit from stop2.

To maintain I/O states for pins configured as general-purpose I/O before entering stop2, restore the contents of the I/O port registers, which have been saved in RAM, to the port registers before writing to the PPDACK bit. If the port registers are not restored from RAM before writing to PPDACK, the pins switch to their reset states when PPDACK is written.

For pins that were configured as peripheral I/O, reconfigure the peripheral module that interfaces to the pin before writing to PPDACK. If the peripheral module is not enabled before writing to PPDACK, the pins are controlled by their associated port control registers when the I/O latches are opened.

6.8.1.1 XOSC2 Considerations for Stop2

If using a low-range oscillator during stop2, reconfigure the ICSC2 register before PPDACK is written. The low-range oscillator (ICSC2[RANGE] = 0) can operate in stop2 as the clock source for the LCD module. If the low-range oscillator is active when entering stop2, it remains active in stop2 regardless of the value of ICSC2[EREFSTEN]. To disable the oscillator in stop2, switch the ICS into FBI or FEI mode before executing the STOP instruction.

The crystal oscillator cannot be used in high range in stop2. Systems which utilize crystals in the high frequency range (up to 25MHz) must disable the crystal oscillator and switch to FBI or FEI mode prior to executing the STOP instruction.

6.8.2 Stop3 Mode

Stop3 mode is entered by executing a STOP instruction under the conditions as shown in Table 6-1. The states of all of the internal registers and logic, RAM contents, and I/O pin states are maintained. The on-chip regulator is placed in standby state.

Stop3 can be exited by asserting $\overline{\text{RESET}}$ or by an interrupt from one of the following sources: the PRACMP, IRTC, ADC, IRQ, SCI, LCD or KBI. The following modules are inactive in stop3: SPI, IIC, MTIM, MTIM16, PDB and TPM

If stop3 is exited by the $\overline{\text{RESET}}$ pin, the MCU is reset and operation resumes after taking the reset vector. Exit by one of the internal interrupt sources results in the MCU taking the appropriate interrupt vector.

6.8.3 Stop4: Low Voltage Detect or BDM Enabled in Stop Mode

Stop4 is differentiated from stop2 and stop3 in that the on-chip regulator is fully engaged.

Entry into halt mode from run mode is enabled if the XCSR[ENBDM] bit is set. This register is described in Chapter 26, "Version 1 ColdFire Debug (CF1_DEBUG)". If XCSR[ENBDM] is set when the CPU executes a STOP instruction, the system clocks to the background debug logic remain active when the MCU enters stop mode. Because of this, background debug communication remains possible. If you attempt to enter stop2 or stop3 with XCSR[ENBDM] set, the MCU enters stop4 instead (see Table 6-1 for details).

Stop4 is also entered if SPMSC1[LVDE, LVDSE] are set, enabling low voltage detect when the STOP instruction is executed. The LVD may only be used when the on-chip regulator is in full regulation mode. Thus, stop3 and stop2 modes are not compatible with use of the LVD.

The LVD system is capable of generating an interrupt or a reset when the supply voltage drops below the LVD voltage.

Stop4 can be exited by asserting $\overline{\text{RESET}}$ or by an interrupt from one of the following sources: the PRACMP, IRTC, LVD, LVW, ADC, IRQ, SCI, LCD or the KBI. The following modules are inactive in stop4: SPI, IIC, MTIM, MTIM16, PDB, and TPM.

6.9 On-Chip Peripheral Modules in Stop and Low-Power Modes

When the MCU enters any stop mode (wait not included), peripheral clocks to the internal peripheral modules are stopped. Even in the exception case (XCSR[ENBDM] = 1), where clocks to the background debug logic continue to operate, clocks to the peripheral systems are halted to reduce power consumption. One exception to the above is that the crystal oscillator can continue to run, and can be used by the LCD module to maintain an LCD display, even during STOP2.

Refer to Section 6.8.1, "Stop2 Mode," for specific information on system behavior in stop modes.

When the MCU enters LPwait or LPrun modes, system clocks to the internal peripheral modules continue based on the settings of the clock gating control registers (SCGC1-5).

Table 6-3 defines terms used in Table 6-4 to describe operation of components on the chip in the various low-power modes.

Table 6-3. Term Definition for Table 6-4

Voltage Regulator	Clocked[1]	Not Clocked
Full Regulation	FullOn	FullNoClk FullADACK[2]
Loose Regulation	SoftOn[3]	SoftNoClk Disabled SoftADACK[4]
Off	N/A	Off

[1] Subject to module enables and settings of System Clock Gating Control Registers (SCGC1– SCGC5).

[2] This ADC-specific mode defines the case where the device is fully regulated and the normal peripheral clock is stopped. In this case, the ADC can run using its internally generated asynchronous ADACK clock.

[3] Analog modules must be in their low-power mode when the device is operated in this state.

[4] This ADC-specific mode defines the case where the device is in loose regulation and the normal peripheral clock is stopped. In this case, the ADC can only be run using its low-power mode and internally generated asynchronous ADACK clock.

Table 6-4. Peripheral Operation as a Function of Power Mode

Peripheral	Mode					
	Stop2	**Stop3**	**Stop4**	**LPwait**	**Wait**	**LPrun**
ADC[1,2]	Off	SoftADACK (Wake Up)	FULLADACK (Wake Up)	SoftOn	FullOn	SoftOn
BDC	Off	SoftOn	On	SoftOn	FullOn	SoftOn
CCS[3]	Off	SoftNoClk	FullNoClk	SoftNoClk	FullNoClk	SoftOn
CF1_CORE	Off	SoftNoClk	FullNoClk	SoftNoClk	FullNoClk	SoftOn
COP	Off	SoftNoClk	FullNoClk	SoftOn	FullOn	SoftOn
CRC	Off	SoftNoClk	FullNoClk	SoftOn	FullOn	SoftOn
Crystal Oscillator 1 (IRTC)	On — 32 kHz	On — 32 kHz	On — 32 kHz	On — 32 kHz	On — 32 kHz	On — 32 kHz
Crystal Oscillator 2 (ICS)	RANGE = 0 HGO = 0	RANGE = 0 HGO = 0	All Modes	RANGE = 0 HGO = 0	All Modes	RANGE = 0 HGO = 0
Flash	Off	SoftNoClk	FullNoClk	SoftNoClk	FullNoClk	SoftOn
I/O Pins	States Held	SoftNoClk	FullNoClk	SoftOn	FullOn	SoftOn
ICS	Off	Stop or BLPE[4]	Stop or any mode	BLPE	Any mode	BLPE
IIC	Off	SoftNoClk	FullNoClk	SoftOn	FullOn	SoftOn
IRQ	Off (Wake Up via POR)[5]	SoftNoClk (Wake Up)	FullNoClk (Wake Up)	SoftOn	FullOn	SoftOn
IRTC	FullOn	FullOn	FullOn	FullOn	FullOn	FullOn
KBIx	Off	SoftNoClk (Wake Up)	FullNoClk (Wake Up)	SoftOn	FullOn	SoftOn
LCD	On[6] (Interrupts Disabled)	On	On	On	On	On
LVD/LVW	Off	Disabled	On (Wake Up)	Disabled	FullOn	Disabled
MTIMx	Off	SoftNoClk	FullNoClk	SoftOn	FullOn	SoftOn
PDB	Off	SoftNoClk	FullNoClk	SoftOn	FullOn	SoftOn
Port I/O Registers	Off	SoftNoClk	FullNoClk	SoftOn	FullOn	SoftOn
PRACMP	Off	NoClk (Wake Up)	NoClk (Wake Up)	On	On	On

Table 6-4. Peripheral Operation as a Function of Power Mode (continued)

Peripheral	Mode					
	Stop2	**Stop3**	**Stop4**	**LPwait**	**Wait**	**LPrun**
RAM	SoftNoClk	SoftNoClk	FullNoClk	SoftNoClk	FullNoClk	SoftOn
SCI*x*	Off	SoftNoClk	FullNoClk	SoftOn	FullOn	SoftOn
SPI*x*	Off	SoftNoClk	FullNoClk	SoftOn	FullOn	SoftOn
TPM	Off	SoftNoClk	FullNoClk	SoftOn	FullOn	SoftOn
Voltage Regulator / PMC	Partial Shutdown. 1 kHz osc if enabled	Loose Regulation. 1 kHz osc if enabled	Full Regulation 1 kHz osc on	SoftOn 1 kHz osc on	FullOn 1 kHz osc on	SoftOn 1 kHz osc on
VREF	Off	NoClk	NoClk	On	On	On

1 LP mode for the ADC is invoked by setting ADLPC = 1. ADACK is selected via the ADCCFG[ADICLK] field in the ADC. See Chapter 21, “Analog-to-Digital Converter (ADC16),” for details.

2 LVD must be enabled to run in stop if converting the bandgap channel.

3 CCS clock multiplexor path remains active in stop3/4, although IP functions are not available then.

4 BLPE refers to the ICS bypassed external low-power state. See Chapter 11, “Internal Clock Source (ICS),” for more details.

5 The $\overline{\text{RESET}}$ pin also has a direct connection to the on-chip regulator wakeup input. Asserting this pin low while in stop2 triggers the PMC to wakeup. As a result, the device undergoes a power-on-reset sequence.

6 The LCD can continue to drive a DISPLAY so long as one of the OSCOUT1 or OSCOUT2 signals into the LCD module is enabled for operation in Stop modes.

Chapter 7
Resets, Interrupts, and General System Control

7.1 Introduction

This section discusses basic reset and interrupt mechanisms and the various sources of reset and interrupt on an MCF51EM256 series microcontroller. Some interrupt sources from peripheral modules are discussed in greater detail within other sections of this document. This section gathers basic information about all reset and interrupt sources in one place for easy reference. A few reset and interrupt sources, including the computer operating properly (COP) watchdog are not part of on-chip peripheral systems with their own chapters.

7.2 Features

Reset and interrupt features include:

- Multiple sources of reset for flexible system configuration and reliable operation
- System reset status (SRS) register to indicate source of most recent reset
- Separate interrupt vector for most modules (reduces polling overhead) (see Table 7-1)

7.3 Microcontroller Reset

Resetting the microcontroller provides a way to start processing from an known set of initial conditions. When the ColdFire processor exits reset, it fetches initial 32-bit values for the supervisor stack pointer and program counter from locations 0x(00)00_0000 and 0x(00)00_0004 respectively. On-chip peripheral modules are disabled and I/O pins are initially configured as general-purpose high-impedance inputs with pullup devices disabled.

The MCF51EM256 series microcontrollers have the following sources for reset:

- Power-on reset (POR)
- External pin reset (PIN)
- Computer operating properly (COP) timer
- Illegal opcode detect (ILOP)
- Illegal address detect (ILAD)
- Low-voltage detect (LVD)
- Background debug forced reset

Each of these sources, with the exception of the background debug forced reset, has an associated bit in the system reset status register (SRS).

7.3.1 RESETB

When the microcontroller is in stop2 mode and system clocks are shut down, a separate asynchronous path from RESETB to the PMC can be used to wake the device from stop2.

NOTE

This pin does not contain a clamp diode to V_{DD} and must not be driven above V_{DD}.

NOTE

The voltage measured on the internally pulled up $\overline{\text{RESET}}$ pin will not be pulled to V_{DD}. The internal gates connected to this pin are pulled to V_{DD}. The $\overline{\text{RESET}}$ pullup must not be used to pull up components external to the microcontroller.

7.3.2 Computer Operating Properly (COP) Watchdog

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

After any reset, the COP watchdog is enabled (see Section 7.7.6, “System Options 1 (SOPT1) Register,” for additional information). If the COP watchdog is not used in an application, it can be disabled by clearing SOPT1[COPT].

The COP counter is reset by writing 0x55 and 0xAA (in this order) to the address of SRS during the selected timeout period. Writes do not affect the data in the read-only SRS. As soon as the write sequence is done, the COP timeout period is restarted. If the program fails to do this during the time-out period, the microcontroller will reset. Also, if any value other than 0x55 or 0xAA is written to SRS, the microcontroller is immediately reset.

The SOPT1[COPCLKS] field selects the clock source used for the COP timer. The clock source options are either the bus clock or an internal 1 kHz clock source. With each clock source, there are three associated time-outs controlled by SOPT1[COPT]. Table 7-10 summarizes the control functions of the COPCLKS and COPT bits. The COP watchdog defaults to operation from the 1 kHz clock source and the longest time-out (2^{10} cycles).

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

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

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

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

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

7.3.3 Illegal Opcode Detect (ILOP)

The default configuration of the V1 ColdFire core enables the generation of an MCU reset in response to the processor's attempted execution of an illegal instruction (except for the ILLEGAL opcode), illegal line A, illegal line F instruction or the detection of a privilege violation (attempted execution of a supervisor instruction while in user mode).

The attempted execution of the STOP instruction with (SOPT[STOPE] = 0 && SOPT[WAITE] = 0) is treated as an illegal instruction.

The attempted execution of the HALT instruction with XCSR[ENBDM] = 0 is treated as an illegal instruction.

The processor generates a reset in response to any of these events if CPUCR[IRD] = 0. If this configuration bit is set, the processor generates the appropriate exception instead of forcing a reset.

7.3.4 Illegal Address Detect (ILAD)

The default configuration of the V1 ColdFire core enables the generation of an MCU reset in response to any processor-detected address error, bus error termination, RTE format error or fault-on-fault condition.

The processor generates a reset if CPUCR[ARD] = 0. If this configuration bit is set, the processor generates the appropriate exception instead of forcing a reset, or simply halts the processor in response to the fault-on-fault condition.

7.4 Interrupts & Exceptions

The interrupt architecture of ColdFire utilizes a 3-bit encoded interrupt priority level sent from the interrupt controller to the core, providing 7 levels of interrupt requests. Level 7 represents the highest priority interrupt level, while level 1 is the lowest priority. The processor samples for active interrupt requests once per instruction by comparing the encoded priority level against a 3-bit interrupt mask value (I) contained in bits 10:8 of the processor’s status register (SR). If the priority level is greater than the SR[I] field at the sample point, the processor suspends normal instruction execution and initiates interrupt exception processing.

Level 7 interrupts are treated as non-maskable and edge-sensitive within the processor, while levels 1-6 are treated as level-sensitive and may be masked depending on the value of the SR[I] field. For correct

operation, the ColdFire processor requires that, once asserted, the interrupt source remain asserted until explicitly disabled by the interrupt service routine.

During the interrupt exception processing, the CPU does the following tasks in order:

1. Enter supervisor mode,
2. Disable trace mode,
3. Use the vector provided by the INTC when the interrupt was signaled (if CPUCR[IACK] = 0) or explicitly fetches an 8-bit vector from the INTC (if CPUCR[IACK] = 1).

This byte-sized operand fetch during exception processing is known as the interrupt acknowledge (IACK) cycle. The fetched data provides an index into the exception vector table which contains up to 256 addresses (depending upon the specific device), each pointing to the beginning of a specific exception service routine.

In particular, the first 64 exception vectors are reserved for the processor to handle reset, error conditions (access, address), arithmetic faults, system calls, etc. Vectors 64–255 are reserved for interrupt service routines. The MCF51EM series microcontrollers support 36 peripheral interrupt sources and an additional seven software interrupt sources. These are mapped into the standard seven ColdFire interrupt levels, with up to 9 levels of prioritization within a given level by the V1 ColdFire interrupt controller. See Table 7-1 for details.

Once the interrupt vector number has been retrieved, the processor continues by creating a stack frame in memory. For ColdFire, all exception stack frames are two longwords in length, and contain 32 bits of vector and status register data, along with the 32-bit program counter value of the instruction that was interrupted. After the exception stack frame is stored in memory, the processor accesses the 32-bit pointer from the exception vector table using the vector number as the offset, and then jumps to that address to begin execution of the service routine. After the status register is stored in the exception stack frame, the SR[I] mask field is set to the level of the interrupt being acknowledged, effectively masking that level and all lower values while in the service routine.

All ColdFire processors guarantee that the first instruction of the service routine is executed before interrupt sampling is resumed. By making this initial instruction a write to the SR, interrupts can be safely disabled, if required. Optionally, the processor can be configured to automatically raise the mask level to 7 for any interrupt during exception processing by setting CPUCR[IME] = 1.

During the execution of the service routine, the appropriate actions must be performed on the peripheral to negate the interrupt request.

For more information on exception processing, see the *ColdFire Programmer's Reference Manual*. For additional information specific to this device, see Chapter 10, "Interrupt Controller (CF1_INTC)."

7.4.1 External Interrupt Request (IRQ) Pin

External interrupts are managed by the IRQ status and control register, IRQSC. When the IRQ function is enabled, synchronous logic monitors the pin for edge-only or edge-and-level events.

7.4.1.1 Pin Configuration Options

The IRQ pin enable (IRQPE) control bit in IRQSC must be set in order for the IRQ pin to act as the interrupt request (IRQ) input. As an IRQ input, the user can choose the polarity of edges or levels detected (IRQEDG), whether the pin detects edges-only or edges and levels (IRQMOD), and whether an event causes an interrupt or only sets the IRQF flag which can be polled by software (IRQIE).

The IRQ pin, when enabled, defaults to use an internal pull device (IRQPDD = 0), configured as a pullup or pulldown depending on the polarity chosen. If the user desires to use an external pullup or pulldown, the IRQPDD can be set to turn off the internal device.

7.4.1.2 Edge and Level Sensitivity

The IRQMOD control bit re-configures the detection logic so it detects edge events and pin levels. In the edge and level detection mode, the IRQF status flag becomes set when an edge is detected (when the IRQ pin changes from the deasserted to the asserted level), but the flag is continuously set (and cannot be cleared) as long as the IRQ pin remains at the asserted level.

7.4.2 Interrupt Vectors, Sources, and Local Masks

Table 7-1 shows address assignments for reset and interrupt vectors. The vector names shown in this table are the labels used in the Freescale Semiconductor-provided equate file for the MCF51EM256 series microcontrollers. The table is sorted by priority of the sources, with higher-priority sources at the top of the table.

Table 7-1. MCF51EM256 Series Exception and Interrupt Vector Table

Interrupt	Local Enable	Source	Description
KBI1_int	KBI1_SC[KBIE]	KBI1_SC[KBF]	Keyboard Interface
KBI2_int	KBI2_SC[KBIE]	KBI2_SC[KBF]	Keyboard Interface
IRQ_int	IRQ_IRQSC[IRQIE]	IRQ_IRQSC[IRQF]	External Pin Interrupt
PMC_lvd	PMC_LVDIE	PMC_LVDF	Low Voltage Interupt (low voltage than the warning interrupt)
PMC_lvw	PMC_LVWIE	PMC_LVWF	Low Voltage Warning Interrupt. Asserted when the regulator starts to drop out of regulation.
SCI1_rx	SCI1_C2[RIE]	SCI1_S1[RDRF]	Reciever Interrupt
	SCI1_C2[ILIE]	SCI1_S1[IDLE]	Idle Line Interrupt
	SCI1_BDH[LBKDIE]	SCI1_S2[LBKDIF]	LIN Break Detect Interrupt
	SCI1_BDH[RXEDGIE]	SCI1_S2[RXEDGIF]	RxD Input Active Edge Interrupt
SCI1_err	SCI1_C3[ORIE]	SCI1_S1[OR]	Overrun Interrupt
	SCI1_C3[FEIE]	SCI1_S1[FE]	Framing Error Interrupt
	SCI1_C3[NEIE]	SCI1_S1[NF]	Noise Error Interrupt
	SCI1_C3[PEIE]	SCI1_S1[PF]	Parity Error Interrupt

Table 7-1. MCF51EM256 Series Exception and Interrupt Vector Table (continued)

Interrupt	Local Enable	Source	Description
SCI1_tx	SCI1_C2[TCIE]	SCI1_S1[TC]	Transmission Complete Interrupt
	SCI1_C2[TIE]	SCI1_S1[TDRE]	Transmit Data Register Empty Interrupt
SCI2_rx	SCI2_C2[RIE]	SCI2_S1[RDRF]	Reciever Interrupt
	SCI2_C2[ILIE]	SCI2_S1[IDLE]	Idle Line Interrupt
	SCI2_BDH[LBKDIE]	SCI2_S2[LBKDIF]	LIN Break Detect Interrupt
	SCI2_BDH[RXEDGIE]	SCI2_S2[RXEDGIF]	RxD Input Active Edge Interrupt
SCI2_err	SCI2_C3[ORIE]	SCI2_S1[OR]	Overrun Interrupt
	SCI2_C3[FEIE]	SCI2_S1[FE]	Framing Error Interrupt
	SCI2_C3[NEIE]	SCI2_S1[NF]	Noise Error Interrupt
	SCI2_C3[PEIE]	SCI2_S1[PF]	Parity Error Interrupt
SCI2_tx	SCI2_C2[TCIE]	SCI2_S1[TC]	Transmission Complete Interrupt
	SCI2_C2[TIE]	SCI2_S1[TDRE]	Transmit Data Register Empty Interrupt
SCI3_rx	SCI3_C2[RIE]	SCI3_S1[RDRF]	Reciever Interrupt
	SCI3_C2[ILIE]	SCI3_S1[IDLE]	Idle Line Interrupt
	SCI3_BDH[LBKDIE]	SCI3_S2[LBKDIF]	LIN Break Detect Interrupt
	SCI3_BDH[RXEDGIE]	SCI3_S2[RXEDGIF]	RxD Input Active Edge Interrupt
SCI3_err	SCI3_C3[ORIE]	SCI3_S1[OR]	Overrun Interrupt
	SCI3_C3[FEIE]	SCI3_S1[FE]	Framing Error Interrupt
	SCI3_C3[NEIE]	SCI3_S1[NF]	Noise Error Interrupt
	SCI3_C3[PEIE]	SCI3_S1[PF]	Parity Error Interrupt
SCI3_tx	SCI3_C2[TCIE]	SCI3_S1[TC]	Transmission Complete Interrupt
	SCI3_C2[TIE]	SCI3_S1[TDRE]	Transmit Data Register Empty Interrupt
SPI1_int	SPI1_C1[SPIE]	SPI1_S[MODF]	Mode Fault Interrupt
		SPI1_S[SPRF]	Receive Buffer Full Interrupt
	SPI1_C1[SPTIE]	SPI1_S[SPTEF]	Transmit Buffer Empty Interrrupt
	SPI1_C2[SPMIE]	SPI1_S[SPMF]	Receive Data Match Interrupt
	SPI1_C3[RNFULLIEN]	SPI1_S[RNEARFF]	Receive FIFO Nearly Full Interrupt
	SPI1_C3[TNEARIEN]	SPI1_S[TNEAREF]	Transmit FIFO Nearly Empty Interrupt
SPI2_int	SPI2_C1[SPIE]	SPI2_S[MODF]	Mode Fault Interrupt
		SPI2_S[SPRF]	Receive Buffer Full Interrupt
	SPI2_C1[SPTIE]	SPI2_S[SPTEF]	Transmit Buffer Empty Interrrupt

Table 7-1. MCF51EM256 Series Exception and Interrupt Vector Table (continued)

Interrupt	Local Enable	Source	Description
SPI3_int	SPI3_C1[SPIE]	SPI3_S[MODF]	Mode Fault Interrupt
		SPI3_S[SPRF]	Receive Buffer Full Interrupt
	SPI3_C1[SPTIE]	SPI3_S[SPTEF]	Transmit Buffer Empty Interrrupt
IIC_int	IIC_CR1[IICIE]	IIC_SR[IICIF]	Complete 1-byte tranfer (TCF) Interrupt Match of received calling address (IAAS) Interrupt Arbitration Lost (ARBL) Interrupt SMBus Timeout (SLTF) Interrupt
ADC1_COCO	ADC1_SC1[AIEN]	ADC1_SC1[COCO]	Conversion Complete Flag
ADC2_COCO	ADC2_SC1[AIEN]	ADC2_SC1[COCO]	Conversion Complete Flag
ADC3_COCO	ADC3_SC1[AIEN]	ADC3_SC1[COCO]	Conversion Complete Flag
ADC4_COCO	ADC4_SC1[AIEN]	ADC4_SC1[COCO]	Conversion Complete Flag
CMP1_CMP	CMP1_CS[ACIEN]	CMP1_CS[ACMPF]	Comparator Interrupt
CMP2_CMP	CMP2_CS[ACIEN]	CMP2_CS[ACMPF]	Comparator Interrupt
MTIM1_ovfl	MTIM1_TOIE	MTIM1_TOF	MTIM8 Overflow Interrupt
MTIM2_ovfl	MTIM2_TOIE	MTIM2_TOF	MTIM8 Overflow Interrupt
MTIM3_ovfl	MTIM3_TOIE	MTIM3_TOF	MTIM8 Overflow Interrupt
TPM_TOF	TPM_SC[TOIE]	TPM_SC[TOF]	Counter Overflow
TPM_ch0	TPM_C0SC[CH0IE]	TPM_C0SC[CH0F]	An input capture or output compare event took place on channel 0
TPM_ch1	TPM_C1SC[CH1IE]	TPM_C1SC[CH1F]	An input capture or output compare event took place on channel 1
IRTC_INT	IRTC_IRTCIER	IRTC_IRTCISR	composite irtc interrupt
LCD_LCD	LCD_LCDC1[LCDIEN]	LCD_LCDS[LCDIF]	LCD Interrupt
FTSR1_int	FTSR1_FCNFG[CBEIE, CCIE]	FTSR1_FSTAT[FCBEF, FCCF]	composite FTSR interrupt
FTSR2_int	FTSR2_FCNFG[CBEIE, CCIE]	FTSR2_FSTAT[FCBEF, FCCF]	composite FTSR interrupt
PDB	PDBSC[IE]	PDBSC[IF]	PDB Interrupt
PDB_ERR	N/A	PDBCH*n*CR[ERRA, ERRB]	PDB sequence error interrupt (always enabled)

Table 7-2. Unsupported FTSR Interrupts

Interrupt	FTSR_Local Enable	FTSR_Source	Description
FTSR_emptybuf	FTSR_FCNFG[CBEIE]	FTSR_FSTAT[FCBEF]	Flash Command Buffer Empty Interrupt
FTSR_coco	FTSR_FCNFG[CCIE]	FTSR_FSTAT[FCCF]	Flash Command Complete Interrupt

Not shown in Table 7-2 are the standard set of ColdFire exceptions, many of which apply to this device. These are listed in Table 7-3.

Additional details with regard to interrupt processing are present in Chapter 10, “Interrupt Controller (CF1_INTC)”.

The CPU configuration register (CPUCR) within the supervisor programming model will allow the users to determine if specific ColdFire exception conditions are to generate a normal exception or a system reset. The default state of the CPUCR will force a system reset for any of the exception types listed in Table 7-3.

Table 7-3. Exception Types to System Reset

Vector	Exception	Reset Disabled via CPUCR	Reported using SRS
64–108	I/O Interrupts	N/A	—
61	Unsupported instruction	N/A	—
47	Trap #15	N/A	—
46	Trap #14	N/A	—
45	Trap #13	N/A	—
44	Trap #12	N/A	—
43	Trap #11	N/A	—
42	Trap #10	N/A	—
41	Trap #9	N/A	—
40	Trap #8	N/A	—
39	Trap #7	N/A	—
38	Trap #6	N/A	—
37	Trap #5	N/A	—
36	Trap #4	N/A	—
35	Trap #3	N/A	—
34	Trap #2	N/A	—
33	Trap #1	N/A	—
32	Trap #0	N/A	—
24	Spurious IRQ	N/A	—
14	Format error	CPUCR[30]	ilad
12	Debug breakpoint IRQ	N/A	—
11	Illegal LineF	CPUCR[30]	ilop
10	Illegal LineA	CPUCR[30]	ilop
9	Trace	N/A	—
8	Privileged Violation	CPUCR[30]	ilop
4	Illegal instruction	CPUCR[30]	ilop[1]

Table 7-3. Exception Types to System Reset (continued)

Vector	Exception	Reset Disabled via CPUCR	Reported using SRS
3	Address error	CPUCR[31]	ilad
2	Access error	CPUCR[31]	ilad
n/a	Flt-on-Flt Halt	CPUCR[31]	ilad

[1] The execution of the ILLEGAL instruction (0x4AFC) always generates an illegal instruction exception, regardless of the state of CPUCR[30].

7.5 Low-Voltage Detect (LVD) System

The MCF51EM256 series microcontrollers include a system to protect against low voltage conditions to protect memory contents and control microcontroller system states during supply voltage variations. The system is comprised of a power-on reset (POR) circuit and a LVD circuit with a user-selectable trip voltage, either high (V_{LVDH}) or low (V_{LVDL}). The LVD circuit is enabled when the SPMSC1[LVDE] bit is set and the trip voltage is selected by the SPMSC3[LVDV] bit. The LVD is disabled upon entering stop2 or stop3 modes unless the LVDSE bit is set. If both LVDE and LVDSE are set when the STOP instruction is processed, the device will enter STOP4 mode. The LVD can be left enabled in this mode.

7.5.1 Power-On Reset Operation

When power is initially applied to the microcontroller, or when the supply voltage drops below the power-on reset re-arm voltage level, V_{POR}, the POR circuit will cause a reset condition. As the supply voltage rises, the LVD circuit will hold the microcontroller in reset until the supply has risen above the LVD low threshold, V_{LVDL}. Both the POR bit and the LVD bit in SRS are set following a POR.

7.5.2 LVD Reset Operation

The LVD can be configured to generate a reset upon detection of a low voltage condition by setting LVDRE to 1. The low voltage detection threshold is determined by the LVDV bit. After an LVD reset has occurred, the LVD system will hold the microcontroller in reset until the supply voltage has risen above the low voltage detection threshold. The LVD bit in the SRS register is set following either an LVD reset or POR.

7.5.3 LVD Interrupt Operation

When a low voltage condition is detected and the LVD circuit is configured using SPMSC1 for interrupt operation (LVDE set, LVDIE set, and LVDRE clear), then LVDF in SPMSC1 will be set and an LVD interrupt request will occur. The LVDF bit is cleared by writing a 1 to the LVDACK bit in SPMSC1.

7.5.4 Low-Voltage Warning (LVW) Interrupt Operation

The LVD system has a low voltage warning flag (LVWF) to indicate to the user that the supply voltage is approaching, but is above, the LVD voltage. The LVW also has an interrupt associated with it, enabled by setting the SPMSC3[LVWIE] bit. If enabled, an LVW interrupt request will occur when the LVWF is set.

LVWF is cleared by writing a 1 to the SPMSC3[LVWACK] bit. There are two user-selectable trip voltages for the LVW, one high (V_{LVWH}) and one low (V_{LVWL}). The trip voltage is selected by SPMSC3[LVWV] bit.

7.6 Peripheral Clock Gating

The MCF51EM series microcontroller includes a clock gating system to manage the bus clock sources to the individual peripherals. Using this system, the user can enable or disable the bus clock to each of the peripherals at the clock source, eliminating unnecessary clocks to peripherals which are not in use; thereby reducing the overall run and wait mode currents.

Out of reset, all peripheral clocks will be enabled. For lowest possible run or wait currents, user software should disable the clock source to any peripheral not in use. The actual clock will be enabled or disabled immediately following the write to the clock gating control registers (SCGC1, SCGC2, SCGC3, SCGC4). Any peripheral with a gated clock can not be used unless its clock is enabled. Writing to the registers of a peripheral with a disabled clock has no effect.

NOTE

User software should disable the peripheral before disabling the clocks to the peripheral. When clocks are re-enabled to a peripheral, the peripheral registers need to be re-initialized by user software.

In stop modes, the bus clock is disabled for all gated peripherals, regardless of the settings in the SCGC1, SCGC2, SCGC3 and SCGC4 registers.

7.7 Reset, Interrupt, and System Control Registers and Control Bits

One 8-bit register in the direct page register space and eight 8-bit registers in the high-page register space are related to reset and interrupt systems.

Refer to Section 3.2, “Detailed Register Addresses and Bit Assignments,” for the absolute address assignments for all registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

Some control bits in the SOPT1 and SPMSC2 registers are related to modes of operation. Although brief descriptions of these bits are provided here, the related functions are discussed in greater detail in Chapter 6, “Modes of Operation.”

7.7.1 Interrupt Pin Request Status and Control Register (IRQSC)

This direct page register includes status and control bits which are used to configure the IRQ function, report status, and acknowledge IRQ events.

	7	6	5	4	3	2	1	0
R	0	IRQPDD	IRQEDG	IRQPE	IRQF	0	IRQIE	IRQMOD
W						IRQACK		
Reset	0	0	0	0	0	0	0	0

Figure 7-1. Interrupt Request Status and Control Register (IRQSC)

Table 7-4. IRQSC Field Descriptions

Field	Description
7	Reserved, should be cleared.
6 IRQPDD	**Interrupt Request (IRQ) Pull Device Disable**— This read/write control bit is used to disable the internal pullup/pulldown device when the IRQ pin is enabled (IRQPE = 1) allowing for an external device to be used. 0 IRQ pull device enabled if IRQPE = 1. 1 IRQ pull device disabled if IRQPE = 1. This bit will override the pullup enable logic in the GPIO controls when IRQ owns the pin.
5 IRQEDG	**Interrupt Request (IRQ) Edge Select** — This read/write control bit is used to select the polarity of edges or levels on the IRQ pin that cause IRQF to be set. The IRQMOD control bit determines whether the IRQ pin is sensitive to both edges and levels or only edges. When IRQEDG = 1 and the internal pull device is enabled, the pullup device is reconfigured as an optional pulldown device. 0 IRQ is falling edge or falling edge/low-level sensitive. 1 IRQ is rising edge or rising edge/high-level sensitive.
4 IRQPE	**IRQ Pin Enable** — This read/write control bit enables the IRQ pin function. When this bit is set the IRQ pin can be used as an external interrupt request. Note that the IRQ pin function must also be enabled via the MC[1] registers. 0 IRQ pin function is disabled. 1 IRQ pin function is enabled.
3 IRQF	**IRQ Flag** — This read-only status bit indicates when an interrupt request event has occurred. 0 No IRQ request. 1 IRQ event detected.
2 IRQACK	**IRQ Acknowledge** — This write-only bit is used to acknowledge interrupt request events (write 1 to clear IRQF). Writing 0 has no meaning or effect. Reads always return 0. If edge-and-level detection is selected (IRQMOD = 1), IRQF cannot be cleared while the IRQ pin remains at its asserted level.
1 IRQIE	**IRQ Interrupt Enable** — This read/write control bit determines whether IRQ events generate an interrupt request. 0 Interrupt request when IRQF set is disabled (use polling). 1 Interrupt requested whenever IRQF = 1.
0 IRQMOD	**IRQ Detection Mode** — This read/write control bit selects either edge-only detection or edge-and-level detection. The IRQEDG control bit determines the polarity of edges and levels that are detected as interrupt request events. See Section 7.4.1.2, “Edge and Level Sensitivity” for more details. 0 IRQ event on falling edges or rising edges only. 1 IRQ event on falling edges and low levels or on rising edges and high levels.

[1] MC = Port Mux Control

7.7.2 System Power Management Status and Control 1 Register (SPMSC1)

This high-page register contains status and control bits to support the low voltage detect function, and to enable the bandgap voltage reference for use by the ADC module. This register should be written during the user's reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

	7	6	5	4	3	2	1	0
R	LVDF[1]	0	LVDIE	LVDRE[2]	LVDSE	LVDE[2]	BGBDS	BGBE
W		LVDACK						
Reset:	0	0	0	1	1	1	0	0

1 LVDF will be set in the case when V_{Supply} transitions below the trip point or after reset and V_{Supply} is already below V_{LVD}.

2 This bit can be written only one time after reset. Additional writes are ignored.

Figure 7-2. System Power Management Status and Control 1 Register (SPMSC1)

Table 7-5. SPMSC1 Field Descriptions

Field	Description
7 LVDF	**Low-Voltage Detect Flag** — The LVDF bit indicates the low-voltage detect status. 0 Low-voltage detect is not present. 1 Low-voltage detect is present or was present.
6 LVDACK	**Low-Voltage Detect Acknowledge** — If LVDF = 1, a low-voltage condition has occurred. To acknowledge this low-voltage detect, write 1 to LVDACK, which will automatically clear LVDF to 0 if the low-voltage detect is no longer present.
5 LVDIE	**Low-Voltage Detect Interrupt Enable** — This bit enables hardware interrupt requests for LVDF. 0 Hardware interrupt disabled (use polling). 1 Request a hardware interrupt when LVDF = 1.
4 LVDRE	**Low-Voltage Detect Reset Enable** — This write-once bit enables LVDF events to generate a hardware reset (provided LVDE = 1). 0 LVDF does not generate hardware resets. 1 Force an MCU reset when an enabled low-voltage detect event occurs.
3 LVDSE	**Low-Voltage Detect Stop Enable** — Provided LVDE = 1, this read/write bit determines whether the low-voltage detect function operates when the MCU is in stop mode. 0 Low-voltage detect disabled during stop mode. 1 Low-voltage detect enabled during stop mode.
2 LVDE	**Low-Voltage Detect Enable** — This write-once bit enables low-voltage detect logic and qualifies the operation of other bits in this register. 0 LVD logic disabled. 1 LVD logic enabled.
1 BGBDS	**Bandgap Buffer Drive Select** — This bit is used to select the high drive mode of the bandgap buffer. 0 Bandgap buffer enabled in low drive mode if BGBE = 1. 1 Bandgap buffer enabled in high drive mode if BGBE = 1.
0 BGBE	**Bandgap Buffer Enable** — This bit enables an internal buffer for the bandgap voltage reference for use by the ADC module on one of its internal channels. 0 Bandgap buffer disabled. 1 Bandgap buffer enabled.

7.7.3 System Power Management Status and Control 2 Register (SPMSC2)

This high-page register contains status and control bits to configure the low power run and wait modes as well as configure the stop mode behavior of the microcontroller. See Section 6.7.2, "Low-Power Wait Mode (LPwait)," for more information.

SPMSC2 is not reset when exiting from stop2.

	7	6	5	4	3	2	1	0
R	LPR	LPRS	LPWUI	0	PPDF	0	PPDE[1]	PPDC
W						PPDACK		
Reset:	0	0	0	0	—	0	0	0

[1] PPDE is a write-once bit that can be used to permanently disable the PPDC bit.

Figure 7-3. System Power Management Status and Control 2 Register (SPMSC2)

Table 7-6. SPMSC2 Bit Field Descriptions

Field	Description
7 LPR	Low-Power Regulator Control. The LPR bit controls entry into the low-power run and low-power wait modes in which the voltage regulator is put into standby. This bit cannot be set if PPDC=1. If PPDC and LPR are set in a single write instruction, only PPDC is actually set. LPR is cleared when an interrupt occurs in low-power mode and the LPWUI bit is 1. 0 Low-power run and low-power wait modes are disabled. 1 Low-power run and low-power wait modes are requested.
6 LPRS	Low-Power Regulator Status. This read-only status bit indicates that the voltage regulator has entered into standby for the low-power run or wait mode. 0 The voltage regulator is not currently in standby. 1 The voltage regulator is currently in standby.
5 LPWUI	Low-Power Wake-Up on Interrupt. This bit controls whether or not the voltage regulator exits standby when any active MCU interrupt occurs. 0 The voltage regulator remains in standby on an interrupt. 1 The voltage regulator exits standby on an interrupt. LPR is cleared.
4	**RESERVED**
3 PPDF	**Partial Power-Down Flag** — This read-only status bit indicates that the microcontroller has recovered from stop2 mode. 0 Microcontroller has not recovered from stop2 mode. 1 Microcontroller recovered from stop2 mode.
2 PPDACK	**Partial Power-Down Acknowledge** — Writing a 1 to PPDACK clears the PPDF bit.

Table 7-6. SPMSC2 Bit Field Descriptions

Field	Description
1 PPDE	Partial Power-Down Enable. The write-once PPDE bit can be used to lockout the partial power-down feature. This is a write-once bit. 0 Partial power-down is not enabled. 1 Partial power-down is enabled and controlled via the PPDC bit.
0 PPDC	**Partial Power-Down Control** — The PPDC bit controls which power-down mode is selected. This bit cannot be set if LPR = 1. If PPDC and LPR are set in a single write instruction, only PPDC will actually be set. PPDE must be set in order for PPDC to be set. 0 Stop3 low power mode enabled. 1 Stop2 partial power-down mode enabled.

1 See data sheet for minimum and maximum values.

7.7.4 System Power Management Status and Control 3 Register (SPMSC3)

This register reports the status of the low voltage warning function and is used to select the low voltage detect trip voltage. SPMSC3 is not reset when exiting from stop2.

	7	6	5	4	3	2	1	0
R	LVWF	0	LVDV	LVWV	LVWIE	0	0	0
W		LVWACK						
POR:	0[1]	0	0	0	0	0	0	0
LVR:	0[1]	0	U	U	0	0	0	0
Any other reset:	0[1]	0	U	U	0	0	0	0

[1] LVWF is set when V_{Supply} transitions below the trip point or after reset and V_{Supply} is already below V_{LVW}.

Figure 7-4. System Power Management Status and Control 3 Register (SPMSC3)

Table 7-7. SPMSC3 Bit Field Descriptions

Field	Description
7 LVWF	Low-Voltage Warning Flag. The LVWF bit indicates the low voltage warning status. 0 Low voltage warning not present. 1 Low voltage warning is present or was present.
6 LVWACK	Low-Voltage Warning Acknowledge. Writing a 1 to LVWACK clears LVWF if a low voltage warning is not present.
5 LVDV	Low-Voltage Detect Voltage Select. The LVDV bit selects the LVD trip point voltage (V_{LVD}). 0 Low trip point selected ($V_{LVD} = V_{LVDL}$). 1 High trip point selected ($V_{LVD} = V_{LVDH}$).
4 LVWV	Low-Voltage Warning Voltage Select. The LVWV bit selects the LVW trip point voltage (V_{LVW}). 0 Low trip point selected ($V_{LVW} = V_{LVWL}$). 1 High trip point selected ($V_{LVW} = V_{LVWH}$).

Table 7-7. SPMSC3 Bit Field Descriptions

Field	Description
3 LVWIE	Low-Voltage Warning Interrupt Enable. This bit enables hardware interrupt requests for LVWF. 0 Hardware interrupt disabled (use polling). 1 Request a hardware interrupt when LVWF is set.
2–0	Reserved, should be cleared.

Table 7-8. LVD and LVW Trip Point Typical Values

LVDV:LVWV	LVW Trip Point	LVD Trip Point
00	V_{LVWL} = 2.15	V_{LVDL} = 1.84
01	V_{LVWH} = 2.6	
10 Not Recommended	V_{LVWL} = 2.15	V_{LVDH} = 2.33
11	V_{LVWH} = 2.6	

7.7.5 System Reset Status Register (SRS)

This register includes read-only status flags to indicate the source of the most recent reset. When a debug host forces reset by setting CSR2[BDFR], none of the status bits in SRS will be set. Writing any value to this register address clears the COP watchdog timer without affecting the contents of this register. The reset state of these bits depends on what caused the microcontroller to reset.

	7	6	5	4	3	2	1	0
R	POR	PIN	COP	ILOP	ILAD	0	LVD	0
W	Writing any value to SRS address clears COP watchdog timer.							
POR:	1	0	0	0	0	0	1	0
LVD:	u	0	0	0	0	0	1	0
Any other reset:	0	Note[1]	Note[1]	Note[1]	Note[1]	0	0	0

[1] Any of these reset sources that are active at the time of reset entry will cause the corresponding bit(s) to be set; bits corresponding to sources that are not active at the time of reset entry will be cleared.

Figure 7-5. System Reset Status (SRS)

Table 7-9. SRS Bit Field Descriptions

Field	Description
7 POR	**Power-On Reset** — Reset was caused by the power-on detection logic. Because the internal supply voltage was ramping up at the time, the low-voltage reset (LVD) status bit is also set to indicate that the reset occurred while the internal supply was below the LVD threshold. 0 Reset not caused by POR. 1 POR caused reset.
6 PIN	**External Reset Pin** — Reset was caused by an active-low level on the external reset pin. 0 Reset not caused by external reset pin. 1 Reset came from external reset pin.
5 COP	**Computer Operating Properly (COP) Watchdog** — Reset was caused by the COP watchdog timer timing out. This reset source can be blocked by SOPT1[COPT] = 0b00. 0 Reset not caused by COP timeout. 1 Reset caused by COP timeout.
4 ILOP	**Illegal Opcode** — Reset was caused by an attempt to execute an unimplemented or illegal opcode. This includes any illegal instruction [except the ILLEGAL (0x4AFC) opcode] or a privilege violation (execution of a privileged instruction in user mode). The STOP instruction is considered illegal if stop is disabled by ((SOPT[STOPE] = 0) && (SOPT[WAITE] = 0)). The HALT instruction is considered illegal if the BDM interface is disabled by XCSR[ENBDM] = 0. 0 Reset not caused by an illegal opcode. 1 Reset caused by an illegal opcode.
3 ILAD	**Illegal Address** — Reset was caused by the processor's attempted access of an illegal address in the memory map, an address error, an RTE format error or the fault-on-fault condition. All the illegal address resets are enabled when CPUCR[ARD] = 0. When CPUCR[ARD] = 1, then the appropriate processor exception is generated instead of the reset, or if a fault-on-fault condition is reached, the processor simply halts. 0 Reset not caused by an illegal access. 1 Reset caused by an illegal access.
2 RESERVED	RESERVED
1 LVD	**Low Voltage Detect** — If the LVD is enabled with LVDRE set, and the supply drops below the LVD trip voltage, an LVD reset will occur. This bit is also set by POR. 0 Reset not caused by LVD trip or POR. 1 Reset caused by LVD trip or POR.
0 RESERVED	RESERVED

7.7.6 System Options 1 (SOPT1) Register

This high-page register has five write-once bits and one write anytime bit. For the write-once bits, only the first write after reset is honored. All bits in the register can be read at any time. Any subsequent attempt to write a write-once bit is ignored to avoid accidental changes to these sensitive settings. SOPT1 must be written to during the reset initialization program to set the desired controls, even if the desired settings are the same as the reset settings.

	7	6	5	4	3	2	1	0
R	0	0	STOPE[1]	WAITE	COPT[1]		COPCLKS[1]	COPW[1]
W								
Reset:	0	0	0	1	1	1	0	0

[1] These bits can be written only one time after reset. Subsequent writes are ignored.

Figure 7-6. System Options 1 (SOPT1) Register

Table 7-10. SOPT1 Bit Field Descriptions

Field	Description
7	Reserved, must be cleared.
6	RESERVED
5 STOPE	**Stop Mode Enable** — This write-once bit is used to enable stop mode. If both stop and wait modes are disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset may be generated depending on CPUCR[IRD].
4 WAITE	**WAIT Mode Enable** — This write-anytime bit is used to enable WAIT mode. If both stop and wait modes are disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset may be generated depending on CPUCR[IRD].
3–2 COPT	**COP Watchdog Timeout** — These write-once bits select the timeout period of the COP. COPT along with SOPT1[COPCLKS] defines the COP timeout period as described in Table 7-11.
1 COPCLKS	**COP Watchdog Clock Select** — This write-once bit selects the clock source of the COP watchdog. 0 Internal 1 kHz clock is source to COP. 1 Bus clock is source to COP.
0 COPW	**COP Window Mode** — This write-once bit specifies whether the COP operates in Normal or Window mode. In Window mode, the 0x55–0xAA write sequence to the SRS register must occur within the last 25% of the selected period; any write to the SRS register during the first 75% of the selected period resets the microcontroller. 0 Normal mode 1 Window mode

Table 7-11. COP Configuration Details

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

1 Windowed COP operation requires the user to clear the COP timer in the last 25% of the selected timeout period. This column displays the minimum number of clock counts required before the COP timer can be reset when in windowed COP mode (SOPT1[COPW] = 1).

2 Values shown in milliseconds based on t_{LPO} = 1 ms.

7.7.7 System Options 2 Register (SOPT2)

This register contains bits that control the PMC LVD trim. This is a reserved register and can not be written by application code during normal operation.

	7	6	5	4	3	2	1	0
R	0	0	0	PMC_LVD_TRIM				
W								
Reset:	0	0	0	—	—	—	—	—

Figure 7-7. System Options 2 Register (SOPT2)

Table 7-12. SOPT2 Bit Field Descriptions

Field	Description
7–5	Reserved, should be cleared.
4–0 PMC_LVD_TRIM	**PMC LVD TRIM.** Reserved, Application code must not write these bits.

7.7.8 System Device Identification Register (SDIDH, SDIDL)

These high-page read-only registers are included so host development systems can identify the ColdFire derivative. This allows the development software to recognize where specific memory blocks, registers, and control bits are located in a target microcontroller.

Additional configuration information about the ColdFire core and memory system is loaded into the 32-bit D0 (core) and D1 (memory) registers at reset. This information can be stored into memory by the system startup code for later use by configuration-sensitive application code. See Section 8.3.3.14, “Reset Exception” for more information.

	7	6	5	4	3	2	1	0
R	REV				ID11	ID10	ID9	ID8
W								
Reset:	—	—	—	—	1	1	0	0

Figure 7-8. System Device Identification Register — High (SDIDH)

Table 7-13. SDIDH Bit Field Descriptions

Field	Description
7–4 REV	**Revision Number.** This field indicates the chip revision number.
3–0 ID[11:8]	**Part Identification Number** — Each derivative in the ColdFire family has a unique identification number. The MCF51EM series microcontrollers have these bits hard coded to the value 0x**C**. See also ID bits in Table 7-14.

	7	6	5	4	3	2	1	0
R	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
W								
Reset:	0	0	0	0	0	0	0 or 1	0 or 1

Figure 7-9. System Device Identification Register — Low (SDIDL)

Table 7-14. SDIDL Bit Field Descriptions

Field	Description
7–0 ID[7–0]	**Part Identification Number** — Each derivative in the ColdFire family has a unique identification number. MCF51EM256 Series devices have the following values in these fields: 0x01 = MCF51EM32 development device 0x02 = MCF51EM256 / MCF51EM128

7.7.9 System Clock Gating Control 1 Register (SCGC1)

This register contains control bits to enable or disable the bus clock to the SCI, IIC and ADC modules on the chip. Gating off the clocks to unused peripherals is used to reduce the microcontroller's run and wait currents. See Section 7.6, "Peripheral Clock Gating," for more information.

	7	6	5	4	3	2	1	0
R W	ADC4	ADC3	ADC2	ADC1	IIC	SCI3	SCI2	SCI1
Reset:	1	1	1	1	1	0	1	1

Figure 7-10. System Clock Gating Control 1 Register (SCGC1)

Table 7-15. SCGC1 Bit Field Descriptions

Field	Description
7 ADC4	**ADC 4 Clock Gate Control** — This bit controls the clock gate to the ADC 4 module. 0 Bus clock to the ADC 4 module is disabled. 1 Bus clock to the ADC 4vmodule is enabled.
6 ADC3	**ADC 3 Clock Gate Control** — This bit controls the clock gate to the ADC 3 module. 0 Bus clock to the ADC 3 module is disabled. 1 Bus clock to the ADC 3 module is enabled.

Table 7-15. SCGC1 Bit Field Descriptions

Field	Description
5 ADC2	**ADC 2 Clock Gate Control** — This bit controls the clock gate to the ADC 2 module. 0 Bus clock to the ADC 2 module is disabled. 1 Bus clock to the ADC 2 module is enabled.
4 ADC1	**ADC 1 Clock Gate Control** — This bit controls the clock gate to the ADC 1 module. 0 Bus clock to the ADC 1 module is disabled. 1 Bus clock to the ADC 1 module is enabled.
3 IIC	**IIC2 Clock Gate Control** — This bit controls the clock gate to the IIC2 module. 0 Bus clock to the IIC2 module is disabled. 1 Bus clock to the IIC2 module is enabled.
2 SCI3	**SCI3 Clock Gate Control** — This bit controls the clock gate to the SCI3 module. 0 Bus clock to the SCI3 module is disabled. 1 Bus clock to the SCI3 module isenabled.
1 SCI2	**SCI2 Clock Gate Control** — This bit controls the clock gate to the SCI2 module. 0 Bus clock to the SCI2 module is disabled. 1 Bus clock to the SCI2 module isenabled.
0 SCI1	**SCI1 Clock Gate Control** — This bit controls the clock gate to the SCI1 module. 0 Bus clock to the SCI1 module is disabled. 1 Bus clock to the SCI1 module isenabled.

7.7.10 System Clock Gating Control 2 Register (SCGC2)

This register contains control bits to enable or disable the bus clock to the SPI, LCD, IRQ, VREF and PRACMP modules. Gating off the clocks to unused peripherals is used to reduce the microcontroller's run and wait currents. See Section 7.6, "Peripheral Clock Gating," for more information.

	7	6	5	4	3	2	1	0
R W	CMP2	CMP1	VREF	IRQ	LCD	SPI3[1]	SPI2	SPI1
Reset:	1	1	1	1	1	—	1	1

[1] This bit is reset to 1 on 100-pin package and 0 on 80-pin package.

Figure 7-11. System Clock Gating Control 2 Register (SCGC2)

Table 7-16. SCGC2 Bit Field Descriptions

Field	Description
7 CMP2	**PRACMP2 Clock Gate Control** — This bit controls the clock gate to the PRACMP2 module. 0 Bus clock to the PRACMP2 module is disabled. 1 Bus clock to the PRACMP2 module isenabled.
6 CMP1	**PRACMP1 Clock Gate Control** — This bit controls the clock gate to the PRACMP1 module. 0 Bus clock to the PRACMP1 module is disabled. 1 Bus clock to the PRACMP1 module isenabled.

Table 7-16. SCGC2 Bit Field Descriptions

Field	Description
5 VREF	**VREF Clock Gate Control** — This bit controls the bus clock gate to the VREF module. 0 Bus clock to VREF is disabled. 1 Bus clock to VREF is enabled.
4 IRQ	**IRQ Clock Gate Control** — This bit controls the clock gate to the IRQ module. 0 Bus clock to theIRQ module is disabled. 1 Bus clock to the IRQ module isenabled.
3 LCD	**LCD Clock Gate Control** — This bit controls the clock gate to the LCD module. 0 Bus clock to the LCD module is disabled. 1 Bus clock to the LCD module is enabled.
2 SPI3	**SPI3 Clock Gate Control** — This bit controls the bus clock gate to the SPI3 module. 0 Bus clock to the SPI3 module is disabled. 1 Bus clock to the SPI3 module is enabled.
1 SPI2	**SPI2 Clock Gate Control** — This bit controls the bus clock gate to the SPI2 module. 0 Bus clock to the SPI2 module is disabled. 1 Bus clock to the SPI2 module is enabled.
0 SPI1	**SPI1 Clock Gate Control** — This bit controls the bus clock gate to the SPI1 module. 0 Bus clock to the SPI1 module is disabled. 1 Bus clock to the SPI1 module is enabled.

7.7.11 System Clock Gating Control 3 Register (SCGC3)

This register contains control bits to enable or disable the bus clock to the GPIO and KBI modules. Gating off the clocks to unused peripherals is used to reduce the microcontroller's run and wait currents. See Section 7.6, "Peripheral Clock Gating," for more information.

	7	6	5	4	3	2	1	0
R W	KBI2	KBI1	PTF	PTE	PTD	PTC	PTB	PTA
Reset:	1	1	1	1	1	1	1	1

Figure 7-12. System Clock Gating Control 3 Register (SCGC3)

Table 7-17. SCGC3 Bit Field Descriptions

Field	Description
7 KBI2	**KBI2 Clock Gate Control** — This bit controls the bus clock gate to the KBI2 module. 0 Bus clock to the KBI2 module is disabled. 1 Bus clock to the KBI2 module is enabled.
6 KBI1	**KBI1 Clock Gate Control** — This bit controls the bus clock gate to the KBI1 module. 0 Bus clock to the KBI1 module is disabled. 1 Bus clock to the KBI1 module is enabled.
5 PTF	**PTF Clock Gate Control** — This bit controls the clock gate to the PTF module. 0 Bus clock to the PTF module is disabled. 1 Bus clock to the PTF module is enabled.

Table 7-17. SCGC3 Bit Field Descriptions

Field	Description
4 PTE	**PTE Clock Gate Control** — This bit controls the clock gate to the PTE module. 0 Bus clock to the PTE module is disabled. 1 Bus clock to the PTE module is enabled.
3 PTD	**PTD Clock Gate Control** — This bit controls the clock gate to the PTD module. 0 Bus clock to the PTD module is disabled. 1 Bus clock to the PTD module is enabled.
2 PTC	**PTC Clock Gate Control** — This bit controls the clock gate to the PTC module. 0 Bus clock to the PTC module is disabled. 1 Bus clock to the PTC module is enabled.
1 PTB	**PTB Clock Gate Control** — This bit controls the clock gate to the PTB module. 0 Bus clock to the PTB module is disabled. 1 Bus clock to the PTB module is enabled.
0 PTA	**PTA Clock Gate Control** — This bit controls the clock gate to the PTA module. 0 Bus clock to the PTA module is disabled. 1 Bus clock to the PTA module is enabled.

7.7.12 System Clock Gating Control 4 Register (SCGC4)

This register contains control bits to enable or disable the bus clock to MTIM, Port Mux Control, TPM , CRC and PDB modules. Gating off the clocks to unused peripherals is used to reduce the microcontroller's run and wait currents. See Section 7.6, "Peripheral Clock Gating," for more information.

	7	6	5	4	3	2	1	0
R	1	PM	CRC	TPM	PDB	MTIM3	MTIM2	MTIM1
W								
Reset:	1	1	1	1	1	1	1	1

Figure 7-13. System Clock Gating Control 3 Register (SCGC4)

Table 7-18. SCGC4 Bit Field Descriptions

Field	Description
7	RESERVED
6 PM	**Port Mux Control** — This bit controls the clock gate to the Port Mux Control module. 0 Bus clock to the Port Mux Control module is disabled. 1 Bus clock to the Port Mux Control module is enabled. This bit can be set to zero once the mux controls have been programmed as desired. This bit affects programming of the mux controls only. The actual data paths to/from pins are unaffected.
5 CRC	**CRC** — This bit controls the clock gate to the CRC module. 0 Bus clock to the CRC module is disabled. 1 Bus clock to the CRC module is enabled.
4 TPM	**TPM Clock Gate Control** — This bit controls the clock gate to the TPM module. 0 Bus clock to the TPM module is disabled. 1 Bus clock to the TPM module is enabled.

Table 7-18. SCGC4 Bit Field Descriptions

3 PDB	**PDB Clock Gate Control** — This bit controls the clock gate to the PDB module. 0 Bus clock to the PDB module is disabled. 1 Bus clock to the PDB module is enabled.
2 MTMI3	**MTMI3 Clock Gate Control** — This bit controls the clock gate to the MTIM3 module. 0 Bus clock to the MTIM3 module is disabled. 1 Bus clock to the MTIM3 module is enabled.
1 MTMI2	**MTMI2 Clock Gate Control** — This bit controls the clock gate to the MTIM2 module. 0 Bus clock to the MTIM2 module is disabled. 1 Bus clock to the MTIM2 module is enabled.
0 MTMI1	**MTMI1 Clock Gate Control** — This bit controls the clock gate to the MTIM1 module. 0 Bus clock to the MTIM1 module is disabled. 1 Bus clock to the MTIM1 module is enabled.

7.7.13 System Clock Gating Control 5 Register (SCGC5)

This register contains control bits to enable or disable the bus clock to the flash modules. Gating off the clocks to unused peripherals is used to reduce the microcontroller's run and wait currents. See Section 7.6, "Peripheral Clock Gating," for more information.

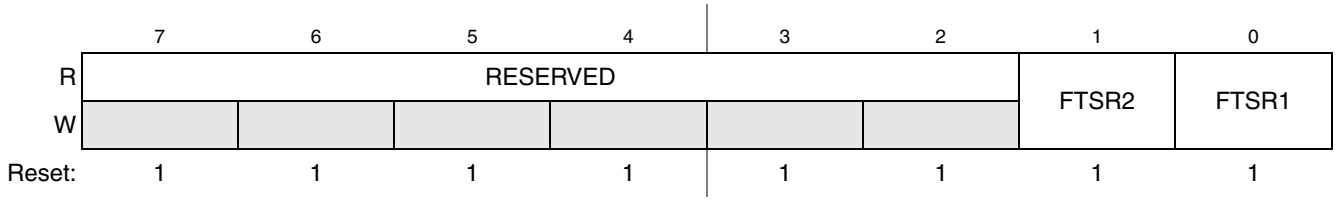


Figure 7-14. System Clock Gating Control 5 Register (SCGC5)

Table 7-19. SCGC5 Bit Field Descriptions

Field	Description
7-2 RESERVED	RESERVED for future use. Write as all 1's.
1 FTSR2	**FTSR2 Clock Gate Control** — This bit controls the bus clock gate to the flash controller #2 registers. This bit does not affect normal program execution from the flash array. Only the clock to the flash control registers is affected. 0 Bus clock to flash registers is disabled. 1 Bus clock to flash registers is enabled.
0 FTSR1	**FTSR1 Clock Gate Control** — This bit controls the bus clock gate to the flash controller #1 registers. This bit does not affect normal program execution from the flash array. Only the clock to the flash control registers is affected. 0 Bus clock to flash registers is disabled. 1 Bus clock to flash registers is enabled.

7.7.14 SIM Clock Options Register (SIMCO)

This register controls operation of the CLKOUT pin. The CS field in this register controls the output mux function for the CLKOUT pin. The various clock sources must be enabled/disabled via the appropriate controls elsewhere in the device.

	7	6	5	4	3	2	1	0
R					CS			
W								
POR:	0	0	0	0	0	0	0	0

Figure 7-15. SIM Clock Options Register (SIMCO)

Table 7-20. SIMCO Bit Field Descriptions

Field	Description
7-4	**RESERVED**
3-0 CS	**CLKOUT Select**— 0000 CLKOUT = 0 0001 CLKOUT = Crystal oscillator 1 0010 CLKOUT = Crystal oscillator 2 0011 CLKOUT = internal RC oscillator 0100 CLKOUT = BUSCLK 0101 CLKOUT = CPUCLK 0110 CLKOUT = LPOCLK 1000 CLKOUT = ADC 1 asynchronous clock 1001 CLKOUT = ADC 2 asynchronous clock 1010 CLKOUT = ADC 3 asynchronous clock 1011 CLKOUT = ADC 4 asynchronous clock 1100 CLKOUT = IRTC off chip clock

7.7.15 Internal Peripheral Select Register 1 (SIMIPS1)

This register configures connections to select peripherals. The fields in this register control clock sources for a number of on-chip timers. The various clock sources must be enabled/disabled via the appropriate controls elsewhere in the device.

	7	6	5	4	3	2	1	0
R	TPM	0	MTIM3		0	MTIM2		MTIM1
W								
POR:	0	0	0	0	0	0	0	0

Figure 7-16. SIM Internal Peripheral Select Register 1 (SIMIPS1)

Table 7-21. SIMIPS1 Bit Field Descriptions

Field	Description
7 TPM	**TPM External Clock Select Pin** — 0 = The external clock input of TPM will be fed by TPMCLK 1 = The external clock input of TPM will be fed by the output of MTIM1
6	**Reserved** — Write as zero.
5-4 MTIM3	**MTIM3 External Clock Pin Select** — 00 = The TCLK input of MTIM3 will be fed by TMRCLK1 01 = The TCLK input of MTIM3 will be fed by TMRCLK2 10 = The TCLK input of MTIM3 will be fed by the output of MTIM1 11 = RESERVED
3	**Reserved** — Write as zero.
2-1 MTIM2	**MTIM2 External Clock Pin Select** — 00 = The TCLK input of MTIM2 will be fed by TMRCLK1 01 = The TCLK input of MTIM2 will be fed by TMRCLK2 10 = The TCLK input of MTIM2 will be fed by the output of MTIM1 11 = RESERVED
0 MTIM1	**MTIM1 External Clock Pin Select** — 0 = The TCLK input of MTIM1 will be fed by TMRCLK1 1 =The TCLK input of MTIM1 will be fed by TMRCLK2

7.7.16 Internal Peripheral Select Register 2 (SIMIPS2)

The fields in this register control sources used for two of the SCI RX pins, as well as modulation choices for the SCI TX pins. The various clock sources used for modulation purposes must be enabled/disabled via the appropriate controls elsewhere in the device. See Figure 2-4 for an illustration of these controls in action.

	7	6	5	4	3	2	1	0
R W	RX2IN	RX1IN	MTBASE2		MTBASE1		MODTX2	MODTX1
POR:	0	0	0	0	0	0	0	0

Figure 7-17. SIM Internal Peripheral Select Register 1 (SIMIPS2)

Table 7-22. SIMIPS2 Register Bit Fields

Field	Description
7 RX2IN	**SCI2 RX Input Pin Select** — 0 = RX2 is fed from the digital input pin (assuming the RX2 is enabled on that pin via the MC registers) 1 = RX2 is fed from the output of comparator 2
6 RX1IN	**SCI1 RX Input Pin Select** — 0 = RX1 is fed from the digital input pin (assuming the RX1 is enabled on that pin via the MC registers) 1 = RX1 is fed from the output of comparator 1

Table 7-22. SIMIPS2 Register Bit Fields

5-4 MTBASE2	**SCI2 TX Modulation Time Base Select** — 00 = TPM CH0 01 = TPM CH1 10 = MTIM2 11 = MTIM3
3-2 MTBASE1	**SCI1 TX Modulation Time Base Select** — 00 = TPM CH0 01 = TPM CH1 10 = MTIM2 11 = MTIM3
1 MODTX2	**Modulate TX2** — 0 = Do not modulate the output of SCI2 1 = Modulate the output of SCI2 with the timebase selected via the MTBASE2 field
0 MODTX1	**Modulate TX1** — 0 = Do not modulate the output of SCI1 1 = Modulate the output of SCI1 with the timebase selected via the MTBASE1 field

Chapter 8
ColdFire Core

8.1 Introduction

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

8.1.1 Overview

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

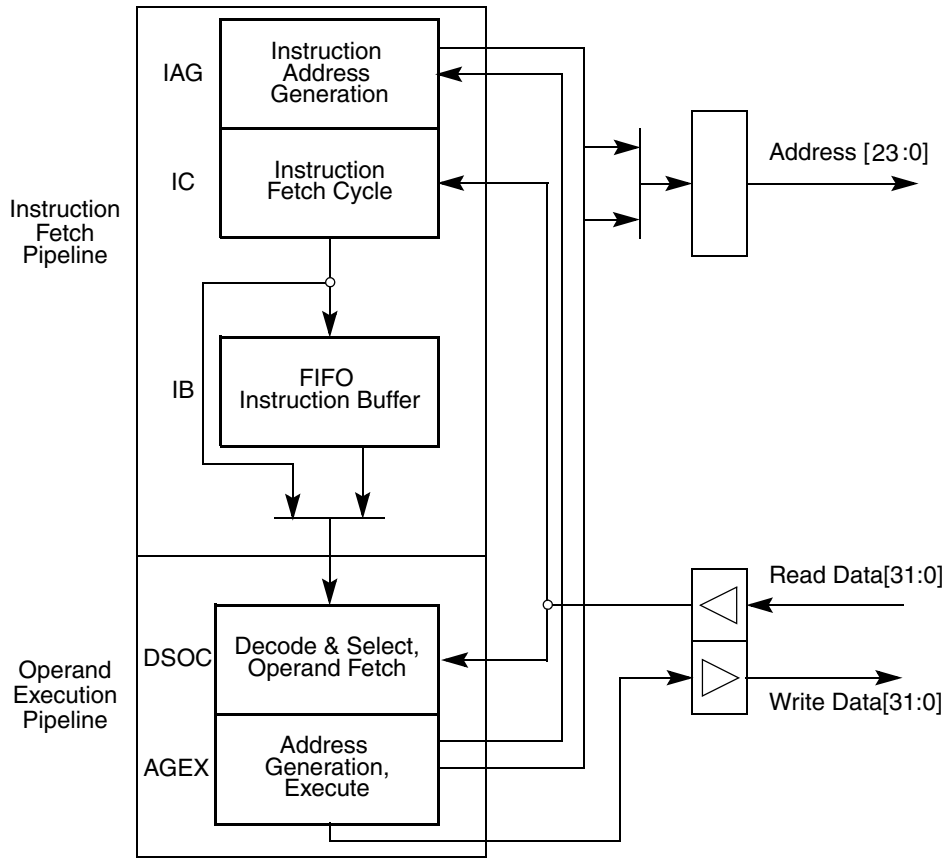


Figure 8-1. V1 ColdFire Core Pipelines

The instruction fetch pipeline (IFP) is a two-stage pipeline for prefetching instructions. The prefetched instruction stream is then gated into the two-stage operand execution pipeline (OEP), that decodes the

instruction, fetches the required operands, and then executes the required function. Because the IFP and OEP pipelines are decoupled by an instruction buffer serving as a FIFO queue, the IFP is able to prefetch instructions in advance of their actual use by the OEP thereby minimizing time stalled waiting for instructions.

The V1 ColdFire core pipeline stages include the following:

- Two-stage instruction fetch pipeline (IFP) (plus optional instruction buffer stage)
  - — Instruction address generation (IAG) — Calculates the next prefetch address
  - — Instruction fetch cycle (IC)—Initiates prefetch on the processor's local bus
  - — Instruction buffer (IB) — Optional buffer stage minimizes fetch latency effects using FIFO queue
- Two-stage operand execution pipeline (OEP)
  - — Decode and select/operand fetch cycle (DSOC)—Decodes instructions and fetches the required components for effective address calculation, or the operand fetch cycle
  - — Address generation/execute cycle (AGEX)—Calculates operand address or executes the instruction

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

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

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

8.2 Memory Map/Register Description

The following sections describe the processor registers in the user and supervisor programming models. The programming model is selected based on the processor privilege level (user mode or supervisor mode) as defined by the S bit of the status register (SR). Table 8-1 lists the processor registers.

The user-programming model consists of the following registers:

- 16 general-purpose 32-bit registers (D0–D7, A0–A7)
- 32-bit program counter (PC)
- 8-bit condition code register (CCR)
- MAC registers (described fully in Chapter 9, "Multiply-Accumulate Unit (MAC)")
  - — One 32-bit accumulator(ACC) register
  - — One 16-bit mask register (MASK)
  - — 8-bit Status register (MACSR)

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

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

Table 8-1. ColdFire Core Programming Model

BDM Command[1]	Register	Width (bits)	Access	Reset Value	Written with MOVEC[2]	Section/Page
Supervisor/User Access Registers						
Load: 0x60 Store: 0x40	Data Register 0 (D0)	32	R/W	See 8.3.3.14/8-19	No	8.2.1/8-4
Load: 0x61 Store: 0x41	Data Register 1 (D1)	32	R/W	See 8.3.3.14/8-19	No	8.2.1/8-4
Load: 0x6–7 Store: 0x4–7	Data Register –7 (D–D7)	32	R/W	POR: Undefined Else: Unaffected	No	8.2.1/8-4
Load: 0x68–E Store: 0x48–E	Address Register 0–6 (A0–A6)	32	R/W	POR: Undefined Else: Unaffected	No	8.2.2/8-4
Load: 0x6F Store: 0x4F	User A7 Stack Pointer (A7)	32	R/W	POR: Undefined Else: Unaffected	No	8.2.3/8-5
Load: 0xE4 Store: 0xC4	MAC Status Register (MACSR)	8	R/W	0x00	No	9.2.1/9-2
Load: 0xE5 Store: 0xC5	MAC Address Mask Register (MASK)	16	R/W	0xFFFF	No	9.2.2/9-4
Load: 0xE6 Store: 0xC6	MAC Accumulator (ACC)	32	R/W	POR: Undefined Else: Unaffected	No	9.2.3/9-5
Load: 0xEE Store: 0xCE	Condition Code Register (CCR)	8	R/W	POR: Undefined Else: Unaffected	No	8.2.4/8-5
Load: 0xEF Store: 0xCF	Program Counter (PC)	32	R/W	Contents of location 0x(00)00_0004	No	8.2.5/8-6
Supervisor Access Only Registers						
Load: 0xE0 Store: 0xC0	Supervisor A7 Stack Pointer (OTHER_A7)	32	R/W	Contents of location 0x(00)00_0000	No	8.2.3/8-5
Load: 0xE1 Store: 0xC1	Vector Base Register (VBR)	32	R/W	See section	Yes; Rc = 0x801	8.2.6/8-7
Load: 0xE2 Store: 0xC2	CPU Configuration Register (CPUCR)	32	W	See section	Yes; Rc = 0x802	8.2.7/8-7

Table 8-1. ColdFire Core Programming Model (continued)

BDM Command[1]	Register	Width (bits)	Access	Reset Value	Written with MOVEC[2]	Section/Page
Load: 0xEE Store: 0xCE	Status Register (SR)	16	R/W	0x27--	No	8.2.8/8-8

1 The values listed in this column represent the 8-bit BDM command code used when accessing the core registers via the 1-pin BDM port. For more information see Chapter 26, "Version 1 ColdFire Debug (CF1_DEBUG)." (These BDM commands are not similar to other ColdFire processors.)

2 If the given register is written using the MOVEC instruction, the 12-bit control register address (Rc) is also specified.

8.2.1 Data Registers (D0–D7)

D0–D7 data registers are for bit (1-bit), byte (8-bit), word (16-bit) and longword (32-bit) operations; they can also be used as index registers.

NOTE

Registers D0 and D1 contain hardware configuration details after reset. See Section 8.3.3.14, "Reset Exception" for more details.

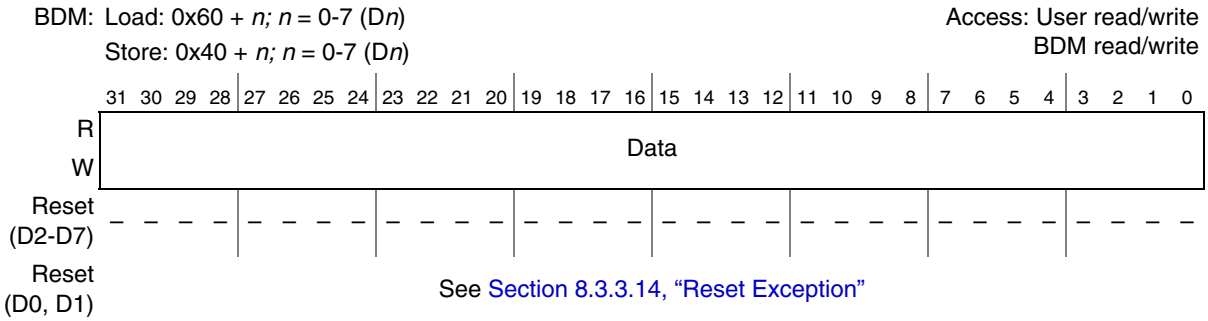


Figure 8-2. Data Registers (D0–D7)

8.2.2 Address Registers (A0–A6)

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

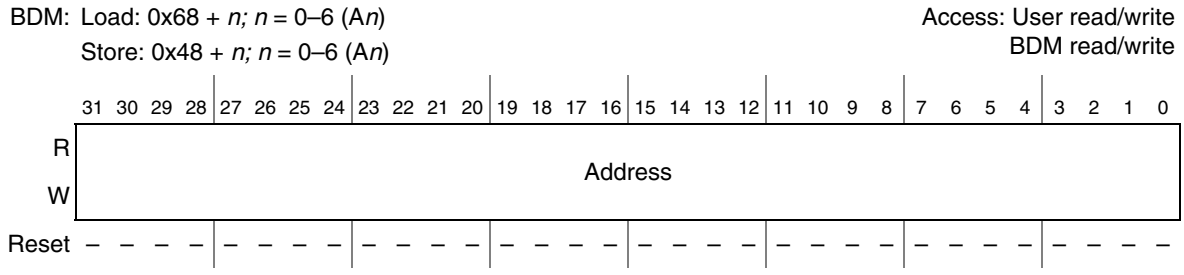


Figure 8-3. Address Registers (A0–A6)

8.2.3 Supervisor/User Stack Pointers (A7 and OTHER_A7)

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

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

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

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

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

These instructions are described in the *ColdFire Family Programmer's Reference Manual*. All other instruction references to the stack pointer, explicit or implicit, access the active A7 register.

NOTE

The USP must be initialized using the `move.l Ay,USP` instruction before any entry into user mode.

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

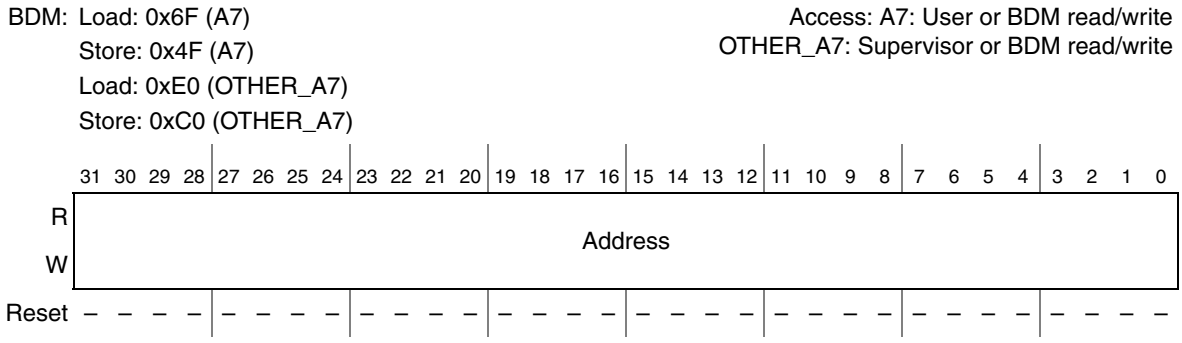


Figure 8-4. Stack Pointer Registers (A7 and OTHER_A7)

8.2.4 Condition Code Register (CCR)

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

BDM: LSB of Status Register (SR)
Load: 0xEE (SR)
Store: 0xCE (SR)

Access: User read/write
BDM read/write

	7	6	5	4	3	2	1	0
R	0	0	0	X	N	Z	V	C
W								
Reset:	0	0	0	—	—	—	—	—

Figure 8-5. Condition Code Register (CCR)

Table 8-2. CCR Field Descriptions

Field	Description
7–5	Reserved, must be cleared.
4 X	Extend condition code bit. Set to the C-bit value for arithmetic operations; otherwise not affected or set to a specified result.
3 N	Negative condition code bit. Set if most significant bit of the result is set; otherwise cleared.
2 Z	Zero condition code bit. Set if result equals zero; otherwise cleared.
1 V	Overflow condition code bit. Set if an arithmetic overflow occurs implying the result cannot be represented in operand size; otherwise cleared.
0 C	Carry condition code bit. Set if a carry out of the operand msb occurs for an addition or if a borrow occurs in a subtraction; otherwise cleared.

8.2.5 Program Counter (PC)

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

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

BDM: Load: 0xEF (PC)
Store: 0xCF (PC)

Access: User read/write
BDM read/write

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

Figure 8-6. Program Counter Register (PC)

8.2.6 Vector Base Register (VBR)

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

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

BDM: 0x801 (VBR)
Load: 0xE1 (VBR)
Store: 0xC1 (VBR)

Access: Supervisor read/write
BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	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	Base Address				–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Figure 8-7. Vector Base Register (VBR)

8.2.7 CPU Configuration Register (CPUCR)

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

BDM: 0x802 (CPUCR)
Load: 0xE2 (CPUCR)
Store: 0xC2 (CPUCR)

Access: Supervisor read/write
BDM read/write

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

Figure 8-8. CPU Configuration Register (CPUCR)

Table 8-3. CPUCR Field Descriptions

Field	Description
31 ARD	Address-related reset disable. Used to disable the generation of a reset event in response to a processor exception caused by an address error, a bus error, an RTE format error, or a fault-on-fault halt condition. 0 The detection of these types of exception conditions or the fault-on-fault halt condition generate a reset event. 1 No reset is generated in response to these exception conditions.
30 IRD	Instruction-related reset disable. Used to disable the generation of a reset event in response to a processor exception caused by the attempted execution of an illegal instruction (except for the ILLEGAL opcode), illegal line A, illegal line F instructions, or a privilege violation. 0 The detection of these types of exception conditions generate a reset event. 1 No reset is generated in response to these exception conditions.
29 IAE	Interrupt acknowledge (IACK) enable. Forces the processor to generate an IACK read cycle from the interrupt controller during exception processing to retrieve the vector number of the interrupt request being acknowledged. The processor's execution time for an interrupt exception is slightly improved when this bit is cleared. 0 The processor uses the vector number provided by the interrupt controller at the time the request is signaled. 1 IACK read cycle from the interrupt controller is generated.
28 IME	Interrupt mask enable. Forces the processor to raise the interrupt level mask (SR[I]) to 7 during every interrupt exception. 0 As part of an interrupt exception, the processor sets SR[I] to the level of the interrupt being serviced. 1 As part of an interrupt exception, the processor sets SR[I] to 7. This disables all level 1-6 interrupt requests but allows recognition of the edge-sensitive level 7 requests.
27 BWD	Buffered write disable. The ColdFire core is capable of marking processor memory writes as bufferable or non-bufferable. 0 Writes are buffered and the bus cycle is terminated immediately with zero wait states. 1 Disable the buffering of writes. In this configuration, the write transfer is terminated based on the response time of the addressed destination memory device. **Note:** If buffered writes are enabled (BWD = 0), any error status is lost as the immediate termination of the data transfer assumes an error-free completion.
26	Reserved, must be cleared.
25 FSD	Flash speculation disabled. Disables certain performance-enhancing features related to address speculation in the flash memory controller. 0 The flash controller tries to speculate on read accesses to improve processor performance by minimizing the exposed flash memory access time. Recall the basic flash access time is two processor cycles. 1 Certain flash address speculation is disabled.
24 CBRR	Crossbar round-robin arbitration enable. Configures the crossbar slave ports to fixed-priority or round-robin arbitration. 0 Fixed-priority arbitration 1 Round robin arbitration
23–0	Reserved, must be cleared.

8.2.8 Status Register (SR)

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

BDM: Load: 0xEE (SR)
Store: 0xCE (SR)

Access: Supervisor read/write
BDM read/write

	System Byte								Condition Code Register (CCR)							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	T	0	S	M	0	I			0	0	0	X	N	Z	V	C
W																
Reset	0	0	1	0	0	1	1	1	0	0	0	—	—	—	—	—

Figure 8-9. Status Register (SR)

Table 8-4. SR Field Descriptions

Field	Description
15 T	Trace enable. When set, the processor performs a trace exception after every instruction.
14	Reserved, must be cleared.
13 S	Supervisor/user state. 0 User mode 1 Supervisor mode
12 M	Master/interrupt state. Bit is cleared by an interrupt exception and software can set it during execution of the RTE or move to SR instructions.
11	Reserved, must be cleared.
10–8 I	Interrupt level mask. Defines current interrupt level. Interrupt requests are inhibited for all priority levels less than or equal to current level, except edge-sensitive level 7 requests, which cannot be masked.
7–0 CCR	Refer to Section 8.2.4, “Condition Code Register (CCR)”.

8.3 Functional Description

8.3.1 Instruction Set Architecture (ISA_C)

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

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

The original ISA definition minimized support for instructions referencing byte- and word-sized operands. Full support for the move byte and move word instructions was provided, but the only other opcodes supporting these data types are CLR (clear) and TST (test). A set of instruction enhancements has been

implemented in subsequent ISA revisions, ISA_B and ISA_C. The new opcodes primarily addressed three areas:

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

Table 8-5 summarizes the instructions added to revision ISA_A to form revision ISA_C. For more details see the *ColdFire Family Programmer's Reference Manual*.

Table 8-5. Instruction Enhancements over Revision ISA_A

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

8.3.2 Exception Processing Overview

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

- A simplified exception vector table
- Reduced relocation capabilities using the vector-base register

- A single exception stack frame format
- Use of separate system stack pointers for user and supervisor modes.

All ColdFire processors use an instruction restart exception model. Exception processing includes all actions from fault condition detection to the initiation of fetch for first handler instruction. Exception processing is comprised of four major steps:

1. The processor makes an internal copy of the SR and then enters supervisor mode by setting the S bit and disabling trace mode by clearing the T bit. The interrupt exception also forces the M bit to be cleared and the interrupt priority mask to set to current interrupt request level.

2. The processor determines the exception vector number. For all faults except interrupts, the processor performs this calculation based on exception type. For interrupts, the processor performs an interrupt-acknowledge (IACK) bus cycle to obtain the vector number from the interrupt controller if CPUCR[IAE] is set. The IACK cycle is mapped to special locations within the interrupt controller's address space with the interrupt level encoded in the address. If CPUCR[IAE] is cleared, the processor uses the vector number supplied by the interrupt controller at the time the request was signaled for improved performance.
3. The processor saves the current context by creating an exception stack frame on the system stack. The exception stack frame is created at a 0-modulo-4 address on top of the system stack pointed to by the supervisor stack pointer (SSP). As shown in Figure 8-10, the processor uses a simplified fixed-length stack frame for all exceptions. The exception type determines whether the program counter placed in the exception stack frame defines the location of the faulting instruction (fault) or the address of the next instruction to be executed (next).
4. The processor calculates the address of the first instruction of the exception handler. By definition, the exception vector table is aligned on a 1 MB boundary. This instruction address is generated by fetching an exception vector from the table located at the address defined in the vector base register. The index into the exception table is calculated as (4 × vector number). After the exception vector has been fetched, the vector contents determine the address of the first instruction of the desired handler. After the instruction fetch for the first opcode of the handler has initiated, exception processing terminates and normal instruction processing continues in the handler.

All ColdFire processors support a 1024-byte vector table aligned on any 1 MB address boundary (see Table 8-6). For the V1 ColdFire core, the only practical locations for the vector table are based at 0x(00)00_0000 in the flash or 0x(00)80_0000 in the internal SRAM.

The table contains 256 exception vectors; the first 64 are defined for the core and the remaining 192 are device-specific peripheral interrupt vectors. See Chapter 10, "Interrupt Controller (CF1_INTC)" for details on the device-specific interrupt sources.

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

Table 8-6. Exception Vector Assignments

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

[1] Fault refers to the PC of the instruction that caused the exception. Next refers to the PC of the instruction that follows the instruction that caused the fault.

All ColdFire processors inhibit interrupt sampling during the first instruction of all exception handlers. This allows any handler to disable interrupts effectively, if necessary, by raising the interrupt mask level contained in the status register. In addition, the ISA_C architecture includes an instruction (STLDSR) that stores the current interrupt mask level and loads a value into the SR. This instruction is specifically intended for use as the first instruction of an interrupt service routine that services multiple interrupt requests with different interrupt levels. Finally, the V1 ColdFire core includes the CPUCR[IME] bit that forces the processor to automatically raise the mask level to 7 during the interrupt exception, removing the need for any explicit instruction in the service routine to perform this function. For more details, see *ColdFire Family Programmer's Reference Manual*.

8.3.2.1 Exception Stack Frame Definition

Figure 8-10 shows exception stack frame. The first longword contains the 16-bit format/vector word (F/V) and the 16-bit status register, and the second longword contains the 32-bit program counter address.

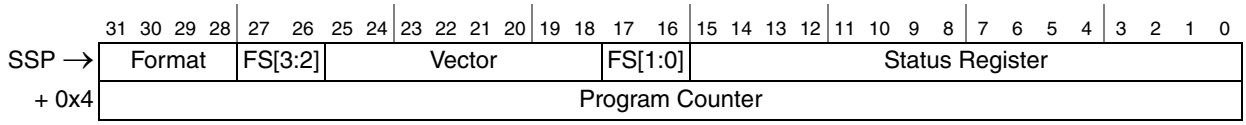


Figure 8-10. Exception Stack Frame Form

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

- A 4-bit format field at the top of the system stack is always written with a value of 4, 5, 6, or 7 by the processor, indicating a two-longword frame format. See Table 8-7.

Table 8-7. Format Field Encodings

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

- There is a 4-bit fault status field, FS[3:0], at the top of the system stack. This field is defined for access and address errors only and written as zeros for all other exceptions. See Table 8-8.

Table 8-8. Fault Status Encodings

FS[3:0]	Definition
00*xx*	Reserved
0100	Error on instruction fetch
0101	Reserved
011x	Reserved
1000	Error on operand write
1001	Reserved
101x	Reserved
1100	Error on operand read
1101	Reserved
111x	Reserved

- The 8-bit vector number, vector[7:0], defines the exception type and is calculated by the processor for all internal faults and represents the value supplied by the interrupt controller in case of an interrupt. See Table 8-6.

8.3.3 Processor Exceptions

8.3.3.1 Access Error Exception

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

The exact processor response to an access error depends on the memory reference being performed. For an instruction fetch, the processor postpones the error reporting until the faulted reference is needed by an instruction for execution. Therefore, faults during instruction prefetches followed by a change of instruction flow do not generate an exception. When the processor attempts to execute an instruction with a faulted opword and/or extension words, the access error is signaled and the instruction is aborted. For this type of exception, the programming model has not been altered by the instruction generating the access error.

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

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

8.3.3.2 Address Error Exception

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

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

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

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

8.3.3.3 Illegal Instruction Exception

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

The ColdFire variable-length instruction set architecture supports three instruction sizes: 16, 32, or 48 bits. The first instruction word is known as the operation word (or opword), while the optional words are known as extension word 1 and extension word 2. The opword is further subdivided into three sections: the upper four bits segment the entire ISA into 16 instruction lines, the next 6 bits define the operation mode (opmode), and the low-order 6 bits define the effective address. See Figure 8-11. The opword line definition is shown in Table 8-9.

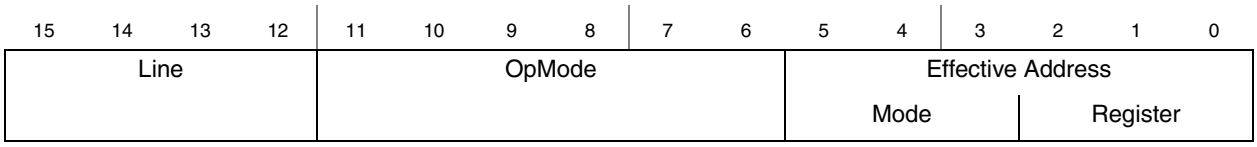


Figure 8-11. ColdFire Instruction Operation Word (Opword) Format

Table 8-9. ColdFire Opword Line Definition

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

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

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

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

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

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

8.3.3.4 Privilege Violation

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

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

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

8.3.3.5 Trace Exception

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

The stop instruction has the following effects:

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

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

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

8.3.3.6 Unimplemented Line-A Opcode

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

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

8.3.3.7 Unimplemented Line-F Opcode

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

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

8.3.3.8 Debug Interrupt

See Chapter 26, “Version 1 ColdFire Debug (CF1_DEBUG),” for a detailed explanation of this exception, which is generated in response to a hardware breakpoint register trigger. The processor does not generate an IACK cycle, but rather calculates the vector number internally (vector number 12). Additionally, SR[M,I] are unaffected by the interrupt.

8.3.3.9 RTE and Format Error Exception

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

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

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

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

8.3.3.10 TRAP Instruction Exception

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

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

8.3.3.11 Unsupported Instruction Exception

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

All ColdFire cores record the processor hardware configuration in the D0 register immediately after the negation of $\overline{\text{RESET}}$. See Section 8.3.3.14, "Reset Exception," for details.

For this device, attempted execution of valid integer divide opcodes and CAU instructions result in the unsupported instruction exception.

8.3.3.12 Interrupt Exception

Interrupt exception processing includes interrupt recognition and the fetch of the appropriate vector from the interrupt controller using an IACK cycle or using the previously-supplied vector number, under control of CPUCR[IAE]. See Chapter 10, "Interrupt Controller (CF1_INTC)," for details on the interrupt controller.

8.3.3.13 Fault-on-Fault Halt

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

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

8.3.3.14 Reset Exception

Asserting the reset input signal ($\overline{\text{RESET}}$) to the processor causes a reset exception. The reset exception has the highest priority of any exception; it provides for system initialization and recovery from catastrophic failure. Reset also aborts any processing in progress when the reset input is recognized. Processing cannot be recovered.

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

NOTE

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

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

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

Information loaded into D0 defines the processor hardware configuration as shown in Figure 8-12.

BDM: Load: 0x60 (D0)
Store: 0x40 (D0)

Access: User read-only
BDM read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PF								VER				REV			
W																
Reset	1	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0

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

Figure 8-12. D0 Hardware Configuration Info

Table 8-10. D0 Hardware Configuration Info Field Description

Field	Description
31–24 PF	Processor family. This field is fixed to a hex value of 0xCF indicating a ColdFire core is present.
23–20 VER	ColdFire core version number. Defines the hardware microarchitecture version of ColdFire core. 0001 V1 ColdFire core (This is the value used for this device.) 0010 V2 ColdFire core 0011 V3 ColdFire core 0100 V4 ColdFire core 0101 V5 ColdFire core Else Reserved for future use
19–16 REV	Processor revision number. The default is 0b0000.
15 MAC	MAC present. This bit signals if the optional multiply-accumulate (MAC) execution engine is present in processor core. 0 MAC execute engine not present in core. 1 MAC execute engine is present in core. (This is the value used for this device.)
14 DIV	Divide present. This bit signals if the hardware divider (DIV) is present in the processor core. 0 Divide execute engine not present in core. (This is the value used for this device.) 1 Divide execute engine is present in core.
13–8	Reserved.
7–4 ISA	ISA revision. Defines the instruction-set architecture (ISA) revision level implemented in ColdFire processor core. 0000 ISA_A 0001 ISA_B 0010 ISA_C (This is the value used for this device.) 1000 ISA_A+ Else Reserved
3–0 DEBUG	Debug module revision number. Defines revision level of the debug module used in the ColdFire processor core. 0000 DEBUG_A 0001 DEBUG_B 0010 DEBUG_C 0011 DEBUG_D 0100 DEBUG_E 1001 DEBUG_B+ (This is the value used for this device.) 1011 DEBUG_D+ 1111 DEBUG_D+PST Buffer Else Reserved

Information loaded into D1 defines the local memory hardware configuration as shown in the figure below.

BDM: Load: 0x61 (D1)
Store: 0x41 (D1)

Access: User read-only
BDM read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	1	0	0	0	0	FLASHSZ[1]					0	0	0
W																
Reset	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	1	ROMSZ				SRAMSZ[2]					0	0	0
W																
Reset	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0

[1] The FLASHSZ size depends on memory size. The size shown is for 256 KB flash.

[2] The SRAMSZ size depends on memory size. The size shown is for 16 KB SRAM.

Figure 8-13. D1 Hardware Configuration Info

Table 8-11. D1 Hardware Configuration Information Field Description

Field	Description
31–24	Reserved.
23–19 FLASHSZ	Flash bank size. 00000-01110 No flash 10000 64 KB flash 10010 128 KB flash 10011 96 KB flash 10100 256 KB flash (This is the value used for this device) 10110 512 KB flash Else Reserved for future use **Note:** The FLASHSZ size depends on memory size. The size shown is for 256 KB flash.
18–16	Reserved
15–12	Reserved, resets to 0b0001
11–8 ROMSZ	Boot ROM size. Indicates the size of the boot ROM. 0000 No boot ROM (This is the value used for this device) 0001 512 bytes 0010 1 KB 0011 2 KB 0100 4 KB 0101 8 KB 0110 16 KB 0111 32 KB Else Reserved for future use

Table 8-11. D1 Hardware Configuration Information Field Description (continued)

Field	Description
7–3 SRAMSZ	SRAM bank size. 00000 No SRAM 00010 512 bytes 00100 1 KB 00110 2 KB 01000 4 KB 01010 8 KB 01100 16 KB (This is the value used for this device) 01111 24 KB 01110 32 KB 10000 64 KB 10010 128 KB Else Reserved for future use
2–0	Reserved.

8.3.4 Instruction Execution Timing

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

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

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

8.3.4.1 Timing Assumptions

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

1. The OEP is loaded with the opword and all required extension words at the beginning of each instruction execution. This implies that the OEP does not wait for the IFP to supply opwords and/or extension words.
2. The OEP does not experience any sequence-related pipeline stalls. The most common example of stall involves consecutive store operations, excluding the MOVEM instruction. For all STORE operations (except MOVEM), certain hardware resources within the processor are marked as busy for two clock cycles after the final decode and select/operand fetch cycle (DSOC) of the store instruction. If a subsequent STORE instruction is encountered within this 2-cycle window, it is stalled until the resource again becomes available. Thus, the maximum pipeline stall involving consecutive STORE operations is two cycles. The MOVEM instruction uses a different set of resources and this stall does not apply.
3. The OEP completes all memory accesses without any stall conditions caused by the memory itself. Thus, the timing details provided in this section assume that an infinite zero-wait state memory is attached to the processor core.

4. All operand data accesses are aligned on the same byte boundary as the operand size; for example, 16-bit operands aligned on 0-modulo-2 addresses, 32-bit operands aligned on 0-modulo-4 addresses.

   The processor core decomposes misaligned operand references into a series of aligned accesses as shown in Table 8-12.

Table 8-12. Misaligned Operand References

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

8.3.4.2 MOVE Instruction Execution Times

Table 8-13 lists execution times for MOVE.{B,W} instructions; Table 8-14 lists timings for MOVE.L.

NOTE

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

ET with {<ea> = (d16,PC)}	equals ET with {<ea> = (d16,An)}
ET with {<ea> = (d8,PC,Xi*SF)}	equals ET with {<ea> = (d8,An,Xi*SF)}

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

Table 8-13. MOVE Byte and Word Execution Times

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

Table 8-13. MOVE Byte and Word Execution Times (continued)

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

Table 8-14. MOVE Long Execution Times

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

8.3.4.3 Standard One Operand Instruction Execution Times

Table 8-15. One Operand Instruction Execution Times

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xn*SF)	xxx.wl	#xxx
BITREV	Dx	1(0/0)	—	—	—	—	—	—	—
BYTEREV	Dx	1(0/0)	—	—	—	—	—	—	—
CLR.B	<ea>	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
CLR.W	<ea>	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
CLR.L	<ea>	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
EXT.W	Dx	1(0/0)	—	—	—	—	—	—	—
EXT.L	Dx	1(0/0)	—	—	—	—	—	—	—
EXTB.L	Dx	1(0/0)	—	—	—	—	—	—	—

Table 8-15. One Operand Instruction Execution Times (continued)

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

8.3.4.4 Standard Two Operand Instruction Execution Times

Table 8-16. Two Operand Instruction Execution Times

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

Table 8-16. Two Operand Instruction Execution Times (continued)

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An) (d16,PC)	(d8,An,Xn*SF) (d8,PC,Xn*SF)	xxx.wl	#xxx
BTST	Dy,<ea>	2(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	—
BTST	#imm,<ea>	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	—	—	—
CMP.B	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
CMP.W	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
CMP.L	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
CMPI.B	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
CMPI.W	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
CMPI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
EOR.L	Dy,<ea>	1(0/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
EORI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
LEA	<ea>,Ax	—	1(0/0)	—	—	1(0/0)	2(0/0)	1(0/0)	—
LSL.L	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
LSR.L	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
MOVEQ.L	#imm,Dx	—	—	—	—	—	—	—	1(0/0)
OR.L	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
OR.L	Dy,<ea>	—	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
ORI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
SUB.L	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
SUB.L	Dy,<ea>	—	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
SUBI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
SUBQ.L	#imm,<ea>	1(0/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
SUBX.L	Dy,Dx	1(0/0)	—	—	—	—	—	—	—

8.3.4.5 Miscellaneous Instruction Execution Times

Table 8-17. Miscellaneous Instruction Execution Times

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xn*SF)	xxx.wl	#xxx
LINK.W	Ay,#imm	2(0/1)	—	—	—	—	—	—	—
MOV3Q.L	#imm,<ea>	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
MOVE.L	Ay,USP	3(0/0)	—	—	—	—	—	—	—
MOVE.L	USP,Ax	3(0/0)	—	—	—	—	—	—	—
MOVE.W	CCR,Dx	1(0/0)	—	—	—	—	—	—	—
MOVE.W	<ea>,CCR	1(0/0)	—	—	—	—	—	—	1(0/0)
MOVE.W	SR,Dx	1(0/0)	—	—	—	—	—	—	—

Table 8-17. Miscellaneous Instruction Execution Times (continued)

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xn*SF)	xxx.wl	#xxx
MOVE.W	<ea>,SR	7(0/0)	—	—	—	—	—	—	7(0/0) [2]
MOVEC	Ry,Rc	9(0/1)	—	—	—	—	—	—	—
MOVEM.L	<ea>,and list	—	1+n(n/0)	—	—	1+n(n/0)	—	—	—
MOVEM.L	and list,<ea>	—	1+n(0/n)	—	—	1+n(0/n)	—	—	—
MVS	<ea>,Dx	1(0/0)	2(1/0)	2(1/0)	2(1/0)	2(1/0)	3(1/0)	2(1/0)	1(0/0)
MVZ	<ea>,Dx	1(0/0)	2(1/0)	2(1/0)	2(1/0)	2(1/0)	3(1/0)	2(1/0)	1(0/0)
NOP		3(0/0)	—	—	—	—	—	—	—
PEA	<ea>	—	2(0/1)	—	—	2(0/1) [4]	3(0/1) [5]	2(0/1)	—
PULSE		1(0/0)	—	—	—	—	—	—	—
STLDSR	#imm	—	—	—	—	—	—	—	5(0/1)
STOP	#imm	—	—	—	—	—	—	—	3(0/0) [3]
TRAP	#imm	—	—	—	—	—	—	—	12(1/2)
TPF		1(0/0)	—	—	—	—	—	—	—
TPF.W		1(0/0)	—	—	—	—	—	—	—
TPF.L		1(0/0)	—	—	—	—	—	—	—
UNLK	Ax	2(1/0)	—	—	—	—	—	—	—
WDDATA	<ea>	—	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	—
WDEBUG	<ea>	—	5(2/0)	—	—	5(2/0)	—	—	—

[1]The n is the number of registers moved by the MOVEM opcode.

[2]If a MOVE.W #imm,SR instruction is executed and imm[13] equals 1, the execution time is 1(0/0).

[3]The execution time for STOP is the time required until the processor begins sampling continuously for interrupts.

[4]PEA execution times are the same for (d16,PC).

[5]PEA execution times are the same for (d8,PC,Xn*SF).

8.3.4.6 MAC Instruction Execution Times

Table 8-18. MAC Instruction Execution Times

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An, Xn*SF)	xxx.wl	#xxx
MAC.L	Ry, Rx	3(0/0)	—	—	—	—	—	—	—
MAC.L	Ry, Rx, <ea>, Rw	—	(1/0)	(1/0)	(1/0)	(1/0)[1]	—	—	—
MAC.W	Ry, Rx	1(0/0)	—	—	—	—	—	—	—
MAC.W	Ry, Rx, <ea>, Rw	—	(1/0)	(1/0)	(1/0)	(1/0)[1]	—	—	—

Table 8-18. MAC Instruction Execution Times (continued)

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

[1] Effective address of (d16,PC) not supported

[2] Storing the accumulator requires one additional processor clock cycle when rounding is performed

8.3.4.7 Branch Instruction Execution Times

Table 8-19. General Branch Instruction Execution Times

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An) (d16,PC)	(d8,An,Xi*SF) (d8,PC,Xi*SF)	xxx.wl	#xxx
BRA		—	—	—	—	2(0/1)	—	—	—
BSR		—	—	—	—	3(0/1)	—	—	—
JMP	<ea>	—	3(0/0)	—	—	3(0/0)	4(0/0)	3(0/0)	—
JSR	<ea>	—	3(0/1)	—	—	3(0/1)	4(0/1)	3(0/1)	—
RTE		—	—	7(2/0)	—	—	—	—	—
RTS		—	—	5(1/0)	—	—	—	—	—

Table 8-20. Bcc Instruction Execution Times

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

Chapter 9
Multiply-Accumulate Unit (MAC)

9.1 Introduction

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

9.1.1 Overview

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

The MAC provides functionality in three related areas:

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

The MAC features a three-stage execution pipeline optimized for 16-bit operands, with a 16x16 multiply array and a single 32-bit accumulator.

The three areas of functionality are addressed in detail in following sections. The logic required to support this functionality is contained in a MAC module (Figure 9-1).

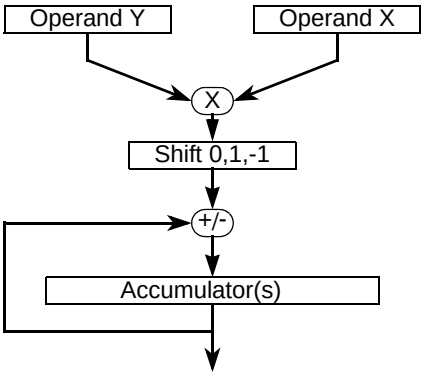


Figure 9-1. Multiply-Accumulate Functionality Diagram

9.1.1.1 Introduction to the MAC

The MAC is an extension of the basic multiplier in most microprocessors. It is typically implemented in hardware within an architecture and supports rapid execution of signal processing algorithms in fewer

cycles than comparable non-MAC architectures. For example, small digital filters can tolerate some variance in an algorithm's execution time, but larger, more complicated algorithms such as orthogonal transforms may have more demanding speed requirements beyond scope of any processor architecture and may require full DSP implementation.

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

Consider a typical filtering operation where the filter is defined as in Equation 9-1.

$$y(i) = \sum_{k=1}^{N-1} a(k)y(i-k) + \sum_{k=0}^{N-1} b(k)x(i-k) \qquad \textit{Eqn. 9-1}$$

Here, the output y(i) is determined by past output values and past input values. This is the general form of an infinite impulse response (IIR) filter. A finite impulse response (FIR) filter can be obtained by setting coefficients a(k) to zero. In either case, the operations involved in computing such a filter are multiplies and product summing. To show this point, reduce Equation 9-1 to a simple, four-tap FIR filter, shown in Equation 9-2, in which the accumulated sum is a past data values and coefficients sum.

$$y(i) = \sum_{k=0}^{3} b(k)x(i-k) = b(0)x(i) + b(1)x(i-1) + b(2)x(i-2) + b(3)x(i-3 \qquad \textit{Eqn. 9-2}$$

9.2 Memory Map/Register Definition

The following table and sections explain the MAC registers:

Table 9-1. MAC Memory Map

BDM[1]	Register	Width (bits)	Access	Reset Value	Section/Page
Read: 0xE4 Write: 0xC4	MAC Status Register (MACSR)	8	R/W	0x00	9.2.1/9-2
Read: 0xE5 Write: 0xC5	MAC Address Mask Register (MASK)	16	R/W	0xFFFF	9.2.2/9-4
Read: 0xE6 Write: 0xC6	Accumulator (ACC)	32	R/W	Undefined	9.2.3/9-5

[1] For more information see Chapter 26, "Version 1 ColdFire Debug (CF1_DEBUG)."

9.2.1 MAC Status Register (MACSR)

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

BDM: Read: 0xE4 (MACSR)
Write: 0xC4

Access: Supervisor read/write
BDM read/write

	7	6	5	4	3	2	1	0
R W	OMC	S/U	F/I	R/T	N	Z	V	C
Reset:	0	0	0	0	0	0	0	0

Figure 9-2. MAC Status Register (MACSR)

Table 9-2. MACSR Field Descriptions

Field	Description
7 OMC	Overflow saturation mode. Enables or disables saturation mode on overflow. If set, the accumulator is set to the appropriate constant (see S/U field description) on any operation that overflows the accumulator. After saturation, the accumulator remains unaffected by any other MAC or MSAC instructions until the overflow bit is cleared or the accumulator is directly loaded.
6 S/U	Signed/unsigned operations. **In integer mode:** S/U determines whether operations performed are signed or unsigned. It also determines the accumulator value during saturation, if enabled. 0 Signed numbers. On overflow, if OMC is enabled, the accumulator saturates to the most positive (0x7FFF_FFFF) or the most negative (0x8000_0000) number, depending on the instruction and the product value that overflowed. 1 Unsigned numbers. On overflow, if OMC is enabled, the accumulator saturates to the smallest value (0x0000_0000) or the largest value (0xFFFF_FFFF), depending on the instruction. **In fractional mode:** S/U controls rounding while storing the accumulator to a general-purpose register. 0 Move accumulator without rounding to a 16-bit value. Accumulator is moved to a general-purpose register as a 32-bit value. 1 The accumulator is rounded to a 16-bit value using the round-to-nearest (even) method when moved to a general-purpose register. See Section 9.3.1.1, “Rounding”. The resulting 16-bit value is stored in the lower word of the destination register. The upper word is zero-filled. This rounding procedure does not affect the accumulator value.
5 F/I	Fractional/integer mode. Determines whether input operands are treated as fractions or integers. 0 Integers can be represented in signed or unsigned notation, depending on the value of S/U. 1 Fractions are represented in signed, fixed-point, two’s complement notation. Values range from -1 to $1 - 2^{-15}$ for 16-bit fractions and -1 to $1 - 2^{-31}$ for 32-bit fractions. See Section 9.3.4, “Data Representation.”
4 R/T	Round/truncate mode. Controls rounding procedure for MSAC.L instructions when in fractional mode. 0 Truncate. The product’s lsbs are dropped before it is combined with the accumulator. 1 Round-to-nearest (even). The 64-bit product of two 32-bit, fractional operands is rounded to the nearest 32-bit value. If the low-order 32 bits equal 0x8000_0000, the upper 32 bits are rounded to the nearest even (lsb = 0) value. See Section 9.3.1.1, “Rounding”.
3 N	Negative. Set if the msb of the result is set, otherwise cleared. N is affected only by MAC, MSAC, and load operations; it is not affected by MULS and MULU instructions.
2 Z	Zero. Set if the result equals zero, otherwise cleared. This bit is affected only by MAC, MSAC, and load operations; it is not affected by MULS and MULU instructions.

Table 9-2. MACSR Field Descriptions (continued)

Field	Description
1 V	Overflow. Set if an arithmetic overflow occurs, implying that the result cannot be represented in the operand size. After set, V remains set until the accumulator register is loaded with a new value or MACSR is directly loaded. MULS and MULU instructions do not change this value.
0	Carry. This field is always zero.

Table 9-3 summarizes the interaction of the MACSR[S/U,F/I,R/T] control bits.

Table 9-3. Summary of S/U, F/I, and R/T Control Bits

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

9.2.2 Mask Register (MASK)

The MASK register performs a simple AND with the operand address for MAC instructions. The processor calculates the normal operand address and, if enabled, that address is then ANDed with {0xFFFF, MASK[15:0]} to form the final address. Therefore, with certain MASK bits cleared, the operand address can be constrained to a certain memory region. This is used primarily to implement circular queues with the (An)+ addressing mode.

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

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

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

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

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

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

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

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

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

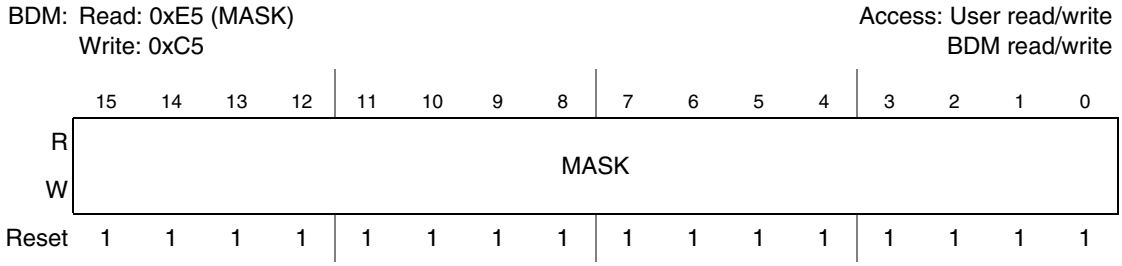


Figure 9-3. Mask Register (MASK)

Table 9-4. MASK Field Descriptions

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

9.2.3 Accumulator Register (ACC)

The accumulator register store 32-bits of the MAC operation result.

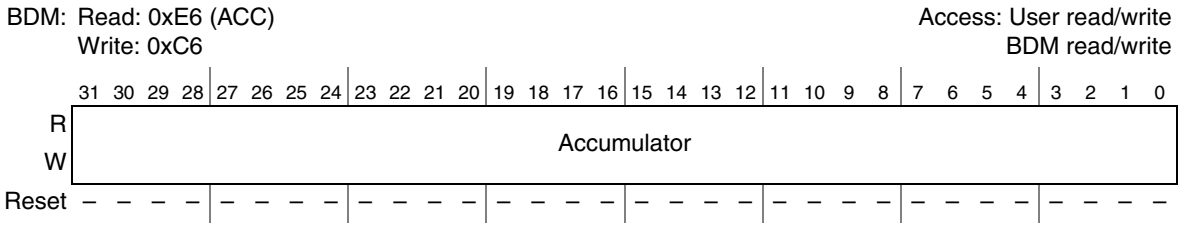


Figure 9-4. Accumulator Register (ACC)

Table 9-5. ACC Field Descriptions

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

9.3 Functional Description

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

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

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

The MAC is optimized for 16-bit multiplications to keep the area consumption low. Two 16-bit operands produce a 32-bit product. Longword operations are performed by reusing the 16-bit multiplier array at the expense of a small amount of extra control logic. Again, the product of two 32-bit operands is a 32-bit result. For longword integer operations, only the least significant 32 bits of the product are calculated. For fractional operations, the entire 64-bit product is calculated and then truncated or rounded to a 32-bit result using the round-to-nearest (even) method.

Because the multiplier array is implemented in a three-stage pipeline, MAC instructions have an effective issue rate of 1 cycle for word operations, 3 cycles for longword integer operations, and 4 cycles for 32-bit fractional operations.

All arithmetic operations use register-based input operands, and summed values are stored in the accumulator. Therefore, an additional MOVE instruction is needed to store data in a general-purpose register.

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

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

9.3.1 Fractional Operation Mode

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

9.3.1.1 Rounding

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

1. The 32-bit accumulator is moved into a general purpose register. If MACSR[S/U] is cleared, the accumulator is stored as is in the destination register; if it is set, the 32-bit value is rounded to a 16-bit value using the round-to-nearest (even) method. The resulting 16-bit number is stored in the lower word of the destination register. The upper word is zero-filled. The accumulator value is unaffected by this rounding procedure.
2. Execution of a MAC (or MSAC) instruction with 32-bit operands. If MACSR[R/T] is zero, multiplying two 32-bit numbers creates a 64-bit product truncated to the upper 32 bits; otherwise, it is rounded using round-to-nearest (even) method.

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

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

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

The rounding algorithm is summarized in the following pseudocode:

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

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

9.3.1.2 Saving and Restoring the MAC Programming Model

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

```
struct   macState {
        int acc;
        int mask;
```

```
        int macsr;
} macState;
```

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

```
MAC_state_save:
        move.l  macsr,d7          ; save the macsr
        clr.l   d0                ; zero the register to ...
        move.l  d0,macsr          ; disable rounding in the macsr
        move.l  acc,d5            ; save the accumulator
        move.l  mask,d6           ; save the address mask
        movem.l #0x00e0,(a7)      ; move the state to memory
```

This code performs the MAC state restore:

```
MAC_state_restore:
        movem.l (a7),#0x00e0; restore the state from memory
        move.l  #0,macsr          ; disable rounding in the macsr
        move.l  d5,acc            ; restore the accumulator
        move.l  d6,mask           ; restore the address mask
        move.l  d7,macsr          ; restore the macsr
```

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

9.3.1.3 MULS/MULU

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

9.3.1.4 Scale Factor in MAC or MSAC Instructions

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

9.3.2 MAC Instruction Set Summary

Table 9-6 summarizes MAC unit instructions.

Table 9-6. MAC Instruction Summary

Command	Mnemonic	Description
Multiply Signed	`muls <ea>y,Dx`	Multiplies two signed operands yielding a signed result
Multiply Unsigned	`mulu <ea>y,Dx`	Multiplies two unsigned operands yielding an unsigned result
Multiply Accumulate	`mac Ry,RxSF` `msac Ry,RxSF`	Multiplies two operands, then adds/subtracts the product to/from the accumulator
Multiply Accumulate with Load	`mac Ry,RxSF,Rw` `msac Ry,RxSF,Rw`	Multiplies two operands, combines the product to the accumulator while loading a register with the memory operand
Load Accumulator	`move.l {Ry,#imm},ACC`	Loads the accumulator with a 32-bit operand
Store Accumulator	`move.l ACC,Rx`	Writes the contents of the accumulator to a CPU register

Table 9-6. MAC Instruction Summary (continued)

Command	Mnemonic	Description
Load MACSR	`move.l {Ry,#imm},MACSR`	Writes a value to MACSR
Store MACSR	`move.l MACSR,Rx`	Write the contents of MACSR to a CPU register
Store MACSR to CCR	`move.l MACSR,CCR`	Write the contents of MACSR to the CCR
Load MAC Mask Reg	`move.l {Ry,#imm},MASK`	Writes a value to the MASK register
Store MAC Mask Reg	`move.l MASK,Rx`	Writes the contents of the MASK to a CPU register

9.3.3 MAC Instruction Execution Times

The instruction execution times for the MAC can be found in Section 8.3.4.6, "MAC Instruction Execution Times".

9.3.4 Data Representation

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

1. Two's complement signed integer: In this format, an N-bit operand value lies in the range $-2^{(N-1)} \leq$ operand $\leq 2^{(N-1)} - 1$. The binary point is right of the lsb.
2. Unsigned integer: In this format, an N-bit operand value lies in the range $0 \leq$ operand $\leq 2^N - 1$. The binary point is right of the lsb.
3. Two's complement, signed fractional: In an N-bit number, the first bit is the sign bit. The remaining bits signify the first N-1 bits after the binary point. Given an N-bit number, $a_{N-1}a_{N-2}a_{N-3}... a_2a_1a_0$, its value is given by the equation in Equation 9-3.

$$value = -(1 \cdot a_{N-1}) + \sum_{i=0}^{N-2} 2^{-(i+1-N)} \cdot ai \qquad \textit{Eqn. 9-3}$$

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

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

9.3.5 MAC Opcodes

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

Remember the following:

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

treated as a sticky flag, meaning after set, it remains set until the accumulator or the MACSR is directly loaded. See Section 9.2.1, “MAC Status Register (MACSR)”.

- The optional 1-bit shift of the product is specified using the notation {<< | >>} SF, where <<1 indicates a left shift and >>1 indicates a right shift. The shift is performed before the product is added to or subtracted from the accumulator. Without this operator, the product is not shifted. If the MAC is in fractional mode (MACSR[F/I] is set), SF is ignored and no shift is performed. Because a product can overflow, the following guidelines are implemented:
  - For unsigned word and longword operations, a zero is shifted into the product on right shifts.
  - For signed, word operations, the sign bit is shifted into the product on right shifts unless the product is zero. For signed, longword operations, the sign bit is shifted into the product unless an overflow occurs or the product is zero, in which case a zero is shifted in.
  - For all left shifts, a zero is inserted into the lsb position.

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

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

               /* perform the multiply */
               product[63:0] = operandY[31:0] * operandX[31:0]

               /* check for product overflow */
               if ((product[63:31] != 0x0000_0000_0) && (product[63:31] != 0xffff_ffff_1))
                  then {        /* product overflow */
                        MACSR.V = 1
                        if (inst == MSAC && MACSR.OMC == 1)
                           then if (product[63] == 1)
                                   then result[31:0] = 0x7fff_ffff
                                   else result[31:0] = 0x8000_0000
                           else if (MACSR.OMC == 1)
                                   then /* overflowed MAC,
                                           saturationMode enabled */
                                        if (product[63] == 1)
                                           then result[31:0] = 0x8000_0000
```

```
                                else result[31:0] = 0x7fff_ffff
            }

        /* scale product before combining with accumulator */
        switch (SF)     /* 2-bit scale factor */
        {
            case 0:     /* no scaling specified */
               break;
            case 1:     /* SF = "<< 1" */
               if (product[31] ^ product[30])
                  then {MACSR.V = 1
                        if (inst == MSAC && MACSR.OMC == 1)
                           then if (product[63] == 1)
                                   then result[31:0] = 0x7fff_ffff
                                   else result[31:0] = 0x8000_0000
                           else if (MACSR.OMC == 1)
                                then /* overflowed MAC,
                                        saturationMode enabled */
                                     if (product[63] == 1)
                                        then result[31:0] = 0x8000_0000
                                        else result[31:0] = 0x7fff_ffff
                        }
                        else product[31:0] = {product[30:0], 0}
               break;
            case 2:     /* reserved encoding */
               break;
            case 3:     /* SF = ">> 1" */
               if (MACSR.OMC == 0 || MACSR.V = 0)
                  then product[31:0] = {product[31], product[31:1]}
               break;
        }

        /* combine with accumulator */
        if (MACSR.V == 0)
           then {if (inst == MSAC)
                    then result[31:0] = acc[31:0] - product[31:0]
                    else result[31:0] = acc[31:0] + product[31:0]
           }

        /* check for accumulation overflow */
        if (accumulationOverflow == 1)
           then {MACSR.V = 1
                 if (MACSR.OMC == 1)
                    then /* accumulation overflow,
                            saturationMode enabled */
                         if (result[31] == 1)
                            then result[31:0] = 0x7fff_ffff
                            else result[31:0] = 0x8000_0000
           }

        /* transfer the result to the accumulator */
        acc[31:0] = result[31:0]
        MACSR.N = result[31]
        if (result[31:0] == 0x0000_0000)
           then MACSR.Z = 1
           else MACSR.Z = 0
}
```

```
   break;
   case 1:
   case 3:              /* signed fractionals */
      if (MACSR.OMC == 0 || MACSR.V == 0)
         then {
               MACSR.V = 0
               if (sz == word)
                  then {if (U/Ly == 1)
                           then operandY[31:0] = {Ry[31:16], 0x0000}
                           else operandY[31:0] = {Ry[15:0],  0x0000}
                        if (U/Lx == 1)
                           then operandX[31:0] = {Rx[31:16], 0x0000}
                           else operandX[31:0] = {Rx[15:0],  0x0000}
                  }
                  else {operandY[31:0] = Ry[31:0]
                        operandX[31:0] = Rx[31:0]
                  }

               /* perform the multiply */
               product[63:0] = (operandY[31:0] * operandX[31:0]) << 1

               /* check for product rounding */
               if (MACSR.R/T == 1)
                  then { /* perform convergent rounding */
                        if (product[31:0] > 0x8000_0000)
                           then product[63:32] = product[63:32] + 1
                           else if ((product[31:0] == 0x8000_0000) && (product[32] == 1))
                                   then product[63:32] = product[63:32] + 1
                  }

               /* combine with accumulator */
               if (inst == MSAC)
                  then result[31:0] = acc[31:0] - product[63:32]
                  else result[31:0] = acc[31:0] + product[63:32]

               /* check for accumulation overflow */
               if (accumulationOverflow == 1)
                  then {MACSR.V = 1
                        if (MACSR.OMC == 1)
                           then /* accumulation overflow,
                                   saturationMode enabled */
                                if (result[31] == 1)
                                   then result[31:0] = 0x7fff_ffff
                                   else result[31:0] = 0x8000_0000
                  }

               /* transfer the result to the accumulator */
               acc[31:0] = result[31:0]
               MACSR.N = result[31]
               if (result[31:0] == 0x0000_0000)
                  then MACSR.Z = 1
                  else MACSR.Z = 0
         }
   break;


   case 2:              /* unsigned integers */
```

```
if (MACSR.OMC == 0 || MACSR.V == 0)
   then {
         MACSR.V = 0
         /* select the input operands */
         if (sz == word)
            then {if (U/Ly == 1)
                     then operandY[31:0] = {0x0000, Ry[31:16]}
                     else operandY[31:0] = {0x0000, Ry[15:0]}
                  if (U/Lx == 1)
                     then operandX[31:0] = {0x0000, Rx[31:16]}
                     else operandX[31:0] = {0x0000, Rx[15:0]}
            }
            else {operandY[31:0] = Ry[31:0]
                  operandX[31:0] = Rx[31:0]
            }

         /* perform the multiply */
         product[63:0] = operandY[31:0] * operandX[31:0]

         /* check for product overflow */
         if (product[63:32] != 0x0000_0000)
            then {        /* product overflow */
                  MACSR.V = 1
                  if (inst == MSAC && MACSR.OMC == 1)
                     then result[31:0] = 0x0000_0000
                     else if (MACSR.OMC == 1)
                             then /* overflowed MAC,
                                     saturationMode enabled */
                                  result[31:0] = 0xffff_ffff
            }

         /* scale product before combining with accumulator */
         switch (SF)     /* 2-bit scale factor */
         {
             case 0:     /* no scaling specified */
                break;
             case 1:     /* SF = “<< 1” */
                if (product[31] == 1)
                   then {MACSR.V = 1
                         if (inst == MSAC && MACSR.OMC == 1)
                            then result[31:0] = 0x0000_0000
                            else if (MACSR.OMC == 1)
                                    then /* overflowed MAC,
                                            saturationMode enabled */
                                         result[31:0] = 0xffff_ffff
                         }
                         else product[31:0] = {product[30:0], 0}
                break;
             case 2:     /* reserved encoding */
                break;
             case 3:     /* SF = “>> 1” */
                product[31:0] = {0, product[31:1]}
                break;
         }

         /* combine with accumulator */
         if (MACSR.V == 0)
```

```
                then {if (inst == MSAC)
                         then result[31:0] = acc[31:0] - product[31:0]
                         else result[31:0] = acc[31:0] + product[31:0]
                }

             /* check for accumulation overflow */
             if (accumulationOverflow == 1)
                then {MACSR.V = 1
                      if (inst == MSAC && MACSR.OMC == 1)
                         then result[31:0] = 0x0000_0000
                         else if (MACSR.OMC == 1)
                              then /* overflowed MAC,
                                      saturationMode enabled */
                                   result[31:0] = 0xffff_ffff
                }

             /* transfer the result to the accumulator */
             acc[31:0] = result[31:0]
             MACSR.N = result[31]
             if (result[31:0] == 0x0000_0000)
                then MACSR.Z = 1
                else MACSR.Z = 0
        }
   break;}
```

Chapter 10
Interrupt Controller (CF1_INTC)

10.1 Introduction

The CF1_INTC interrupt controller (CF1_INTC) is intended for use in low-cost microcontroller designs using the Version 1 (V1) ColdFire processor core. In keeping with the general philosophy for devices based on this low-end 32-bit processor, the interrupt controller generally supports less programmability compared to similar modules in other ColdFire microcontrollers and embedded microprocessors. However, CF1_INTC provides the required functionality with a minimal silicon cost.

These requirements guide the CF1_INTC module definition to support Freescale's Controller Continuum:

- The priorities of the interrupt requests between comparable HCS08 and V1 ColdFire devices are identical.
- Supports a mode of operation (through software convention with hardware assists) equivalent to the S08's interrupt processing with only one level of nesting.
- Leverages the current ColdFire interrupt controller programming model and functionality, but with a minimal hardware implementation and cost.

Table 10-1 provides a high-level architectural comparison between HCS08 and ColdFire exception processing as these differences are important in the definition of the CF1_INTC module. Throughout this document, the term IRQ refers to an interrupt request and ISR refers to an interrupt service routine to process an interrupt exception.

Table 10-1. Exception Processing Comparison

Attribute	HCS08	V1 ColdFire
Exception Vector Table	32 two-byte entries, fixed location at upper end of memory	103 four-byte entries, located at lower end of memory at reset, relocatable with the VBR
More on Vectors	2 for CPU + 30 for IRQs, reset at upper address	64 for CPU + 39 for IRQs, reset at lowest address
Exception Stack Frame	5-byte frame: CCR, A, X, PC	8-byte frame: F/V, SR, PC; General-purpose registers (An, Dn) must be saved/restored by the ISR
Interrupt Levels	1 = *f*(CCR[I])	7= *f* (SR[I]) with automatic hardware support for nesting
Non-Maskable IRQ Support	No	Yes, with level 7 interrupts
Core-enforced IRQ Sensitivity	No	Level 7 is edge sensitive, else level sensitive
INTC Vectoring	Fixed priorities and vector assignments	Fixed priorities and vector assignments, plus any 2 IRQs can be remapped as the highest priority level 6 requests

Table 10-1. Exception Processing Comparison (continued)

Attribute	HCS08	V1 ColdFire
Software IACK	No	Yes
Exit Instruction from ISR	RTI	RTE

10.1.1 Overview

Interrupt exception processing includes interrupt recognition, aborting the current instruction execution stream, storing an 8-byte exception stack frame in the memory, calculation of the appropriate vector, and passing control to the specified interrupt service routine.

Unless specifically noted otherwise, all ColdFire processors sample for interrupts once during each instruction's execution during the first cycle of execution in the OEP. Additionally, all ColdFire processors use an instruction restart exception model.

The ColdFire processor architecture defines a 3-bit interrupt priority mask field in the processor's status register (SR[I]). This field, and the associated hardware, support seven levels of interrupt requests with the processor providing automatic nesting capabilities. The levels are defined in descending numeric order with 7 > 6 ... > 1. Level 7 interrupts are treated as non-maskable, edge-sensitive requests while levels 6–1 are maskable, level-sensitive requests. The SR[I] field defines the processor's current interrupt level. The processor continuously compares the encoded IRQ level from CF1_INTC against SR[I]. Recall that interrupt requests are inhibited for all levels less than or equal to the current level, except the edge-sensitive level 7 request that cannot be masked.

Exception processing for ColdFire processors is streamlined for performance and includes all actions from detecting the fault condition to the initiation of fetch for the first handler instruction. Exception processing is comprised of four major steps.

1. The processor makes an internal copy of the status register (SR) and enters supervisor mode by setting SR[S] and disabling trace mode by clearing SR[T]. The occurrence of an interrupt exception also forces the master mode (M) bit to clear and the interrupt priority mask (I) to set to the level of the current interrupt request.
2. The processor determines the exception vector number. For all faults except interrupts, the processor performs this calculation based on the exception type. For interrupts, the processor performs an IACK bus cycle to obtain the vector number from the interrupt controller if CPUCR[IAE] equals 1. The IACK cycle is mapped to special locations within the interrupt controller's IPS address space with the interrupt level encoded in the address. If CPUCR[IAE] equals 0, the processor uses the vector number supplied by the interrupt controller at the time the request was signaled (for improved performance).
3. The processor saves the current context by creating an exception stack frame on the system stack. As a result, exception stack frame is created at a 0-modulo-4 address on top of the system stack defined by the supervisor stack pointer (SSP). The processor uses an 8-byte stack frame for all exceptions. It contains the vector number of the exception, the contents of the status register at the time of the exception, and the program counter (PC) at the time of the exception. The exception

type determines whether the program counter placed in the exception stack frame defines the location of the faulting instruction (fault) or the address of the next instruction to be executed (next). For interrupts, the stacked PC is always the address of the next instruction to be executed.

4. The processor calculates the address of the first instruction of the exception handler. By definition, the exception vector table is aligned on a 1MB boundary. This instruction address is generated by fetching a 32-bit exception vector from the table located at the address defined in the vector base register (VBR). The index into the exception table is calculated as (4 × vector number). After the exception vector has been fetched, the contents of the vector serves as a 32-bit pointer to the address of the first instruction of the desired handler. After the instruction fetch for the first opcode of the handler has been initiated, exception processing terminates and normal instruction processing continues in the handler.

All ColdFire processors support a 1024-byte vector table aligned on any 1-MB address boundary. For the V1 ColdFire core, the only practical locations for the vector table are based at 0x(00)00_0000 in the flash or 0x(00)80_0000 in the RAM. The table contains 256 exception vectors; the first 64 are reserved for internal processor exceptions, and the remaining 192 are device-specific interrupt vectors. The IRQ assignment table is partially populated depending on the exact set of peripherals for the given device.

The exception vector table for MCF51EM256 series devices is shown in Table 10-2.

Table 10-2. MCF51EM256/128 Exception Vector Table

Vector Number(s)	Interrupt Source Number	Vector Address Offset	Interrupt Level	Priority within Level	Stacked Program Counter	Assignment
0	—	0x000	—	N/A	—	Initial supervisor stack pointer
1	—	0x004	—	N/A	—	Initial program counter
2–63	—	0x008–0x0FC	—	N/A	—	Reserved for internal CPU exceptions
64	0	0x100	7	mid	Next	IRQ_pin
65	1	0x104	7	3	Next	Low_voltage_detect
66	2	0x108	7	2	Next	Reserved
67	3	0x10C	7	1	Next	PDB_ERR
	INTC_PL6P7		6	7		Reserved for remapped vector #1
	INTC_PL6P6		6	6		Reserved for remapped vector #2
68	4	0x110	6	5	Next	PDB
69	5	0x114	6	4	Next	TPM1_ch0
70	6	0x118	6	3	Next	TPM1_ch1
71	7	0x11C	6	2	Next	TPM1_ovfl
72	8	0x120	6	1	Next	Reserved
73	9	0x124	5	7	Next	MTIM2_ovfl
74	10	0x128	5	6	Next	MTIM3_ovfl

Table 10-2. MCF51EM256/128 Exception Vector Table (continued)

Vector Number(s)	Interrupt Source Number	Vector Address Offset	Interrupt Level	Priority within Level	Stacked Program Counter	Assignment
75	11	0x12C	5	5	Next	MTIM1_ovfl
76	12	0x130	5	4	Next	ADC4
77	13	0x134	5	3	Next	ADC3
78	14	0x138	5	2	Next	ADC2
79	15	0x13C	5	1	Next	ADC1
80	16	0x140	4	7	Next	SPI1
81	17	0x144	4	6	Next	SPI2
82	18	0x148	4	5	Next	SPI3
83	19	0x14C	4	4	Next	SCI1_err
84	20	0x150	4	3	Next	SCI1_rx
85	21	0x154	4	2	Next	SCI1_tx
86	22	0x158	4	1	Next	Reserved
87	23	0x15C	3	7	Next	SCI2_err
88	24	0x160	3	6	Next	SCI2_rx
89	25	0x164	3	5	Next	SCI2_tx
90	26	0x168	3	4	Next	SCI3_err
91	27	0x16C	3	3	Next	SCI3_rx
92	28	0x170	3	2	Next	SCI3_tx
93	29	0x174	3	1	Next	Reserved
94	30	0x178	2	7	Next	Reserved
95	31	0x17C	2	6	Next	IIC
96	32	0x180	2	5	Next	CMP1
97	33	0x184	2	4	Next	CMP2
98	34	0x188	2	3	Next	Reserved
99	35	0x18C	2	2	Next	KBI1
100	36	0x190	2	1	Next	KBI2
101	37	0x194	1	7	Next	Reserved
102	38	0x198	1	6	Next	IRTC
103	INTC_FRC[56]	0x19C	7	0	Next	Level 7 Software Interrupt
104	INTC_FRC[57]	0x1A0	6	0	Next	Level 6 Software Interrupt
105	INTC_FRC[58]	0x1A4	5	0	Next	Level 5 Software Interrupt

Table 10-2. MCF51EM256/128 Exception Vector Table (continued)

Vector Number(s)	Interrupt Source Number	Vector Address Offset	Interrupt Level	Priority within Level	Stacked Program Counter	Assignment
106	INTC_FRC[59]	0x1A8	4	0	Next	Level 4 Software Interrupt
107	INTC_FRC[60]	0x1AC	3	0	Next	Level 3 Software Interrupt
108	INTC_FRC[61]	0x1B0	2	0	Next	Level 2 Software Interrupt
109	INTC_FRC[62]	0x1B4	1	0	Next	Level 1 Software Interrupt
110	39	0x1B8	1	5	Next	LCD
111	40	0x1BC	1	4	Next	Reserved
112	41	0x1C0	1	3	Next	FTSR1
113	42	0x1C4	1	2	Next	FTSR2
114	43	0x1C8	1	1	Next	Reserved
115–255	—	0x1CC–0x3FC	N/A	N/A	Next	Reserved

The basic ColdFire interrupt controller supports up to 63 request sources mapped as nine priorities for each of the seven supported levels (7 levels × 9 priorities per level). Within the nine priorities within a level, the mid-point is typically reserved for package-level IRQ inputs. The levels and priorities within the level follow a descending order: 7 > 6 > ... > 1 > 0.

The HCS08 architecture supports a 32-entry exception vector table: the first two vectors are reserved for internal CPU/system exceptions and the remaining are available for I/O interrupt requests. The requirement for an exact match between the interrupt requests and priorities across two architectures means the sources are mapped to a sparsely-populated two-dimensional ColdFire array of seven interrupt levels and nine priorities within the level. The following association between the HCS08 and ColdFire vector numbers applies:

```
ColdFire Vector Number = 62 + HCS08 Vector Number
```

The CF1_INTC performs a cycle-by-cycle evaluation of the active requests and signals the highest-level, highest-priority request to the V1 ColdFire core in the form of an encoded interrupt level and the exception vector associated with the request. The module also includes a byte-wide interface to access its programming model. These interfaces are shown in the simplified block diagram of Figure 10-1.

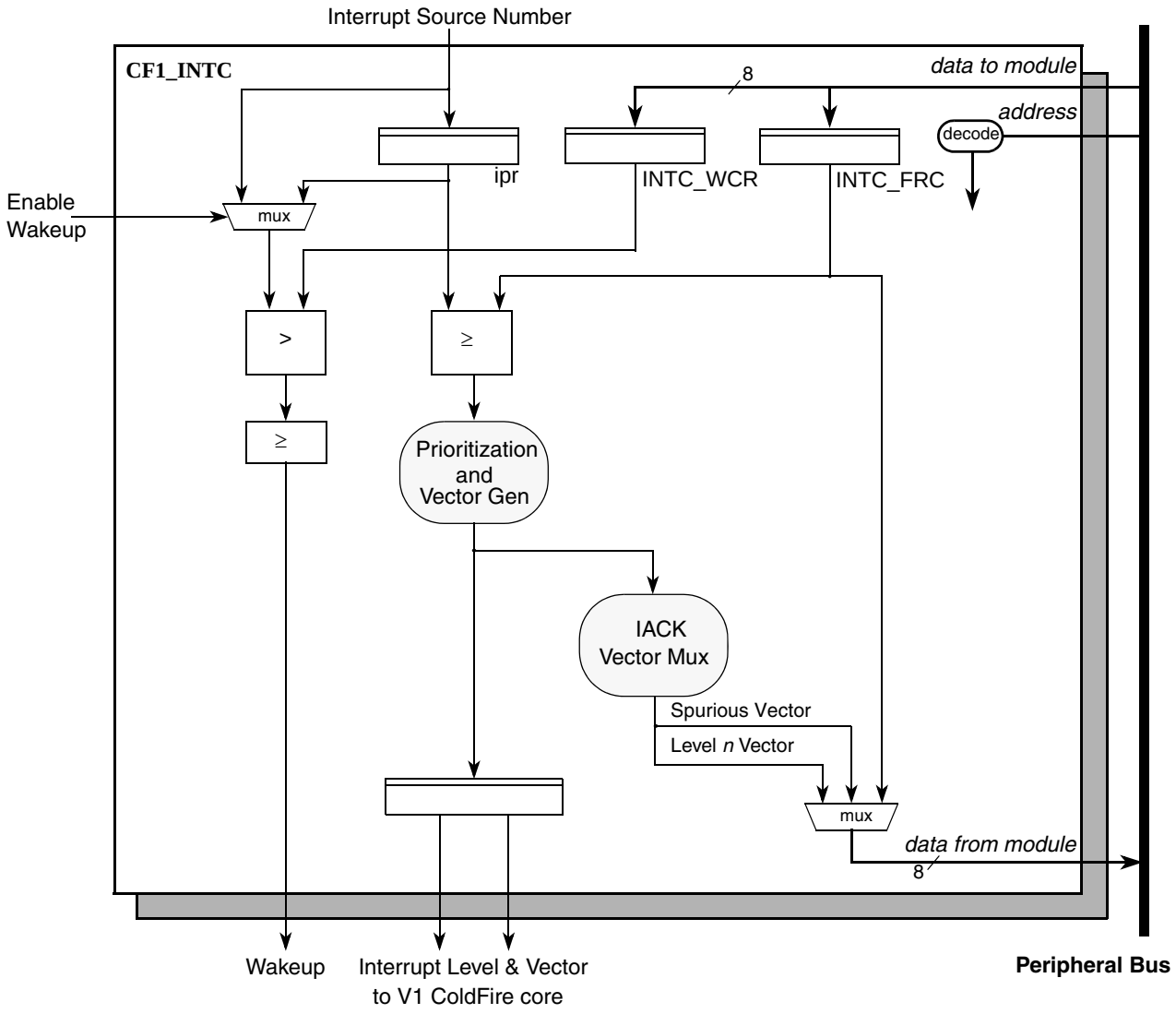


Figure 10-1. CF1_INTC Block Diagram

10.1.2 Features

The Version 1 ColdFire interrupt controller includes:

- Memory-mapped off-platform slave module
  - — 64-byte space located at top end of memory: 0x(FF)FF_FFC0–0x(FF)FF_FFFF
  - — Programming model accessed via the peripheral bus
  - — Encoded interrupt level and vector sent directly to processor core
- Support of 30 peripheral I/O interrupt requests plus seven software (one per level) interrupt requests
- Fixed association between interrupt request source and level plus priority
  - — 30 I/O requests assigned across seven available levels and nine priorities per level

  - Exactly matches HCS08 interrupt request priorities
  - Up to two requests can be remapped to the highest maskable level + priority
- Unique vector number for each interrupt source
  - ColdFire vector number = 62 + HCS08 vector number
  - Details on IRQ and vector assignments are device-specific
- Support for service routine interrupt acknowledge (software IACK) read cycles for improved system performance
- Combinatorial path provides wakeup signal from wait and stop modes

10.1.3 Modes of Operation

The CF1_INTC module does not support any special modes of operation. As a memory-mapped slave peripheral located on the platform's slave bus, it responds based strictly on the memory addresses of the connected bus.

One special behavior of the CF1_INTC deserves mention. When the device enters a wait or stop mode and certain clocks are disabled, there is an input signal that can be asserted to enable a purely-combinational logic path for monitoring the assertion of an interrupt request. After a request of unmasked level is asserted, this combinational logic path asserts an output signal that is sent to the clock generation logic to re-enable the internal device clocks to exit the low-power mode.

10.2 External Signal Description

The CF1_INTC module does not include any external interfaces.

10.3 Memory Map/Register Definition

The CF1_INTC module provides a 64-byte programming model mapped to the upper region of the 16 MB address space. All the register names are prefixed with INTC_ as an abbreviation for the full module name.

The programming model is referenced using 8-bit accesses. Attempted references to undefined (reserved) addresses or with a non-supported access type (for example, a write to a read-only register) generate a bus error termination.

The programming model follows the definition from previous ColdFire interrupt controllers. This compatibility accounts for the various memory holes in this module's memory map.

The CF1_INTC module is based at address 0x(FF)FF_FFC0 (referred to as CF1_INTC_BASE throughout the chapter) and occupies the upper 64 bytes of the peripheral space. The module memory map is shown in Table 10-3.

Table 10-3. CF1_INTC Memory Map

Offset Address	Register Name	Register Description	Width (bits)	Access	Reset Value	Section/ Page
0x10	INTC_FRC	CF1_INTC Force Interrupt Register	8	R/W	0x00	10.3.1/10-8
0x18	INTC_PL6P7	CF1_INTC Programmable Level 6, Priority 7	8	R/W	0x00	10.3.2/10-9
0x19	INTC_PL6P6	CF1_INTC Programmable Level 6, Priority 6	8	R/W	0x00	10.3.2/10-9
0x1B	INTC_WCR	CF1_INTC Wakeup Control Register	8	R/W	0x80	10.3.3/10-10
0x1E	INTC_SFRC	CF1_INTC Set Interrupt Force Register	8	Write	—	10.3.4/10-11
0x1F	INTC_CFRC	CF1_INTC Clear Interrupt Force Register	8	Write	—	10.3.5/10-12
0x20	INTC_SWIACK	CF1_INTC Software Interrupt Acknowledge	8	Read	0x00	10.3.6/10-13
0x24	INTC_LVL1IACK	CF1_INTC Level 1 Interrupt Acknowledge	8	Read	0x18	10.3.6/10-13
0x28	INTC_LVL2IACK	CF1_INTC Level 2 Interrupt Acknowledge	8	Read	0x18	10.3.6/10-13
0x2C	INTC_LVL3IACK	CF1_INTC Level 3 Interrupt Acknowledge	8	Read	0x18	10.3.6/10-13
0x30	INTC_LVL4IACK	CF1_INTC Level 4 Interrupt Acknowledge	8	Read	0x18	10.3.6/10-13
0x34	INTC_LVL5IACK	CF1_INTC Level 5 Interrupt Acknowledge	8	Read	0x18	10.3.6/10-13
0x38	INTC_LVL6IACK	CF1_INTC Level 6 Interrupt Acknowledge	8	Read	0x18	10.3.6/10-13
0x3C	INTC_LVL7IACK	CF1_INTC Level 7 Interrupt Acknowledge	8	Read	0x18	10.3.6/10-13

10.3.1 Force Interrupt Register (INTC_FRC)

The INTC_FRC register allows software to generate a unique interrupt for each possible level at the lowest priority within the level for functional or debug purposes. These interrupts may be self-scheduled by setting one or more of the bits in the INTC_FRC register. In some cases, the handling of a normal interrupt request may cause critical processing by the service routine along with the scheduling (using the INTC_FRC register) of a lower priority level interrupt request to be processed at a later time for less-critical task handling.

The INTC_FRC register may be modified directly using a read-modify-write sequence or through a simple write operation using the set/clear force interrupt registers (INTC_SFRC, INTC_CFRC).

NOTE

Take special notice of the bit numbers within this register, 63–56. This is for compatibility with other ColdFire interrupt controllers.

Offset: CF1_INTC_BASE + 0x10 (INTC_FRC) Access: Read/Write

	63	62	61	60	59	58	57	56
R	0	LVL1	LVL2	LVL3	LVL4	LVL5	LVL6	LVL7
W								
Reset	0	0	0	0	0	0	0	0

Figure 10-2. Force Interrupt Register (INTC_FRC)

Table 10-4. INTC_FRC Field Descriptions

Field	Description
63	Reserved, must be cleared.
62 LVL1	Force Level 1 interrupt. 0 Negates the forced level 1 interrupt request. 1 Forces a level 1 interrupt request.
61 LVL2	Force Level 2 interrupt. 0 Negates the forced level 2 interrupt request. 1 Forces a level 2 interrupt request.
60 LVL3	Force Level 3 interrupt. 0 Negates the forced level 3 interrupt request. 1 Forces a level 3 interrupt request.
59 LVL4	Force Level 4 interrupt. 0 Negates the forced level 4 interrupt request. 1 Forces a level 4 interrupt request.
58 LVL5	Force Level 5 interrupt. 0 Negates the forced level 5 interrupt request. 1 Forces a level 5 interrupt request.
57 LVL6	Force Level 6 interrupt. 0 Negates the forced level 6 interrupt request. 1 Forces a level 6 interrupt request.
55 LVL7	Force Level 7 interrupt. 0 Negates the forced level 7 interrupt request. 1 Forces a level 7 interrupt request.

10.3.2 INTC Programmable Level 6, Priority {7,6} Registers (INTC_PL6P{7,6})

The level seven interrupt requests cannot have their levels reassigned. However, any of the remaining peripheral interrupt requests can be reassigned as the highest priority maskable requests using these two registers (INTC_PL6P7 and INTC_PL6P6). The vector number associated with the interrupt requests does not change. Rather, only the interrupt request's level and priority are altered, based on the contents of the INTC_PL6P{7,6} registers.

NOTE

The requests associated with the INTC_FRC register have a fixed level and priority that cannot be altered.

The INTC_PL6P7 register specifies the highest-priority, maskable interrupt request that is defined as the level six, priority seven request. The INTC_PL6P6 register specifies the second-highest-priority, maskable interrupt request defined as the level six, priority six request. Reset clears both registers, disabling any request re-mapping.

For an example of the use of these registers, see Section 10.6.2, “Using INTC_PL6P{7,6} Registers.”

Offset: CF1_INTC_BASE + 0x18 (INTC_PL6P7)
CF1_INTC_BASE + 0x19 (INTC_PL6P6)

Access: Read/Write

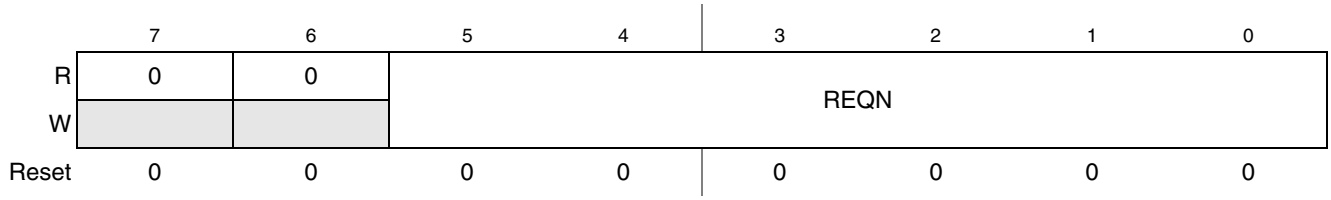


Figure 10-3. Programmable Level 6, Priority {7,6} Registers (INTC_PL6P{7,6})

Table 10-5. INTC_PL6P{7,6} Field Descriptions

Field	Description
7–6	Reserved, must be cleared.
5–0 REQN	Request number. Defines the peripheral IRQ number to be remapped as the level 6, priority 7 (for INTC_PL6P7) request and level 6, priority 6 (for INTC_PL6P6). **Note:** The value must be in a valid interrupt number. Unused or reserved interrupt numbers are ignored.

10.3.3 INTC Wakeup Control Register (INTC_WCR)

The interrupt controller provides a combinatorial logic path to generate a special wakeup signal to exit from the wait or stop modes. The INTC_WCR register defines wakeup condition for interrupt recognition during wait and stop modes. This mode of operation works as follows:

1. Write to the INTC_WCR to enable this operation (set INTC_WCR[ENB]) and define the interrupt mask level needed to force the core to exit wait or stop mode (INTC_WCR[MASK]). The maximum value of INTC_WCR[MASK] is 0x6 (0b110). The INTC_WCR is enabled with a mask level of 0 as the default after reset.
2. Execute a stop instruction to place the processor into wait or stop mode.
3. After the processor is stopped, the interrupt controller enables special logic that evaluates the incoming interrupt sources in a purely combinatorial path; no clocked storage elements are involved.
4. If an active interrupt request is asserted and the resulting interrupt level is greater than the mask value contained in INTC_WCR[MASK], the interrupt controller asserts the wakeup output signal. This signal is routed to the clock generation logic to exit the low-power mode and resume processing.

Typically, the interrupt mask level loaded into the processor's status register field (SR[I]) during the execution of the stop instruction matches the INTC_WCR[MASK] value.

The interrupt controller's wakeup signal is defined as:

```
wakeup = INTC_WCR[ENB] & (level of any asserted_int_request > INTC_WCR[MASK])
```

Offset: CF1_INTC_BASE + 0x1B (INTC_WCR) Access: Read/Write

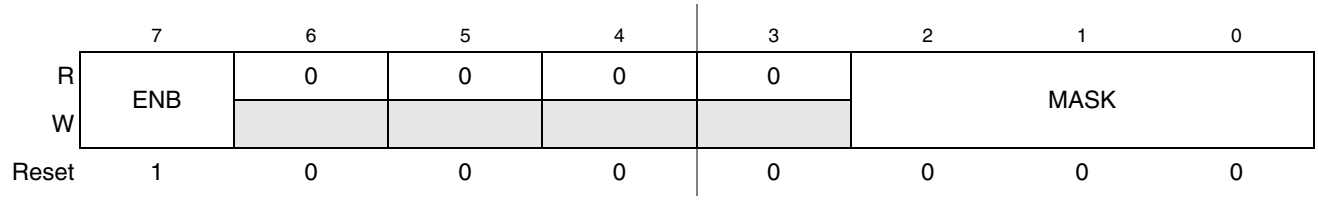


Figure 10-4. Wakeup Control Register (INTC_WCR)

Table 10-6. INTC_WCR Field Descriptions

Field	Description
7 ENB	Enable wakeup signal. 0 Wakeup signal disabled 1 Enables the assertion of the combinational wakeup signal to the clock generation logic.
6–3	Reserved, must be cleared.
2–0 MASK	Interrupt mask level. Defines the interrupt mask level during wait or stop mode and is enforced by the hardware to be within the range 0–6. If INTC_WCR[ENB] is set, when an interrupt request of a level higher than MASK occurs, the interrupt controller asserts the wakeup signal to the clock generation logic.

10.3.4 INTC Set Interrupt Force Register (INTC_SFRC)

The INTC_SFRC register provides a simple memory-mapped mechanism to set a given bit in the INTC_FRC register to assert a specific level interrupt request. The data value written causes the appropriate bit in the INTC_FRC register to be set. Attempted reads of this register generate an error termination.

This register is provided so interrupt service routines can generate a forced interrupt request without the need to perform a read-modify-write sequence on the INTC_FRC register.

Offset: CF1_INTC_BASE + 0x1E (INTC_SFRC) Access: Write-only

Figure 10-5. INTC_SFRC Register

Table 10-7. INTC_SFRC Field Descriptions

Field	Description
7–6	Reserved, must be cleared.
5–0 SET	For data values within the 56–62 range, the corresponding bit in the INTC_FRC register is set, as defined below. 0x38 Bit 56, INTC_FRC[LVL7] is set 0x39 Bit 57, INTC_FRC[LVL6] is set 0x3A Bit 58, INTC_FRC[LVL5] is set 0x3B Bit 59, INTC_FRC[LVL4] is set 0x3C Bit 60, INTC_FRC[LVL3] is set 0x3D Bit 61, INTC_FRC[LVL2] is set 0x3E Bit 62, INTC_FRC[LVL1] is set **Note:** Data values outside this range do not affect the INTC_FRC register. It is recommended the data values be restricted to the 0x38–0x3E (56–62) range to ensure compatibility with future devices.

10.3.5 INTC Clear Interrupt Force Register (INTC_CFRC)

The INTC_CFRC register provides a simple memory-mapped mechanism to clear a given bit in the INTC_FRC register to negate a specific level interrupt request. The data value on the register write causes the appropriate bit in the INTC_FRC register to be cleared. Attempted reads of this register generate an error termination.

This register is provided so interrupt service routines can negate a forced interrupt request without the need to perform a read-modify-write sequence on the INTC_FRC register.

Offset: CF1_INTC_BASE + 0x1F (INTC_CFRC) Access: Write-only

	7	6	5	4	3	2	1	0
R								
W	0	0	CLR					
Reset	0	0	0	0	0	0	0	0

Figure 10-6. INTC_CFRC Register

Table 10-8. INTC_CFRC Field Descriptions

Field	Description
7–6	Reserved, must be cleared.
5–0 CLR	For data values within the 56–62 range, the corresponding bit in the INTC_FRC register is cleared, as defined below. 0x38 Bit 56, INTC_FRC[LVL7] is cleared 0x39 Bit 57, INTC_FRC[LVL6] is cleared 0x3A Bit 58, INTC_FRC[LVL5] is cleared 0x3B Bit 59, INTC_FRC[LVL4] is cleared 0x3C Bit 60, INTC_FRC[LVL3] is cleared 0x3D Bit 61, INTC_FRC[LVL2] is cleared 0x3E Bit 62, INTC_FRC[LVL1] is cleared **Note:** Data values outside this range do not affect the INTC_FRC register. It is recommended the data values be restricted to the 0x38–0x3E (56–62) range to ensure compatibility with future devices.

10.3.6 INTC Software and Level-*n* IACK Registers (*n* = 1,2,3,...,7)

The eight read-only interrupt acknowledge (IACK) registers can be explicitly addressed by the memory-mapped accesses or implicitly addressed by a processor-generated interrupt acknowledge cycle during exception processing when CPUCR[IAE] is set. In either case, the interrupt controller's actions are similar.

First, consider an IACK cycle to a specific level, a level-n IACK. When this type of IACK arrives in the interrupt controller, the controller examines all currently-active level-n interrupt requests, determines the highest priority within the level, and then responds with the unique vector number corresponding to that specific interrupt source. The vector number is supplied as the data for the byte-sized IACK read cycle.

If there is no active interrupt source at the time of the level-*n* IACK, a special spurious interrupt vector (vector number 24 (0x18)) is returned. It is the responsibility of the service routine to manage this error situation.

This protocol implies the interrupting peripheral is not accessed during the acknowledge cycle because the interrupt controller completely services the acknowledge. This means the interrupt source must be explicitly disabled in the peripheral device by the interrupt service routine. This approach provides unique vector capability for all interrupt requests, regardless of the complexity of the peripheral device.

Second, the interrupt controller also supports the concept of a software IACK. This is the ability to query the interrupt controller near the end of an interrupt service routine (after the current interrupt request has been negated) to determine if there are any pending (but currently masked) interrupt requests. If the response to the software IACK's byte operand read is non-zero, the service routine uses the returned value as the vector number of the highest pending interrupt request and passes control to the appropriate new handler. If the returned value is zero, there is no pending interrupt request.

This process avoids the overhead of a context restore and RTE instruction execution, followed immediately by another interrupt exception and context save. In system environments with high rates of interrupt activity, this mechanism can noticeably improve overall performance. For additional details on software IACKs, see Section 10.6.3, “More on Software IACKs.”

Offset: CF1_INTC_BASE + 0x20 (INTC_SWIACK) Access: Read-only
CF1_INTC_BASE + 0x20 + (4×*n*) (INTC_LVL*n*IACK)

	7	6	5	4	3	2	1	0
R	0	VECN						
W								
SWIACK Reset	0	0	0	0	0	0	0	0
LVL*n*IACK Reset	0	0	0	1	1	0	0	0

Table 10-9. Software and Level-n IACK Registers (INTC_SWIACK, INTC_LVL*n*IACK)

Table 10-10. INTC_SWIACK, INTC_LVL*n*IACK Field Descriptions

Field	Description
7	Reserved, must be cleared.
6–0 VECN	Vector number. Indicates the appropriate vector number. For the SWIACK register, it is the highest-level, highest-priority request currently being asserted in the CF1_INTC module. If there are no pending requests, VECN is zero. For the LVL*n*IACK register, it is the highest priority request within the specified level-*n*. If there are no pending requests within the level, VECN is 0x18 (24) to signal a spurious interrupt.

10.3.7 Interrupt Request Level and Priority Assignments

This section provides a two-dimensional view of levels and priorities within the level (Table 10-12).

The CF1_INTC module implements a sparsely-populated 7 × 9 matrix of levels (7) and priorities within each level (9). In this representation, the leftmost top cell (level 7, priority 7) is the highest interrupt request while the rightmost lowest cell (level 1, priority 0) is the lowest interrupt request. The following legend is used for this table:

Table 10-11. Legend for Table 10-12

Interrupt Request Source	
Interrupt Source Number	Vector Number

NOTE

For remapped and forced interrupts, the interrupt source number entry indicates the register or register field that enables the corresponding interrupt.

Table 10-12. ColdFire [Level][Priority within Level] Matrix Interrupt Assignments

Level	Priority within Level																	
	7		6		5		4		Midpoint		3		2		1		0	
7	—		—		—		—		IRQ_pin		Low_voltage		—		PDB_err		force_lvl7	
									0	64	1	65			3	67	FRC[56]	103
6	remapped		remapped		PDB		TPM1_ch0		—		TPM1_ch1		TPM1_ovfl		—		force_lvl6	
	PL6P7	*	PL6P6	*	4	68	5	69			6	70	7	71			FRC[57]	104
5	MTIM2_ovfl		MTIM3_ovfl		MTIM1_ovfl		ADC4		—		ADC3		ADC2		ADC1		force_lvl5	
	9	73	10	74	11	75	12	76			13	77	14	78	15	79	FRC[58]	105
4	SPI1		SPI2		SPI3		SCI1_err		—		SCI1_rx		SCI1_tx		—		force_lvl4	
	16	80	17	81	18	82	19	83			20	84	21	85			FRC[59]	106

Table 10-12. ColdFire [Level][Priority within Level] Matrix Interrupt Assignments (continued)

Level	Priority within Level															
	7		6		5		4		Midpoint	3		2		1		0	
3	SCI2_err		SCI2_rx		SCI2_tx		SCI3_err		—	SCI3_rx		SCI3_tx		—		force_lvl3	
	23	87	24	88	25	89	26	90		27	91	28	92			FRC[60]	107
2	—		IIC		CMP1		CMP2		—	—		KBI1		KBI2		force_lvl2	
			31	95	32	96	33	97				35	99	36	100	FRC[61]	108
1	—		IRTC		LCD		—		—	FTSR1		FTSR2		—		force_lvl1	
			38	102	39	110				41	112	42	113			FRC[62]	109

10.4 Functional Description

The basic operation of the CF1_INTC is detailed in the preceding sections. This section describes special rules applicable to non-maskable level seven interrupt requests and the module's interfaces.

10.4.1 Handling of Non-Maskable Level 7 Interrupt Requests

The CPU treats level seven interrupts as non-maskable, edge-sensitive requests, while levels one through six are maskable, level-sensitive requests. As a result of this definition, level seven interrupt requests are a special case. The edge-sensitive nature of these requests means the encoded 3-bit level input from the CF1_INTC to the V1 ColdFire core must change state before the CPU detects an interrupt. A non-maskable interrupt (NMI) is generated each time the encoded interrupt level changes to level seven (regardless of the SR[I] field) and each time the SR[I] mask changes from seven to a lower value while the encoded request level remains at seven.

10.5 Initialization Information

The reset state of the CF1_INTC module enables the default IRQ mappings and clears any software-forced interrupt requests (INTC_FRC is cleared). Immediately after reset, the CF1_INTC begins its cycle-by-cycle evaluation of any asserted interrupt requests and forms the appropriate encoded interrupt level and vector information for the V1 Coldfire processor core. The ability to mask individual interrupt requests using the interrupt controller's IMR is always available, regardless of the level of a particular interrupt request.

10.6 Application Information

This section discusses three application topics: emulation of the HCS08's one level interrupt nesting structure, elevating the priority of two IRQs, and more details on the operation of the software interrupt acknowledge (SWIACK) mechanism.

10.6.1 Emulation of the HCS08's 1-Level IRQ Handling

As noted in Table 10-1, the HCS08 architecture specifies a 1-level IRQ nesting capability. Interrupt masking is controlled by CCR[I], the interrupt mask flag: clearing CCR[I] enables interrupts, while setting CCR[I] disables interrupts. The ColdFire architecture defines seven interrupt levels, controlled by the 3-bit interrupt priority mask field in the status register, SR[I], and the hardware automatically supports nesting of interrupts.

To emulate the HCS08's 1-level IRQ capabilities on V1 ColdFire, only two SR[I] settings are used:

- Writing 0 to SR[I] enables interrupts.
- Writing 7 to SR[I] disables interrupts.

The ColdFire core treats the level seven requests as non-maskable, edge-sensitive interrupts.

ColdFire processors inhibit interrupt sampling during the first instruction of all exception handlers. This allows any handler to effectively disable interrupts, if necessary, by raising the interrupt mask level contained in the status register as the first instruction in the ISR. In addition, the V1 instruction set architecture (ISA_C) includes an instruction (STLDSR) that stores the current interrupt mask level and loads a value into the SR. This instruction is specifically intended for use as the first instruction of an interrupt service routine that services multiple interrupt requests with different interrupt levels. For more details see the *ColdFire Family Programmer's Reference Manual*. A MOVE-to-SR instruction also performs a similar function.

To emulate the HCS08's 1-level IRQ nesting mechanisms, the ColdFire implementation enables interrupts by clearing SR[I] (typically when using RTE to return to a process) and disables interrupts upon entering every interrupt service routine by one of three methods:

1. Execution of STLDSR #0x2700 as the first instruction of an ISR.
2. Execution of MOVE.w #0x2700,SR as the first instruction of an ISR.
3. Static assertion of CPUCR[IME] that forces the processor to load SR[I] with seven automatically upon the occurrence of an interrupt exception. Because this method removes the need to execute multi-cycle instructions of #1 or #2, this approach improves system performance.

10.6.2 Using INTC_PL6P{7,6} Registers

Section 10.3.2, "INTC Programmable Level 6, Priority {7,6} Registers (INTC_PL6P{7,6})," describes control registers that provide the ability to dynamically alter the request level and priority of two IRQs. Specifically, these registers provide the ability to reassign two IRQs to be the highest level 6 (maskable) requests. Consider the following example.

Suppose the system operation desires to remap the receive and transmit interrupt requests of a serial communication device (SCI1) as the highest two maskable interrupts. The default assignments for the SCI1 transmit and receive interrupts are:

- sci1_rx = interrupt source 20 = vector 84 = level 4, priority 3
- sci1_tx = interrupt source 21 = vector 85 = level 4, priority 2

To remap these two requests, the INTC_PL6P{7,6} registers are programmed with the desired interrupt source number:

- Setting INTC_PL6P7 to 20 (0x14), remaps sci1_rx as level 6, priority 7.
- Setting INTC_PL6P6 to 21 (0x15), remaps sci1_tx as level 6, priority 6.

The reset state of the INTC_PL6P{7,6} registers disables any request remapping.

10.6.3 More on Software IACKs

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

To illustrate this concept, consider the following ISR code snippet shown in Figure 10-7.

```
                 align   4
                 irqxx_entry:
00588: 4fef fff0 lea     -16(sp),sp         # allocate stack space
0058c: 48d7 0303 movem.l #0x0303,(sp)       # save d0/d1/a0/a1 on stack

                 irqxx_alternate_entry:
00590:
       ....
                 irqxx_swiack:
005c0: 71b8 ffe0 mvz.b   INTC_SWIACK.w,d0   # perform software IACK
005c4: 0c00 0041 cmpi.b  #0x41,d0           # pending IRQ or level 7?
005c8: 6f0a      ble.b   irqxx_exit         # no pending IRQ, then exit
005ca: 91c8      sub.l   a0,a0              # clear a0
005cc: 2270 0c00 move.l  0(a0,d0.l*4),a1    # fetch pointer from xcpt table
005d0: 4ee9 0008 jmp     8(a1)              # goto alternate isr entry point

                 align   4
                 irqxx_exit:
005d4: 4cd7 0303 movem.l (sp),#0x0303       # restore d0/d1/a0/a1
005d8: 4fef 0010 lea     16(sp),sp          # deallocate stack space
005dc: 4e73      rte                        # return from handler
```

Figure 10-7. ISR Code Snippet with SWIACK

This snippet includes the prologue and epilogue for an interrupt service routine as well as code needed to perform software IACK.

At the entry point (`irqxx_entry`), there is a two-instruction prologue to allocate space on the supervisor stack to save the four volatile registers (d0, d1, a0, a1) defined in the ColdFire application binary interface. After saving these registers, the ISR continues at the alternate entry point.

The software IACK is performed near the end of the ISR, after the source of the current interrupt request is negated. First, the appropriate memory-mapped byte location in the interrupt controller is read (PC = 0x5C0). The CF1_INTC module returns the vector number of the highest priority pending request. If no request is pending, zero is returned. The compare instruction is needed to manage a special case

involving pending level seven requests. Because the level seven requests are non-maskable, the ISR is interrupted to service one of these requests. To avoid any race conditions, this check ignores the level seven vector numbers. The result is the conditional branch (PC = 0x5C8) is taken if there are no pending requests or if the pending request is a level seven.

If there is a pending non-level seven request, execution continues with a three instruction sequence to calculate and then branch to the appropriate alternate ISR entry point. This sequence assumes the exception vector table is based at address 0x(00)00_0000 and that each ISR uses the same two-instruction prologue shown here. The resulting alternate entry point is a fixed offset (8 bytes) from the normal entry point defined in the exception vector table.

The ISR epilogue includes a three instruction sequence to restore the volatile registers from the stack and return from the interrupt exception.

This example is intentionally simple, but does show how performing the software IACK and passing control to an alternate entry point when there is a pending but masked interrupt request can avoid the execution of the ISR epilogue, another interrupt exception, and the ISR prologue.

Chapter 11
Internal Clock Source (ICS)

11.1 Introduction

The Internal Clock Source (ICS) module provides several clock source choices for this device. The module contains a frequency-locked loop (FLL) controllable by either an internal or an external reference clock. The module can select the FLL clock, or either of the internal or external reference clocks as a source for the MCU system clock. The selected clock source is passed through a reduced bus divider which allows a lower output clock frequency to be derived. The ICS also controls a crystal oscillator (XOSC), which allows an external crystal, ceramic resonator, or another external clock source to produce the external reference clock.

The MCF51EM256 series devices are unique in that they support an independent real time clock, which includes a dedicated oscillator in a separate power domain. The IRTC oscillator can also be used as the reference clock into the ICS.

11.1.1 Clock Check & Select Function

The “Clock Check & Select” feature of Figure 1-3 and Figure 11-1 is a very simple block used to check the oscillator clocks for activity and control the mux which selects the external clock for the ICS. This function is implemented external to the ICS module itself, and is specific to the MCF51EM256 series devices.

Four registers make up this operation:

Table 11-1. Clock Check & Select Registers

Register	Function
CCSCTRL	Clock Check & Select Control Register
CCSTMR1	8-bit counter incremented by XOSC1 timebase
CCSTMR2	8-bit counter incremented by XOSC2 timebase
CCSTMRIR	8-bit counter incremented via IR clock timebase

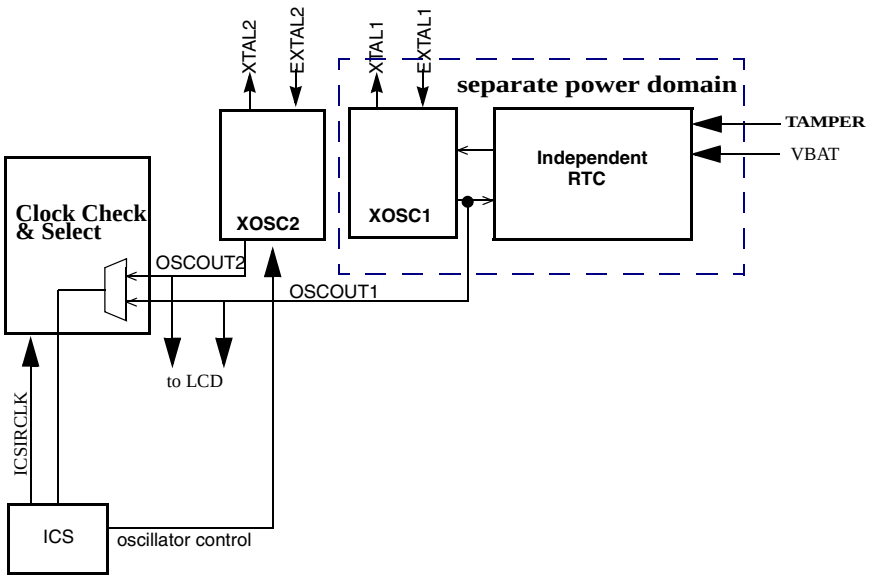


Figure 11-1. ICS Clock Check and Select Block Diagram

11.1.2 Clock Check & Select Control (CCSCTRL)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	EN	TEST	SEL
W								
Reset:	0	0	0	0	0	0	0	0

Figure 11-2. Clock Check & Select Control Register (CSSCTRL)

Table 11-2. CSS Control Register Field Descriptions

Field	Description
7-3 RESERVED	**Reserved** — These bits read as zero. Writes are ignored
2 EN	EN bit 0 = The OSCOUT1, OSCOUT2 and ICSIRCLK inputs to the clock check counters are disabled for power saving. 1 = The clock inputs are enabled.
1 TEST	TEST Writing a “1” to this bit will clear CSSTMR1, CSSTMR2 and CSSTMRIR. The three counters will then begin incrementing until any one of the three hits 0xFF, at which point the test completes, and TEST is cleared.
0 SEL	External Clock Select 0 = XOSC2 is selected as the external clock input to the ICS (default) 1 = XOSC1 is selected as the external clock input to the ICS This bit should only be changed when the ICS is NOT utilizing the external clock input.

11.1.3 CSS XOSC1 Timer Register (CCSTMR1)

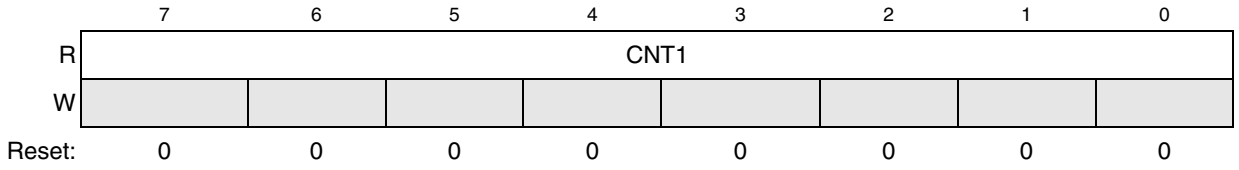


Figure 11-3. CSS XOSC1 Timer Register (CSSTMR1)

Table 11-3. CSSTMR1 Register Field Descriptions

Field	Description
7-0 CNT1	**CNT1** - This register is one of three used to compare XOSC1, XOSC2 and internal relaxation oscillator frequencies. It is initialized to zero upon a write of “1” to CSSCTRL[TEST]. It contains a valid value once CSSCTRL[TEST] resets itself to “0”. By comparing the values of the three registers, application code can determine the crude health of the various clock sources.

11.1.4 CSS XOSC2 Timer Register (CCSTMR2)

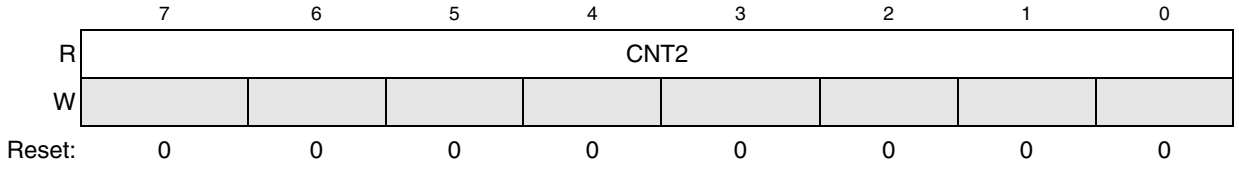


Figure 11-4. CSS XOSC2 Timer Register (CSSTMR2)

Table 11-4. CSSTMR2 Register Field Descriptions

Field	Description
7-0 CNT2	**CNT2** - This register is one of three used to compare XOSC1, XOSC2 and internal relaxation oscillator frequencies. It is initialized to zero upon a write of “1” to CSSCTRL[TEST]. It contains a valid value once CSSCTRL[TEST] resets itself to “0”. By comparing the values of the three registers, application code can determine the crude health of the various clock sources.

11.1.5 CSS Internal Reference Clock Timer Register (CCSTMRIR)

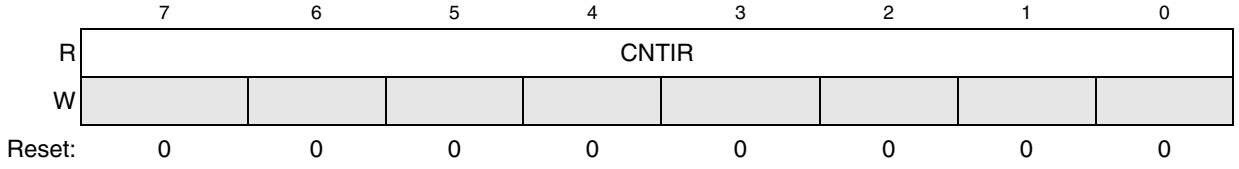


Figure 11-5. CSS Internal Reference Clock Timer Register (CSSTMRIR)

Table 11-5. CSSTMRIR Register Field Descriptions

Field	Description
7-0 CNTIR	**CNTIR** - This register is one of three used to compare XOSC1, XOSC2 and internal relaxation oscillator frequencies. It is initialized to zero upon a write of “1” to CSSCTRL[TEST]. It contains a valid value once CSSCTRL[TEST] resets itself to “0”. By comparing the values of the three registers, application code can determine the crude health of the various clock sources.

11.1.6 Operation

As noted, this module is essentially a glorified multiplexor coupled with a simple test for comparing the relative health of 3-possible on-chip clock sources. The self-test feature allows all three counters to clock for exactly the same amount of time. Any of the three counters hitting the maximum value of 0xFF terminates the test. Software can then compare the three registers to obtain a crude (~1.2%) measure of how well the three frequencies correlate.

This self test can be run at any time.

11.1.7 Features

Key features of the ICS module are:

- Frequency-locked loop (FLL) is trimmable for accuracy
- Internal or external reference clocks can be used to control the FLL
- Reference divider is provided for external clock
- Internal reference clock has 9 trim bits available
- Internal or external reference clocks can be selected as the clock source for the MCU
- Whichever clock is selected as the source can be divided down
  - 2-bit select for clock divider is provided
    - Allowable dividers are: 1, 2, 4, 8
- Control signals for a low power oscillator clock generator (OSCOUT) as the ICS external reference clock are provided
  - HGO, RANGE, EREFS, ERCLKEN, EREFSTEN
- FLL Engaged Internal mode is automatically selected out of reset
- BDC clock is provided as a constant divide by 2 of the low range DCO output
- Three selectable digitally-controlled oscillators (DCO) optimized for different frequency ranges.
- Option to maximize output frequency for a 32768 Hz external reference clock source.

11.1.8 Block Diagram

Figure 11-6 is the ICS block diagram.

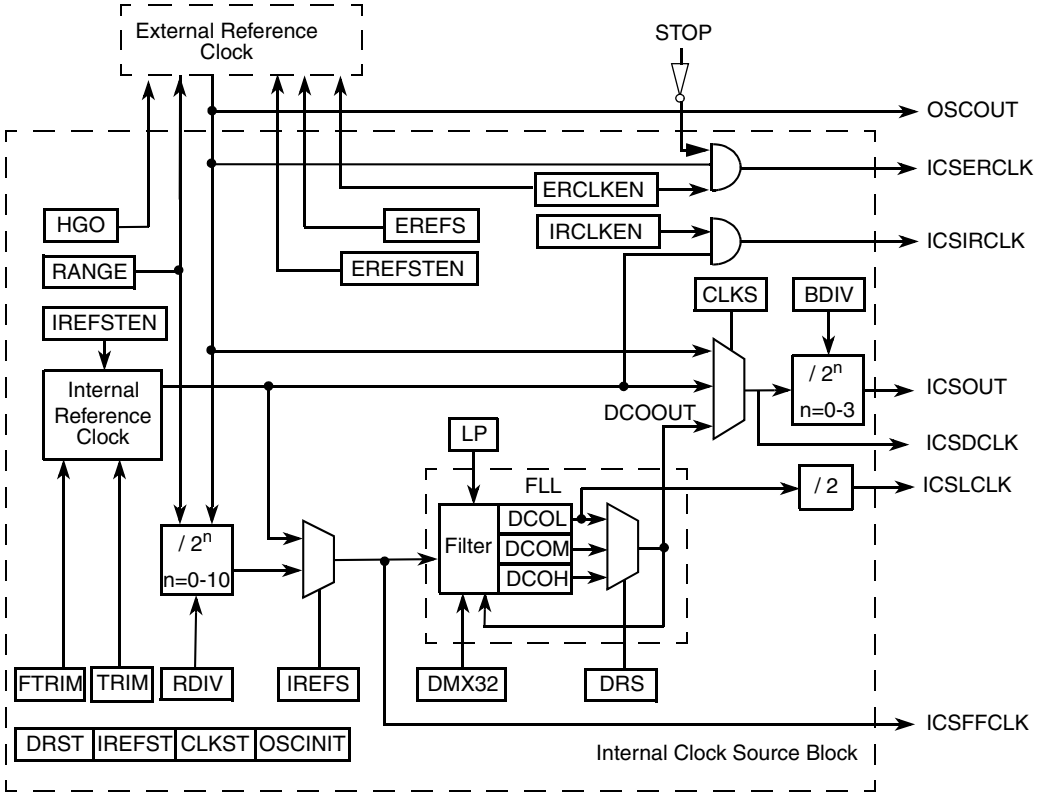


Figure 11-6. Internal Clock Source (ICS) Block Diagram

11.1.9 Modes of Operation

There are seven modes of operation for the ICS: FEI, FEE, FBI, FBILP, FBE, FBELP, and stop.

11.1.9.1 FLL Engaged Internal (FEI)

In FLL engaged internal mode, which is the default mode, the ICS supplies a clock derived from the FLL which is controlled by the internal reference clock. The BDC clock is supplied from the FLL.

11.1.9.2 FLL Engaged External (FEE)

In FLL engaged external mode, the ICS supplies a clock derived from the FLL which is controlled by an external reference clock source. The BDC clock is supplied from the FLL.

11.1.9.3 FLL Bypassed Internal (FBI)

In FLL bypassed internal mode, the FLL is enabled and controlled by the internal reference clock, but is bypassed. The ICS supplies a clock derived from the internal reference clock. The BDC clock is supplied from the FLL.

11.1.9.4 FLL Bypassed Internal Low Power (FBILP)

In FLL bypassed internal low power mode, the FLL is disabled and bypassed, and the ICS supplies a clock derived from the internal reference clock. The BDC clock is not available.

11.1.9.5 FLL Bypassed External (FBE)

In FLL bypassed external mode, the FLL is enabled and controlled by an external reference clock, but is bypassed. The ICS supplies a clock derived from the external reference clock source. The BDC clock is supplied from the FLL.

11.1.9.6 FLL Bypassed External Low Power (FBELP)

In FLL bypassed external low power mode, the FLL is disabled and bypassed, and the ICS supplies a clock derived from the external reference clock. The BDC clock is not available.

11.1.9.7 Stop (STOP)

In stop mode, the FLL is disabled and the internal or the ICS external reference clocks source (OSCOUT) can be selected to be enabled or disabled. The BDC clock is not available and the ICS does not provide an MCU clock source.

NOTE

The DCO frequency changes from the pre-stop value to its reset value and the FLL will need to re-acquire the lock before the frequency is stable. Timing sensitive operations should wait for the FLL acquistition time, tAquire, before executing.

11.2 External Signal Description

There are no ICS signals that connect off chip.

11.3 Register Definition

Figure 11-6 is a summary of ICS registers.

Table 11-6. ICS Register Summary

Name		7	6	5	4	3	2	1	0
ICSC1	R	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
	W								
ICSC2	R	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
	W								
ICSTRM	R	TRIM							
	W								

Table 11-6. ICS Register Summary (continued)

Name		7	6	5	4	3	2	1	0
ICSSC	R	DRST		DMX32	IREFST	CLKST		OSCINIT	FTRIM
	W	DRS							

11.3.1 ICS Control Register 1 (ICSC1)

	7	6	5	4	3	2	1	0
R W	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
Reset:	0	0	0	0	0	1	0	0

Figure 11-7. ICS Control Register 1 (ICSC1)

Table 11-7. ICS Control Register 1 Field Descriptions

Field	Description
7:6 CLKS	**Clock Source Select** — Selects the clock source that controls the bus frequency. The actual bus frequency depends on the value of the BDIV bits. 00 Output of FLL is selected. 01 Internal reference clock is selected. 10 External reference clock is selected. 11 Reserved, defaults to 00.
5:3 RDIV	**Reference Divider** — Selects the amount to divide down the external reference clock. Resulting frequency must be in the range 31.25 kHz to 39.0625 kHz. See Table 11-8 for the divide-by factors.
2 IREFS	**Internal Reference Select** — The IREFS bit selects the reference clock source for the FLL. 1 Internal reference clock selected. 0 External reference clock selected.
1 IRCLKEN	**Internal Reference Clock Enable** — The IRCLKEN bit enables the internal reference clock for use as ICSIRCLK. 1 ICSIRCLK active. 0 ICSIRCLK inactive.
0 IREFSTEN	**Internal Reference Stop Enable** — The IREFSTEN bit controls whether or not the internal reference clock remains enabled when the ICS enters stop mode. 1 Internal reference clock stays enabled in stop if IRCLKEN is set before entering stop. 0 Internal reference clock is disabled in stop.

Table 11-8. Reference Divide Factor

RDIV	RANGE=0	RANGE=1
0	1[1]	32
1	2	64
2	4	128
3	8	256

Table 11-8. Reference Divide Factor

RDIV	RANGE=0	RANGE=1
4	16	512
5	32	1024
6	64	Reserved
7	128	Reserved

[1] Reset default

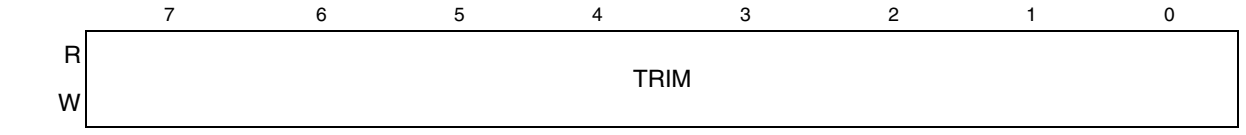
11.3.2 ICS Control Register 2 (ICSC2)

Figure 11-8. ICS Control Register 2 (ICSC2)

Table 11-9. ICS Control Register 2 Field Descriptions

Field	Description
7:6 BDIV	**Bus Frequency Divider** — Selects the amount to divide down the clock source selected by the CLKS bits. This controls the bus frequency. 00 Encoding 0 — Divides selected clock by 1. 01 Encoding 1 — Divides selected clock by 2 (reset default). 10 Encoding 2 — Divides selected clock by 4. 11 Encoding 3 — Divides selected clock by 8.
5 RANGE	**Frequency Range Select** — Selects the frequency range for the external oscillator. 1 High frequency range selected for the external oscillator. 0 Low frequency range selected for the external oscillator.
4 HGO	**High Gain Oscillator Select** — The HGO bit controls the external oscillator mode of operation. 1 Configure external oscillator for high gain operation. 0 Configure external oscillator for low power operation.
3 LP	**Low Power Select** — The LP bit controls whether the FLL is disabled in FLL bypassed modes. 1 FLL is disabled in bypass modes unless BDM is active. 0 FLL is not disabled in bypass mode.
2 EREFS	**External Reference Select** — The EREFS bit selects the source for the external reference clock. 1 Oscillator requested. 0 External Clock Source requested.
1 ERCLKEN	**External Reference Enable** — The ERCLKEN bit enables the external reference clock for use as ICSERCLK. 1 ICSERCLK active. 0 ICSERCLK inactive.
0 EREFSTEN	**External Reference Stop Enable** — The EREFSTEN bit controls whether or not the external reference clock source (OSCOUT) remains enabled when the ICS enters stop mode. 1 External reference clock source stays enabled in stop if ERCLKEN is set before entering stop. 0 External reference clock source is disabled in stop.

11.3.3 ICS Trim Register (ICSTRM)



Reset: Note: TRIM is loaded during reset from a factory programmed location when not in BDM mode. If in a BDM mode, a default value of 0x80 is loaded.

Figure 11-9. ICS Trim Register (ICSTRM)

Table 11-10. ICS Trim Register Field Descriptions

Field	Description
7:0 TRIM	**ICS Trim Setting** — The TRIM bits control the internal reference clock frequency by controlling the internal reference clock period. The bits' effect are binary weighted (in other words, bit 1 adjusts twice as much as bit 0). Increasing the binary value in TRIM will increase the period, and decreasing the value will decrease the period. An additional fine trim bit is available in ICSSC as the FTRIM bit.

11.3.4 ICS Status and Control (ICSSC)

	7	6	5	4	3	2	1	0
R	DRST		DMX32	IREFST	CLKST		OSCINIT	FTRIM[1]
W	DRS							
Reset:	0	0	0	1	0	0	0	

Figure 11-10. ICS Status and Control Register (ICSSC)

1 FTRIM is loaded during reset from a factory programmed location when not in any BDM mode. If in a BDM mode, FTRIM gets loaded with a value of 1'b0.

Table 11-11. ICS Status and Control Register Field Descriptions

Field	Description
7-6 DRST DRS	**DCO Range Status** — The DRST read field indicates the current frequency range for the FLL output, DCOOUT. See Table 11-12. The DRST field does not update immediately after a write to the DRS field due to internal synchronization between clock domains. Writing the DRS bits to 2'b11 is ignored and the DRST bits remain with the current setting. **DCO Range Select** — The DRS field selects the frequency range for the FLL output, DCOOUT. Writes to the DRS field while the LP bit is set are ignored. 00 Low range. 01 Mid range. 10 High range. 11 Reserved.
5 DMX32	**DCO Maximum frequency with 32.768 kHz reference** — The DMX32 bit controls whether or not the DCO frequency range is narrowed to its maximum frequency with a 32.768 kHz reference. See Table 11-12. 0 DCO has default range of 25%. 1 DCO is fined tuned for maximum frequency with 32.768 kHz reference.
4 IREFST	**Internal Reference Status** — The IREFST bit indicates the current source for the reference clock. The IREFST bit does not update immediately after a write to the IREFS bit due to internal synchronization between clock domains. 0 Source of reference clock is external clock. 1 Source of reference clock is internal clock.

Table 11-11. ICS Status and Control Register Field Descriptions (continued)

Field	Description
3-2 CLKST	**Clock Mode Status** — The CLKST bits indicate the current clock mode. The CLKST bits don't update immediately after a write to the CLKS bits due to internal synchronization between clock domains. 00 Output of FLL is selected. 01 FLL Bypassed, Internal reference clock is selected. 10 FLL Bypassed, External reference clock is selected. 11 Reserved.
1 OSCINIT	**OSC Initialization** — If the external reference clock is selected by ERCLKEN or by the ICS being in FEE, FBE, or FBELP mode, and if EREFS is set, then this bit is set after the initialization cycles of the external oscillator clock have completed. This bit is only cleared when either ERCLKEN or EREFS are cleared.
0 FTRIM	**ICS Fine Trim** — The FTRIM bit controls the smallest adjustment of the internal reference clock frequency. Setting FTRIM will increase the period and clearing FTRIM will decrease the period by the smallest amount possible.

Table 11-12. DCO frequency range[1]

DRS	DMX32	Reference range	FLL factor	DCO range
00	**0**	31.25 - 39.0625 kHz	512	16 - 20 MHz
	1	32.768 kHz	608	19.92 MHz
01	**0**	31.25 - 39.0625 kHz	1024	32 - 40 MHz
	1	32.768 kHz	1216	39.85 MHz
10	**0**	31.25 - 39.0625 kHz	1536	48 - 60 MHz
	1	32.768 kHz	1824	59.77 MHz
11	Reserved			

[1] The resulting bus clock frequency should not exceed the maximum specified bus clock frequency of the device.

11.4 Functional Description

11.4.1 Operational Modes

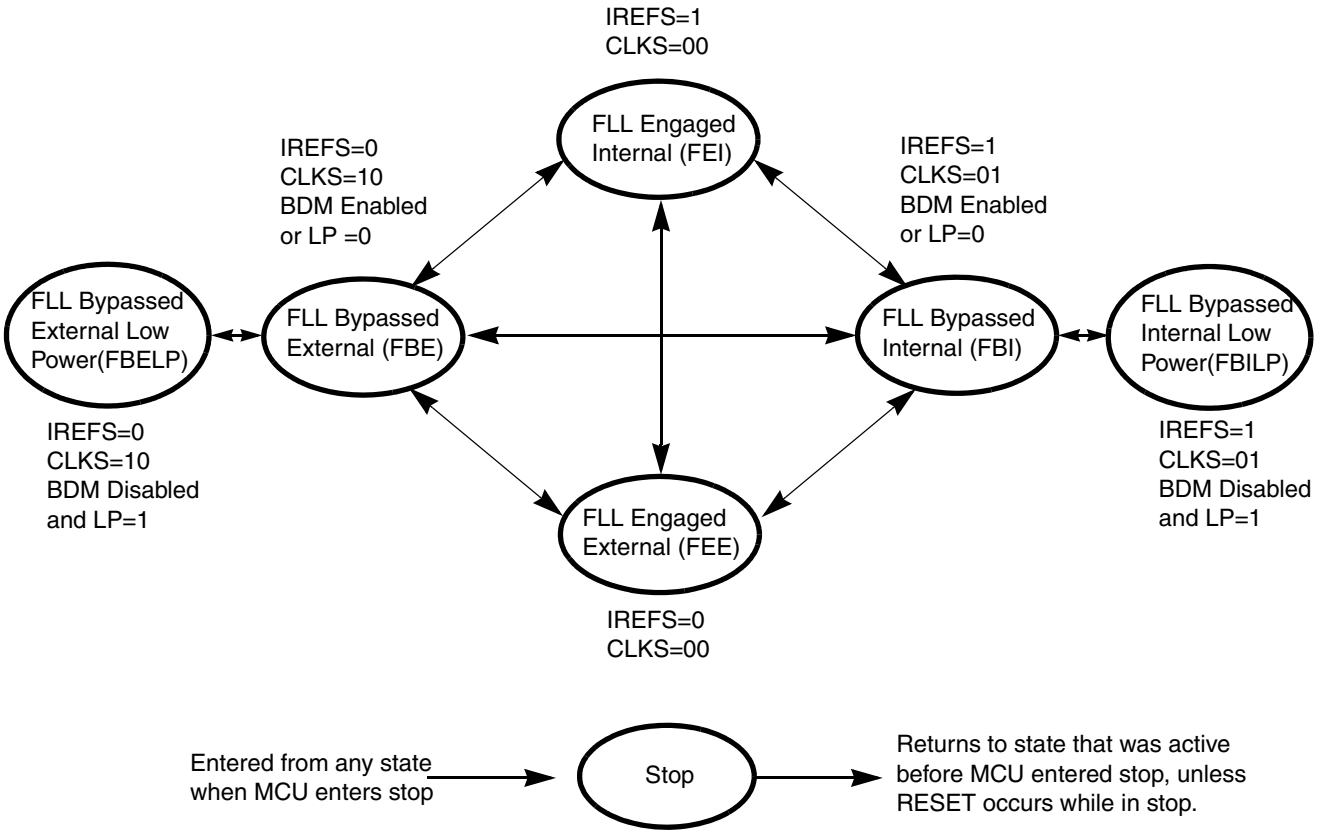


Figure 11-11. Clock Switching Modes

The seven states of the ICS are shown as a state diagram and are described below. The arrows indicate the allowed movements between the states.

11.4.1.1 FLL Engaged Internal (FEI)

FLL engaged internal (FEI) is the default mode of operation and is entered when all the following conditions occur:

- CLKS bits are written to 00.
- IREFS bit is written to 1.

In FLL engaged internal mode, the ICSOUT clock is derived from the FLL clock, which is controlled by the internal reference clock. The FLL loop locks the frequency to the FLL factor times the internal reference frequency. The ICSLCLK is available for BDC communications, and the internal reference clock is enabled.

11.4.1.2 FLL Engaged External (FEE)

The FLL engaged external (FEE) mode is entered when all the following conditions occur:

- CLKS bits are written to 00.
- IREFS bit is written to 0.
- RDIV bits are written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz.

In FLL engaged external mode, the ICSOUT clock is derived from the FLL clock which is controlled by the external reference clock source.The FLL loop locks the frequency to the FLL factor times the external reference frequency, as selected by the RDIV bits. The ICSLCLK is available for BDC communications, and the external reference clock is enabled.

11.4.1.3 FLL Bypassed Internal (FBI)

The FLL bypassed internal (FBI) mode is entered when all the following conditions occur:

- CLKS bits are written to 01.
- IREFS bit is written to 1.
- BDM mode is active or LP bit is written to 0.

In FLL bypassed internal mode, the ICSOUT clock is derived from the internal reference clock. The FLL clock is controlled by the internal reference clock, and the FLL loop locks the FLL frequency to the FLL factor times the internal reference frequency. The ICSLCLK will be available for BDC communications, and the internal reference clock is enabled.

11.4.1.4 FLL Bypassed Internal Low Power (FBILP)

The FLL bypassed internal low power (FBILP) mode is entered when all the following conditions occur:

- CLKS bits are written to 01.
- IREFS bit is written to 1.
- BDM mode is not active and LP bit is written to 1.

In FLL bypassed internal low power mode, the ICSOUT clock is derived from the internal reference clock and the FLL is disabled. The ICSLCLK will be not be available for BDC communications, and the internal reference clock is enabled.

11.4.1.5 FLL Bypassed External (FBE)

The FLL bypassed external (FBE) mode is entered when all the following conditions occur:

- CLKS bits are written to 10.
- IREFS bit is written to 0.
- RDIV bits are written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz.
- BDM mode is active or LP bit is written to 0.

In FLL bypassed external mode, the ICSOUT clock is derived from the external reference clock source. The FLL clock is controlled by the external reference clock, and the FLL loop locks the FLL frequency to the FLL factor times the external reference frequency, as selected by the RDIV bits, so that the ICSLCLK will be available for BDC communications, and the external reference clock is enabled.

11.4.1.6 FLL Bypassed External Low Power (FBELP)

The FLL bypassed external low power (FBELP) mode is entered when all the following conditions occur:

- CLKS bits are written to 10.
- IREFS bit is written to 0.
- BDM mode is not active and LP bit is written to 1.

In FLL bypassed external low power mode, the ICSOUT clock is derived from the external reference clock source and the FLL is disabled. The ICSLCLK will be not be available for BDC communications. The external reference clock source is enabled.

11.4.1.7 Stop

Stop mode is entered whenever the MCU enters a STOP state. In this mode, all ICS clock signals are static except in the following cases:

ICSIRCLK will be active in stop mode when all the following conditions occur:

- IRCLKEN bit is written to 1.
- IREFSTEN bit is written to 1.

OSCOUT will be active in stop mode when all the following conditions occur:

- ERCLKEN bit is written to 1.
- EREFSTEN bit is written to 1.

11.4.2 Mode Switching

The IREF bit can be changed at anytime, but the actual switch to the newly selected clock is shown by the IREFST bit. When switching between FLL engaged internal (FEI) and FLL engaged external (FEE) modes, the FLL begins locking again after the switch is completed.

The CLKS bits can also be changed at anytime, but the actual switch to the newly selected clock is shown by the CLKST bits. If the newly selected clock is not available, the previous clock remains selected.

The DRS bits can be changed at anytime except when LP bit is 1. If the DRS bits are changed while in FLL engaged internal (FEI) or FLL engaged external (FEE), the bus clock remains at the previous DCO range until the new DCO starts. When the new DCO starts the bus clock switches to it. After switching to the new DCO the FLL remains unlocked for several reference cycles. Once the selected DCO startup time is over, the FLL is locked. The completion of the switch is shown by the DRST bits.

11.4.3 Bus Frequency Divider

The BDIV bits can be changed at anytime and the actual switch to the new frequency occurs immediately.

11.4.4 Low Power Bit Usage

The low power bit (LP) is provided to allow the FLL to be disabled and thus conserve power when it is not being used. The DRS bits can not be written while LP bit is 1.

However, in some applications it may be desirable to allow the FLL to be enabled and to lock for maximum accuracy before switching to an FLL engaged mode. To do this, write the LP bit to 0.

11.4.5 DCO Maximum Frequency with 32.768 kHz Oscillator

The FLL has an option to change the clock multiplier for the selected DCO range such that it results in the maximum bus frequency with a common 32.768 kHz crystal reference clock.

11.4.6 Internal Reference Clock

When IRCLKEN is set the internal reference clock signal is presented as ICSIRCLK, which can be used as an additional clock source. To re-target the ICSIRCLK frequency, write a new value to the TRIM bits in the ICSTRM register to trim the period of the internal reference clock:

- Writing a larger value slows down the ICSIRCLK frequency.
- Writing a smaller value to the ICSTRM register speeds up the ICSIRCLK frequency.

The TRIM bits effect the ICSOUT frequency if the ICS is in FLL engaged internal (FEI), FLL bypassed internal (FBI), or FLL bypassed internal low power (FBILP) mode.

Until ICSIRCLK is trimmed, programming low reference divider (RDIV) factors may result in ICSOUT frequencies that exceed the maximum chip-level frequency and violate the chip-level clock timing specifications (see the Device Overview chapter).

If IREFSTEN is set and the IRCLKEN bit is written to 1, the internal reference clock keeps running during stop mode in order to provide a fast recovery upon exiting stop.

All MCU devices are factory programmed with a trim value in a reserved memory location. This value is uploaded to the ICSTRM register and ICS FTRIM register during any reset initialization. For finer precision, trim the internal oscillator in the application and set the FTRIM bit accordingly.

11.4.7 External Reference Clock

The ICS module supports an external reference clock with frequencies between 31.25 kHz to 40 MHz in all modes. When the ERCLKEN is set, the external reference clock signal is presented as ICSERCLK, which can be used as an additional clock source in run mode. When IREFS = 1, the external reference clock is not used by the FLL and will only be used as ICSERCLK. In these modes, the frequency can be equal to the maximum frequency the chip-level timing specifications support (see the Device Overview chapter).

If EREFSTEN is set and the ERCLKEN bit is written to 1, the external reference clock source (OSCOUT) keeps running during stop mode in order to provide a fast recovery upon exiting stop.

11.4.8 Fixed Frequency Clock

The ICS presents the divided FLL reference clock as ICSFFCLK for use as an additional clock source. ICSFFCLK frequency must be no more than 1/4 of the ICSOUT frequency to be valid.

11.4.9 Local Clock

The ICS presents the low range DCO output clock divided by two as ICSLCLK for use as a clock source for BDC communications. ICSLCLK is not available in FLL bypassed internal low power (FBILP) and FLL bypassed external low power (FBELP) modes.

Chapter 12
8-Bit Serial Peripheral Interface (SPI)

12.1 Introduction

The SPI2 and SPI3 in MCF51EM256 series MCUs are an 8-bit serial peripheral interface (SPI) module.

NOTE

Ignore any references to stop1 low-power mode in this chapter, because this device does not support it. For details on low-power mode operation, refer to Table 6-4 in Chapter 6, “Modes of Operation.”

12.1.1 Features

Features of the SPI module include:

- Master or slave mode operation
- Full-duplex or single-wire bidirectional option
- Programmable transmit bit rate
- Double-buffered transmit and receive
- Serial clock phase and polarity options
- Slave select output
- Selectable MSB-first or LSB-first shifting

12.1.2 Block Diagrams

This section includes block diagrams showing SPI system connections, the internal organization of the SPI module, and the SPI clock dividers that control the master mode bit rate.

12.1.2.1 SPI System Block Diagram

Figure 12-1 shows the SPI modules of two MCUs connected in a master-slave arrangement. The master device initiates all SPI data transfers. During a transfer, the master shifts data out (on the MOSI pin) to the slave while simultaneously shifting data in (on the MISO pin) from the slave. The transfer effectively exchanges the data that was in the SPI shift registers of the two SPI systems. The SPSCK signal is a clock output from the master and an input to the slave. The slave device must be selected by a low level on the slave select input ($\overline{SS}$ pin). In this system, the master device has configured its $\overline{SS}$ pin as an optional slave select output.

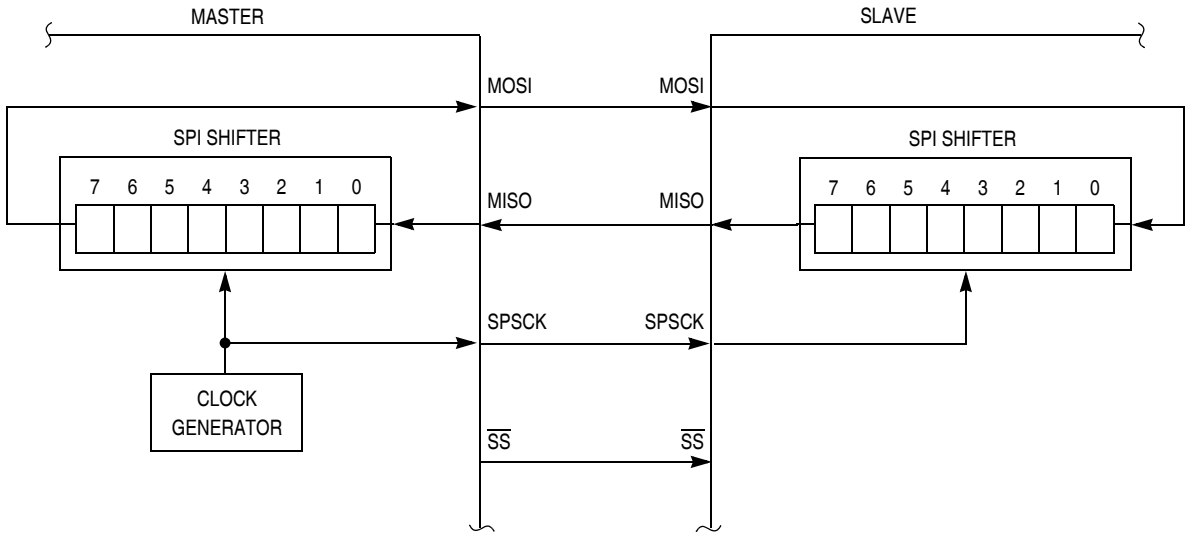


Figure 12-1. SPI System Connections

The most common uses of the SPI system include connecting simple shift registers for adding input or output ports or connecting small peripheral devices such as serial A/D or D/A converters. Although Figure 12-1 shows a system where data is exchanged between two MCUs, many practical systems involve simpler connections where data is unidirectionally transferred from the master MCU to a slave or from a slave to the master MCU.

12.1.2.2 SPI Module Block Diagram

Figure 12-2 is a block diagram of the SPI module. The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter (write to SPIxD) and gets transferred to the SPI shift register at the start of a data transfer. After shifting in a byte of data, the data is transferred into the double-buffered receiver where it can be read (read from SPIxD). Pin multiplexing logic controls connections between MCU pins and the SPI module.

When the SPI is configured as a master, the clock output is routed to the SPSCK pin, the shifter output is routed to MOSI, and the shifter input is routed from the MISO pin.

When the SPI is configured as a slave, the SPSCK pin is routed to the clock input of the SPI, the shifter output is routed to MISO, and the shifter input is routed from the MOSI pin.

In the external SPI system, simply connect all SPSCK pins to each other, all MISO pins together, and all MOSI pins together. Peripheral devices often use slightly different names for these pins.

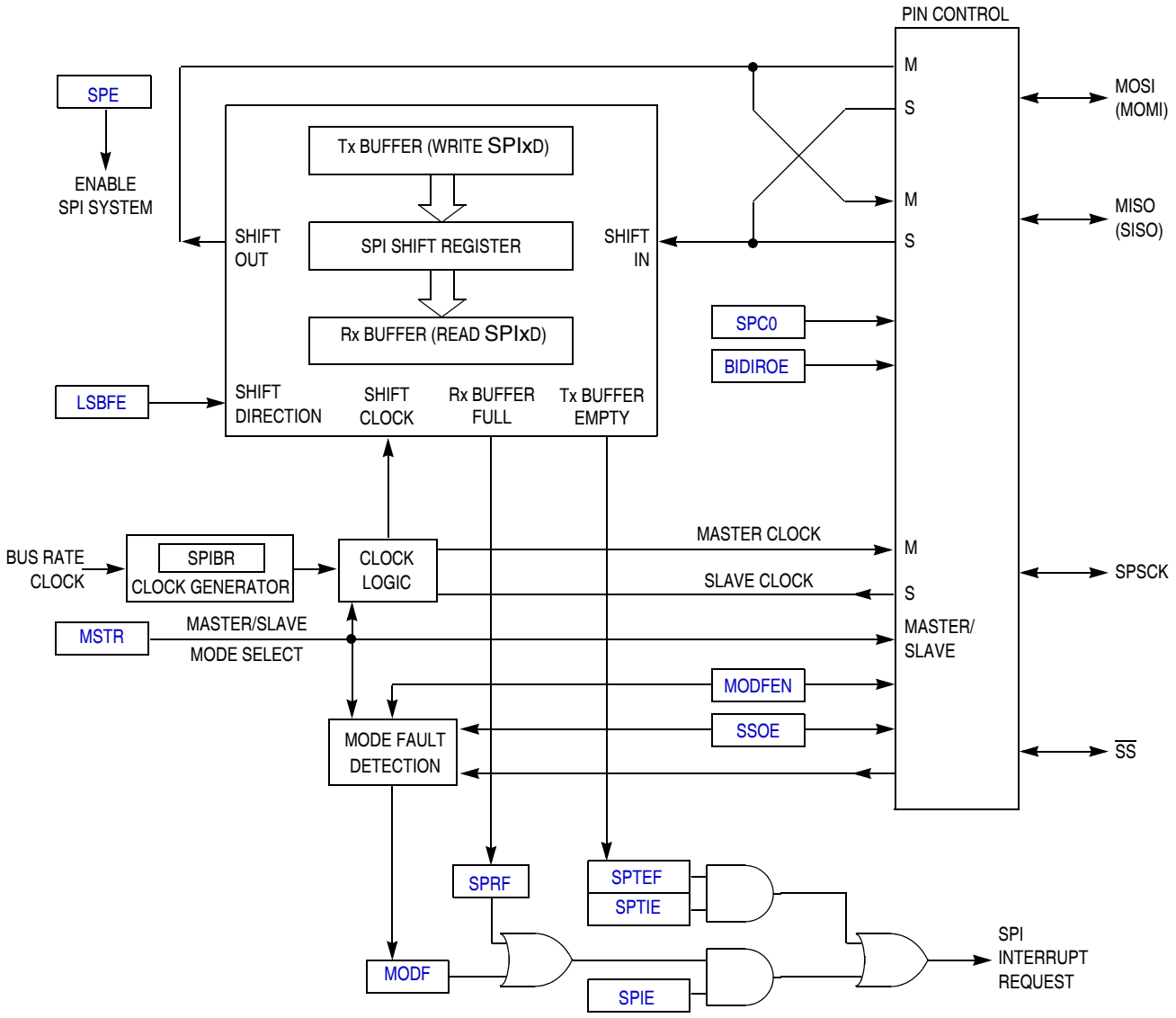


Figure 12-2. SPI Module Block Diagram

12.1.3 SPI Baud Rate Generation

As shown in Figure 12-3, the clock source for the SPI baud rate generator is the bus clock. The three prescale bits (SPPR2:SPPR1:SPPR0) choose a prescale divisor of 1, 2, 3, 4, 5, 6, 7, or 8. The three rate select bits (SPR3:SPR2:SPR1:SPR0) divide the output of the prescaler stage by 2, 4, 8, 16, 32, 64, 128, 256,or 512 to get the internal SPI master mode bit-rate clock.

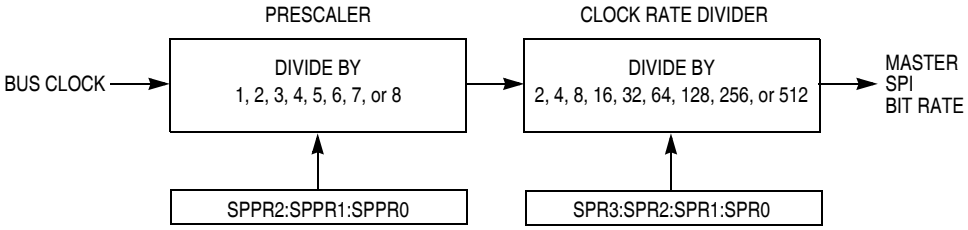


Figure 12-3. SPI Baud Rate Generation

12.2 External Signal Description

The SPI optionally shares four port pins. The function of these pins depends on the settings of SPI control bits. When the SPI is disabled (SPE = 0), these four pins revert to being general-purpose port I/O pins that are not controlled by the SPI.

12.2.1 SPSCK — SPI Serial Clock

When the SPI is enabled as a slave, this pin is the serial clock input. When the SPI is enabled as a master, this pin is the serial clock output.

12.2.2 MOSI — Master Data Out, Slave Data In

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data output. When the SPI is enabled as a slave and SPC0 = 0, this pin is the serial data input. If SPC0 = 1 to select single-wire bidirectional mode, and master mode is selected, this pin becomes the bidirectional data I/O pin (MOMI). Also, the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE = 0) or an output (BIDIROE = 1). If SPC0 = 1 and slave mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

12.2.3 MISO — Master Data In, Slave Data Out

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data input. When the SPI is enabled as a slave and SPC0 = 0, this pin is the serial data output. If SPC0 = 1 to select single-wire bidirectional mode, and slave mode is selected, this pin becomes the bidirectional data I/O pin (SISO) and the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE = 0) or an output (BIDIROE = 1). If SPC0 = 1 and master mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

12.2.4 $\overline{SS}$ — Slave Select

When the SPI is enabled as a slave, this pin is the low-true slave select input. When the SPI is enabled as a master and mode fault enable is off (MODFEN = 0), this pin is not used by the SPI and reverts to being a general-purpose port I/O pin. When the SPI is enabled as a master and MODFEN = 1, the slave select output enable bit determines whether this pin acts as the mode fault input (SSOE = 0) or as the slave select output (SSOE = 1).

12.3 Modes of Operation

12.3.1 SPI in Stop Modes

The SPI is disabled in all stop modes, regardless of the settings before executing the STOP instruction. During either stop1 or stop2 mode, the SPI module will be fully powered down. Upon wake-up from stop1 or stop2 mode, the SPI module will be in the reset state. During stop3 mode, clocks to the SPI module are halted. No registers are affected. If stop3 is exited with a reset, the SPI will be put into its reset state. If stop3 is exited with an interrupt, the SPI continues from the state it was in when stop3 was entered.

12.4 Register Definition

The SPI has five 8-bit registers to select SPI options, control baud rate, report SPI status, and for transmit/receive data.

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all SPI registers. This section refers to registers and control bits only by their names, and a Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

12.4.1 SPI Control Register 1 (SPIxC1)

This read/write register includes the SPI enable control, interrupt enables, and configuration options.

	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 12-4. SPI Control Register 1 (SPIxC1)

Table 12-1. SPIxC1 Field Descriptions

Field	Description
7 SPIE	**SPI Interrupt Enable (for SPRF and MODF)** — This is the interrupt enable for SPI receive buffer full (SPRF) and mode fault (MODF) events. 0 Interrupts from SPRF and MODF inhibited (use polling) 1 When SPRF or MODF is 1, request a hardware interrupt
6 SPE	**SPI System Enable** — Disabling the SPI halts any transfer that is in progress, clears data buffers, and initializes internal state machines. SPRF is cleared and SPTEF is set to indicate the SPI transmit data buffer is empty. 0 SPI system inactive 1 SPI system enabled
5 SPTIE	**SPI Transmit Interrupt Enable** — This is the interrupt enable bit for SPI transmit buffer empty (SPTEF). 0 Interrupts from SPTEF inhibited (use polling) 1 When SPTEF is 1, hardware interrupt requested

Table 12-1. SPIxC1 Field Descriptions (continued)

Field	Description
4 MSTR	**Master/Slave Mode Select** 0 SPI module configured as a slave SPI device 1 SPI module configured as a master SPI device
3 CPOL	**Clock Polarity** — This bit effectively places an inverter in series with the clock signal from a master SPI or to a slave SPI device. Refer to Section 12.5.1, "SPI Clock Formats" for more details. 0 Active-high SPI clock (idles low) 1 Active-low SPI clock (idles high)
2 CPHA	**Clock Phase** — This bit selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to Section 12.5.1, "SPI Clock Formats" for more details. 0 First edge on SPSCK occurs at the middle of the first cycle of an 8-cycle data transfer 1 First edge on SPSCK occurs at the start of the first cycle of an 8-cycle data transfer
1 SSOE	**Slave Select Output Enable** — This bit is used in combination with the mode fault enable (MODFEN) bit in SPCR2 and the master/slave (MSTR) control bit to determine the function of the $\overline{SS}$ pin as shown in Table 12-2.
0 LSBFE	**LSB First (Shifter Direction)** 0 SPI serial data transfers start with most significant bit 1 SPI serial data transfers start with least significant bit

Table 12-2. $\overline{SS}$ Pin Function

MODFEN	SSOE	Master Mode	Slave Mode
0	0	General-purpose I/O (not SPI)	Slave select input
0	1	General-purpose I/O (not SPI)	Slave select input
1	0	$\overline{SS}$ input for mode fault	Slave select input
1	1	Automatic $\overline{SS}$ output	Slave select input

NOTE

Ensure that the SPI should not be disabled (SPE=0) at the same time as a bit change to the CPHA bit. These changes should be performed as separate operations or unexpected behavior may occur.

12.4.2 SPI Control Register 2 (SPIxC2)

This read/write register is used to control optional features of the SPI system. Bits 7, 6, 5, and 2 are not implemented and always read 0.

	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 12-5. SPI Control Register 2 (SPIxC2)

Table 12-3. SPIxC2 Register Field Descriptions

Field	Description
4 MODFEN	**Master Mode-Fault Function Enable** — When the SPI is configured for slave mode, this bit has no meaning or effect. (The $\overline{SS}$ pin is the slave select input.) In master mode, this bit determines how the $\overline{SS}$ pin is used (refer to Table 12-2 for more details). 0 Mode fault function disabled, master $\overline{SS}$ pin reverts to general-purpose I/O not controlled by SPI 1 Mode fault function enabled, master $\overline{SS}$ pin acts as the mode fault input or the slave select output
3 BIDIROE	**Bidirectional Mode Output Enable** — When bidirectional mode is enabled by SPI pin control 0 (SPC0) = 1, BIDIROE determines whether the SPI data output driver is enabled to the single bidirectional SPI I/O pin. Depending on whether the SPI is configured as a master or a slave, it uses either the MOSI (MOMI) or MISO (SISO) pin, respectively, as the single SPI data I/O pin. When SPC0 = 0, BIDIROE has no meaning or effect. 0 Output driver disabled so SPI data I/O pin acts as an input 1 SPI I/O pin enabled as an output
1 SPISWAI	**SPI Stop in Wait Mode** 0 SPI clocks continue to operate in wait mode 1 SPI clocks stop when the MCU enters wait mode
0 SPC0	**SPI Pin Control 0** — The SPC0 bit chooses single-wire bidirectional mode. If MSTR = 0 (slave mode), the SPI uses the MISO (SISO) pin for bidirectional SPI data transfers. If MSTR = 1 (master mode), the SPI uses the MOSI (MOMI) pin for bidirectional SPI data transfers. When SPC0 = 1, BIDIROE is used to enable or disable the output driver for the single bidirectional SPI I/O pin. 0 SPI uses separate pins for data input and data output 1 SPI configured for single-wire bidirectional operation

12.4.3 SPI Baud Rate Register (SPIxBR)

This register is used to set the prescaler and bit rate divisor for an SPI master. This register may be read or written at any time.

	7	6	5	4	3	2	1	0
R	0	SPPR2	SPPR1	SPPR0	SPR3	SPR2	SPR1	SPR0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 12-6. SPI Baud Rate Register (SPIxBR)

Table 12-4. SPIxBR Register Field Descriptions

Field	Description
6:4 SPPR[2:0]	**SPI Baud Rate Prescale Divisor** — This 3-bit field selects one of eight divisors for the SPI baud rate prescaler as shown in Table 12-5. The input to this prescaler is the bus rate clock (BUSCLK). The output of this prescaler drives the input of the SPI baud rate divider (see Figure 12-3).
2:0 SPR[3:0]	**SPI Baud Rate Divisor** — This 4-bit field selects one of eight divisors for the SPI baud rate divider as shown in Table 12-6. The input to this divider comes from the SPI baud rate prescaler (see Figure 12-3). The output of this divider is the SPI bit rate clock for master mode.

Table 12-5. SPI Baud Rate Prescaler Divisor

SPPR2:SPPR1:SPPR0	Prescaler Divisor
0:0:0	1
0:0:1	2
0:1:0	3
0:1:1	4
1:0:0	5
1:0:1	6
1:1:0	7
1:1:1	8

Table 12-6. SPI Baud Rate Divisor

SPR3:SPR2:SPR1:SPR0	Rate Divisor
0:0:0:0	2
0:0:0:1	4
0:0:1:0	8
0:0:1:1	16
0:1:0:0	32
0:1:0:1	64
0:1:1:0	128
0:1:1:1	256
1:0:0:0	512
All other combinations	reserved

12.4.4 SPI Status Register (SPIxS)

This register has three read-only status bits. Bits 6, 3, 2, 1, and 0 are not implemented and always read 0. Writes have no meaning or effect.

	7	6	5	4	3	2	1	0
R	SPRF	0	SPTEF	MODF	0	0	0	0
W								
Reset	0	0	1	0	0	0	0	0

= Unimplemented or Reserved

Figure 12-7. SPI Status Register (SPIxS)

Table 12-7. SPIxS Register Field Descriptions

Field	Description
7 SPRF	**SPI Read Buffer Full Flag** — SPRF is set at the completion of an SPI transfer to indicate that received data may be read from the SPI data register (SPIxD). SPRF is cleared by reading SPRF while it is set, then reading the SPI data register. 0 No data available in the receive data buffer 1 Data available in the receive data buffer
5 SPTEF	**SPI Transmit Buffer Empty Flag** — This bit is set when there is room in the transmit data buffer. It is cleared by reading SPIxS with SPTEF set, followed by writing a data value to the transmit buffer at SPIxD. SPIxS must be read with SPTEF = 1 before writing data to SPIxD or the SPIxD write will be ignored. SPTEF generates an SPTEF CPU interrupt request if the SPTIE bit in the SPIxC1 is also set. SPTEF is automatically set when a data byte transfers from the transmit buffer into the transmit shift register. For an idle SPI (no data in the transmit buffer or the shift register and no transfer in progress), data written to SPIxD is transferred to the shifter almost immediately so SPTEF is set within two bus cycles allowing a second 8-bit data value to be queued into the transmit buffer. After completion of the transfer of the value in the shift register, the queued value from the transmit buffer will automatically move to the shifter and SPTEF will be set to indicate there is room for new data in the transmit buffer. If no new data is waiting in the transmit buffer, SPTEF simply remains set and no data moves from the buffer to the shifter. 0 SPI transmit buffer not empty 1 SPI transmit buffer empty
4 MODF	**Master Mode Fault Flag** — MODF is set if the SPI is configured as a master and the slave select input goes low, indicating some other SPI device is also configured as a master. The $\overline{SS}$ pin acts as a mode fault error input only when MSTR = 1, MODFEN = 1, and SSOE = 0; otherwise, MODF will never be set. MODF is cleared by reading MODF while it is 1, then writing to SPI control register 1 (SPIxC1). 0 No mode fault error 1 Mode fault error detected

12.4.5 SPI Data Register (SPIxD)

	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 12-8. SPI Data Register (SPIxD)

Reads of this register return the data read from the receive data buffer. Writes to this register write data to the transmit data buffer. When the SPI is configured as a master, writing data to the transmit data buffer initiates an SPI transfer.

Data should not be written to the transmit data buffer unless the SPI transmit buffer empty flag (SPTEF) is set, indicating there is room in the transmit buffer to queue a new transmit byte.

Data may be read from SPIxD any time after SPRF is set and before another transfer is finished. Failure to read the data out of the receive data buffer before a new transfer ends causes a receive overrun condition and the data from the new transfer is lost.

12.5 Functional Description

An SPI transfer is initiated by checking for the SPI transmit buffer empty flag (SPTEF = 1) and then writing a byte of data to the SPI data register (SPIxD) in the master SPI device. When the SPI shift register is available, this byte of data is moved from the transmit data buffer to the shifter, SPTEF is set to indicate there is room in the buffer to queue another transmit character if desired, and the SPI serial transfer starts.

During the SPI transfer, data is sampled (read) on the MISO pin at one SPSCK edge and shifted, changing the bit value on the MOSI pin, one-half SPSCK cycle later. After eight SPSCK cycles, the data that was in the shift register of the master has been shifted out the MOSI pin to the slave while eight bits of data were shifted in the MISO pin into the master's shift register. At the end of this transfer, the received data byte is moved from the shifter into the receive data buffer and SPRF is set to indicate the data can be read by reading SPIxD. If another byte of data is waiting in the transmit buffer at the end of a transfer, it is moved into the shifter, SPTEF is set, and a new transfer is started.

Normally, SPI data is transferred most significant bit (MSB) first. If the least significant bit first enable (LSBFE) bit is set, SPI data is shifted LSB first.

When the SPI is configured as a slave, its $\overline{SS}$ pin must be driven low before a transfer starts and $\overline{SS}$ must stay low throughout the transfer. If a clock format where CPHA = 0 is selected, $\overline{SS}$ must be driven to a logic 1 between successive transfers. If CPHA = 1, $\overline{SS}$ may remain low between successive transfers. See Section 12.5.1, "SPI Clock Formats" for more details.

Because the transmitter and receiver are double buffered, a second byte, in addition to the byte currently being shifted out, can be queued into the transmit data buffer, and a previously received character can be in the receive data buffer while a new character is being shifted in. The SPTEF flag indicates when the transmit buffer has room for a new character. The SPRF flag indicates when a received character is available in the receive data buffer. The received character must be read out of the receive buffer (read SPIxD) before the next transfer is finished or a receive overrun error results.

In the case of a receive overrun, the new data is lost because the receive buffer still held the previous character and was not ready to accept the new data. There is no indication for such an overrun condition so the application system designer must ensure that previous data has been read from the receive buffer before a new transfer is initiated.

12.5.1 SPI Clock Formats

To accommodate a wide variety of synchronous serial peripherals from different manufacturers, the SPI system has a clock polarity (CPOL) bit and a clock phase (CPHA) control bit to select one of four clock formats for data transfers. CPOL selectively inserts an inverter in series with the clock. CPHA chooses between two different clock phase relationships between the clock and data.

Figure 12-9 shows the clock formats when CPHA = 1. At the top of the figure, the eight bit times are shown for reference with bit 1 starting at the first SPSCK edge and bit 8 ending one-half SPSCK cycle after the sixteenth SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output

pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{SS}$ OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master $\overline{SS}$ output goes to active low one-half SPSCK cycle before the start of the transfer and goes back high at the end of the eighth bit time of the transfer. The $\overline{SS}$ IN waveform applies to the slave select input of a slave.

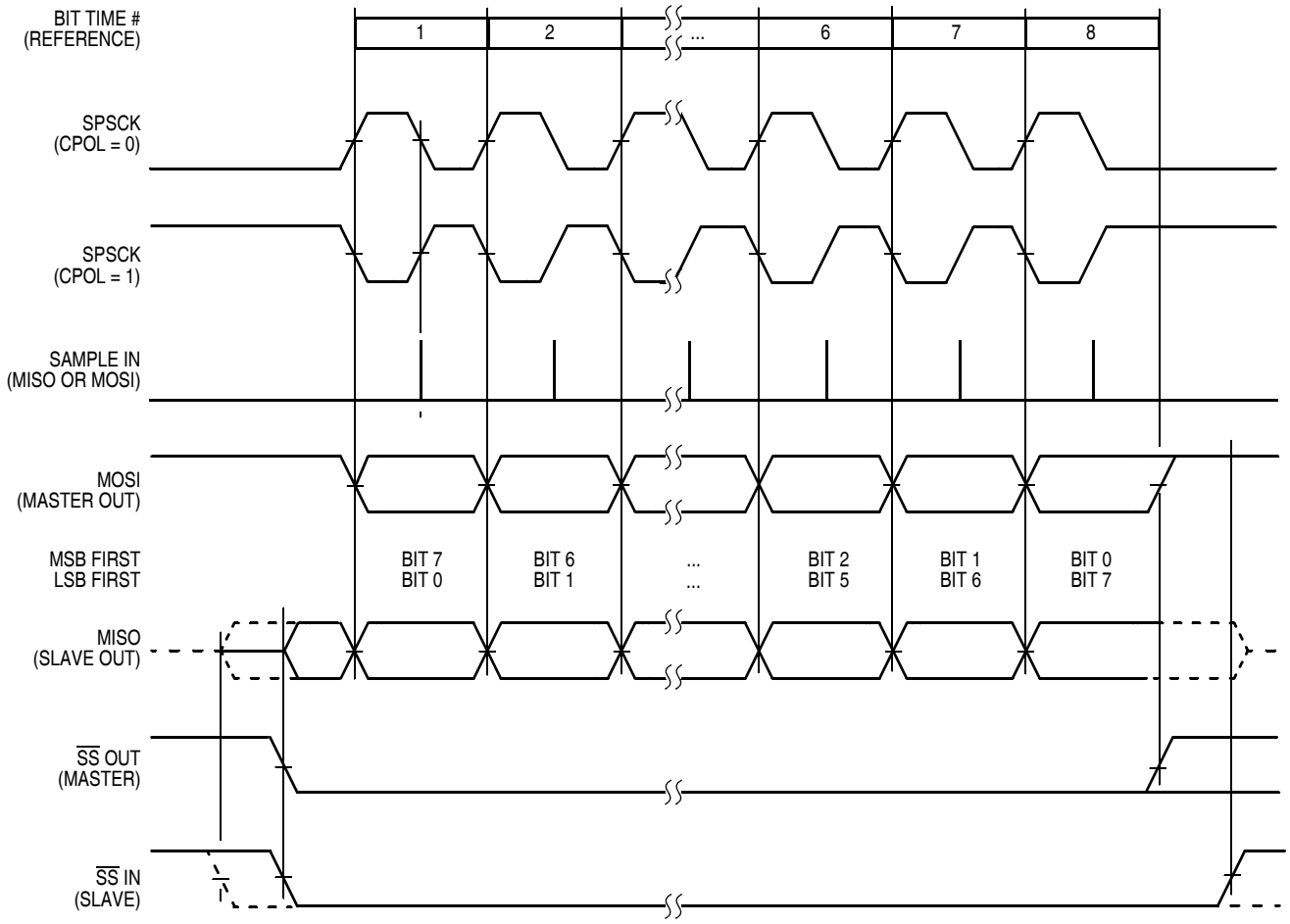


Figure 12-9. SPI Clock Formats (CPHA = 1)

When CPHA = 1, the slave begins to drive its MISO output when $\overline{SS}$ goes to active low, but the data is not defined until the first SPSCK edge. The first SPSCK edge shifts the first bit of data from the shifter onto the MOSI output of the master and the MISO output of the slave. The next SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled, and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CHPA = 1, the slave's $\overline{SS}$ input is not required to go to its inactive high level between transfers.

Figure 12-10 shows the clock formats when CPHA = 0. At the top of the figure, the eight bit times are shown for reference with bit 1 starting as the slave is selected ($\overline{SS}$ IN goes low), and bit 8 ends at the last SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting

in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{SS}$ OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master $\overline{SS}$ output goes to active low at the start of the first bit time of the transfer and goes back high one-half SPSCK cycle after the end of the eighth bit time of the transfer. The $\overline{SS}$ IN waveform applies to the slave select input of a slave.

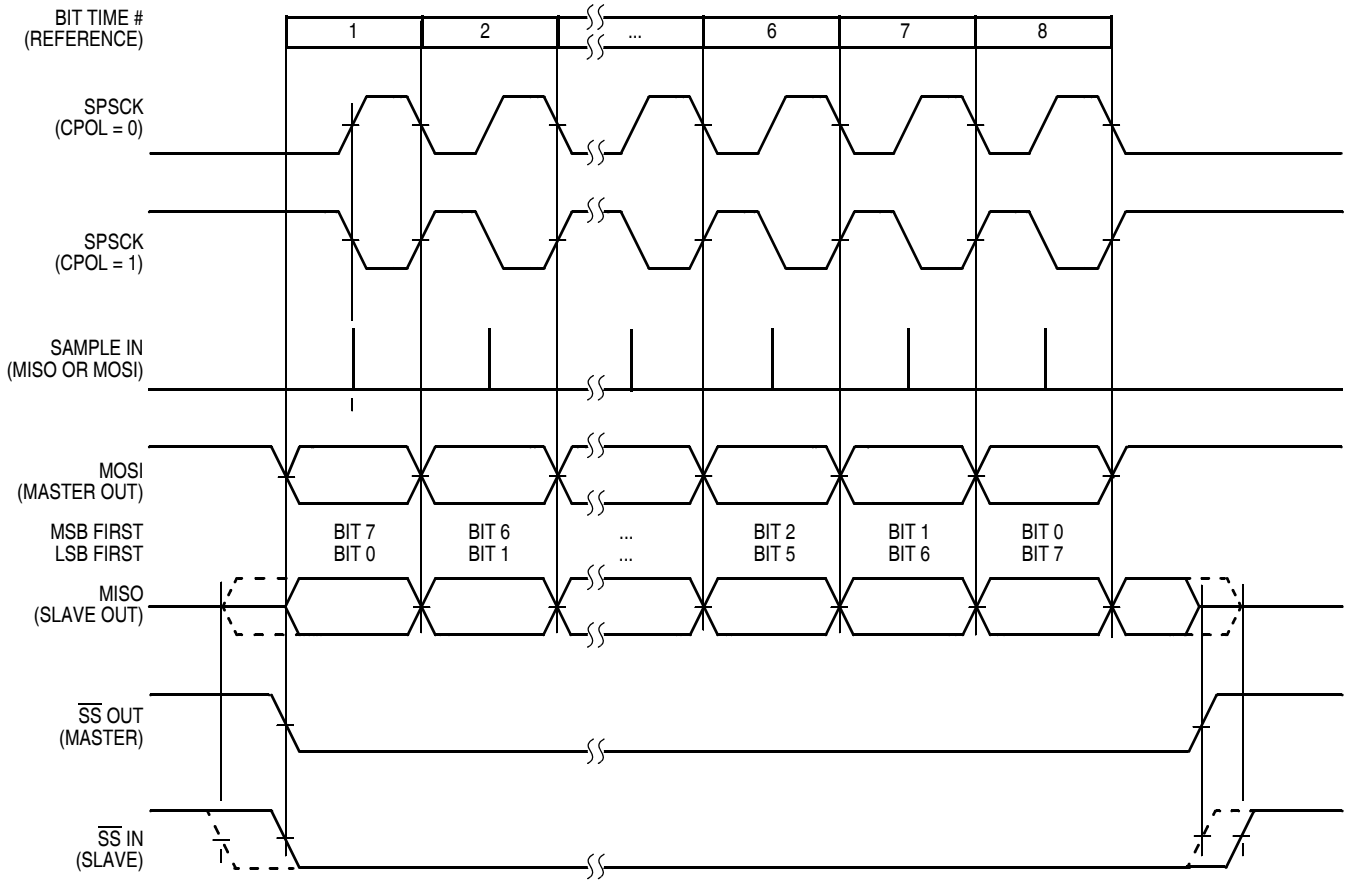


Figure 12-10. SPI Clock Formats (CPHA = 0)

When CPHA = 0, the slave begins to drive its MISO output with the first data bit value (MSB or LSB depending on LSBFE) when $\overline{SS}$ goes to active low. The first SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the second SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CPHA = 0, the slave's $\overline{SS}$ input must go to its inactive high level between transfers.

12.5.2 SPI Interrupts

There are three flag bits, two interrupt mask bits, and one interrupt vector associated with the SPI system. The SPI interrupt enable mask (SPIE) enables interrupts from the SPI receiver full flag (SPRF) and mode fault flag (MODF). The SPI transmit interrupt enable mask (SPTIE) enables interrupts from the SPI transmit buffer empty flag (SPTEF). When one of the flag bits is set, and the associated interrupt mask bit is set, a hardware interrupt request is sent to the CPU. If the interrupt mask bits are cleared, software can poll the associated flag bits instead of using interrupts. The SPI interrupt service routine (ISR) should check the flag bits to determine what event caused the interrupt. The service routine should also clear the flag bit(s) before returning from the ISR (usually near the beginning of the ISR).

12.5.3 Mode Fault Detection

A mode fault occurs and the mode fault flag (MODF) becomes set when a master SPI device detects an error on the $\overline{SS}$ pin (provided the $\overline{SS}$ pin is configured as the mode fault input signal). The $\overline{SS}$ pin is configured to be the mode fault input signal when MSTR = 1, mode fault enable is set (MODFEN = 1), and slave select output enable is clear (SSOE = 0).

The mode fault detection feature can be used in a system where more than one SPI device might become a master at the same time. The error is detected when a master's $\overline{SS}$ pin is low, indicating that some other SPI device is trying to address this master as if it were a slave. This could indicate a harmful output driver conflict, so the mode fault logic is designed to disable all SPI output drivers when such an error is detected.

When a mode fault is detected, MODF is set and MSTR is cleared to change the SPI configuration back to slave mode. The output drivers on the SPSCK, MOSI, and MISO (if not bidirectional mode) are disabled.

MODF is cleared by reading it while it is set, then writing to the SPI control register 1 (SPIxC1). User software should verify the error condition has been corrected before changing the SPI back to master mode.

Chapter 13
8- or 16-Bit Serial Peripheral Interface (SPI16)

13.1 Introduction

The SPI1 in MCF51EM256 series MCUs is a 16-bit serial peripheral interface (SPI) module with FIFO which is capable of being configured for 8- or 16-bit data.

13.1.1 Features

The SPI includes these distinctive features:

- Master mode or slave mode operation
- Full-duplex or single-wire bidirectional mode
- Programmable transmit bit rate
- Double-buffered transmit and receive data register
- Serial clock phase and polarity options
- Slave select output
- Mode fault error flag with CPU interrupt capability
- Control of SPI operation during wait mode
- Selectable MSB-first or LSB-first shifting
- Programmable 8- or 16-bit data transmission length
- Receive data buffer hardware match feature
- 64bit FIFO mode for high speed/large amounts of data transfers.

13.1.2 Modes of Operation

The SPI functions in three modes, run, wait, and stop.

- Run Mode

  This is the basic mode of operation.
- Wait Mode

  SPI operation in wait mode is a configurable low power mode, controlled by the SPISWAI bit located in the SPIxC2 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 stop3/stop4 mode for reduced power consumption. If the SPI is configured as a master, any transmission in progress stops, but is resumed after the CPU goes into Run Mode. If the SPI is configured as a slave, reception and transmission of a data continues, so that the slave stays synchronized to the master.

The SPI is completely disabled in all other stop modes. When the CPU wakes from these stop modes, all SPI register content will be reset.

This is a high level description only, detailed descriptions of operating modes are contained in section Section 13.4.10, “Low Power Mode Options.”

13.1.3 Block Diagrams

This section includes block diagrams showing SPI system connections, the internal organization of the SPI module, and the SPI clock dividers that control the master mode bit rate.

13.1.3.1 SPI System Block Diagram

Figure 13-1 shows the SPI modules of two MCUs connected in a master-slave arrangement. The master device initiates all SPI data transfers. During a transfer, the master shifts data out (on the MOSI pin) to the slave while simultaneously shifting data in (on the MISO pin) from the slave. The transfer effectively exchanges the data that was in the SPI shift registers of the two SPI systems. The SPSCK signal is a clock output from the master and an input to the slave. The slave device must be selected by a low level on the slave select input ($\overline{SS}$ pin). In this system, the master device has configured its $\overline{SS}$ pin as an optional slave select output.

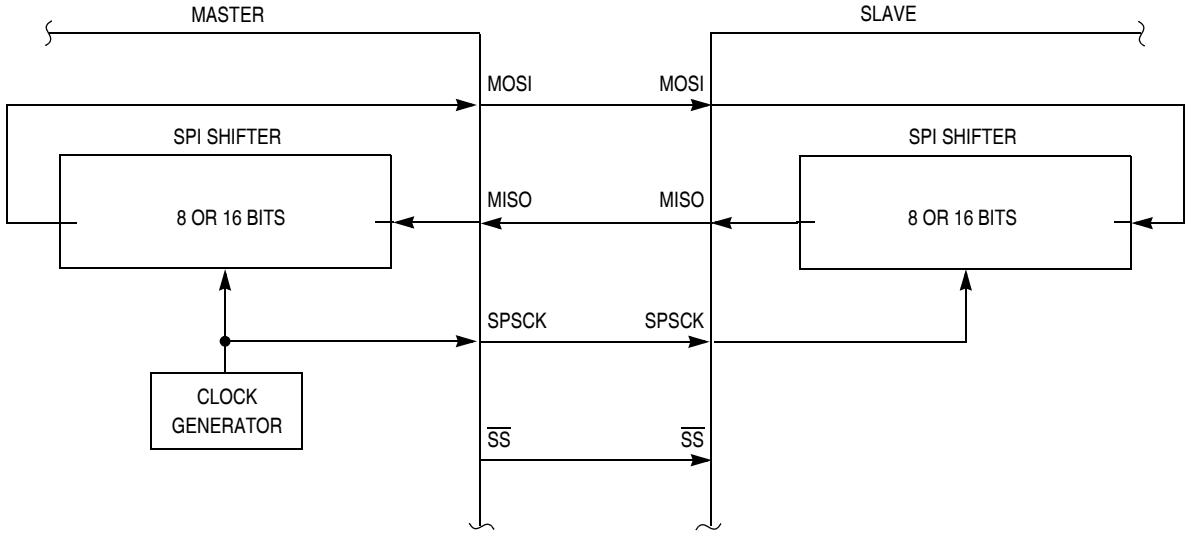


Figure 13-1. SPI System Connections

13.1.3.2 SPI Module Block Diagram

Figure 13-2 is a block diagram of the SPI module. The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter (write to SPIxDH:SPIxDL) and gets transferred to the SPI shift register at the start of a data transfer. After shifting in 8 or 16 bits (as determined by SPIMODE bit) of data, the data is transferred into the double-buffered receiver where it can be read (read from SPIxDH:SPIxDL). Pin multiplexing logic controls connections between MCU pins and the SPI module.

Additionally there is an 8-byte receive FIFO and an 8-byte transmit FIFO that once enabled provide features to allow less CPU interrupts to occur when transmitting/receiving high volume/high speed data. When FIFO mode is enabled, the SPI can still function in either 8-bit or 16-bit mode (as per **SPIMODE** bit) and 3 additional flags help monitor the FIFO status and two of these flags can provide CPU interrupts.

When the SPI is configured as a master, the clock output is routed to the SPSCK pin, the shifter output is routed to MOSI, and the shifter input is routed from the MISO pin.

When the SPI is configured as a slave, the SPSCK pin is routed to the clock input of the SPI, the shifter output is routed to MISO, and the shifter input is routed from the MOSI pin.

In the external SPI system, simply connect all SPSCK pins to each other, all MISO pins together, and all MOSI pins together. Peripheral devices often use slightly different names for these pins.

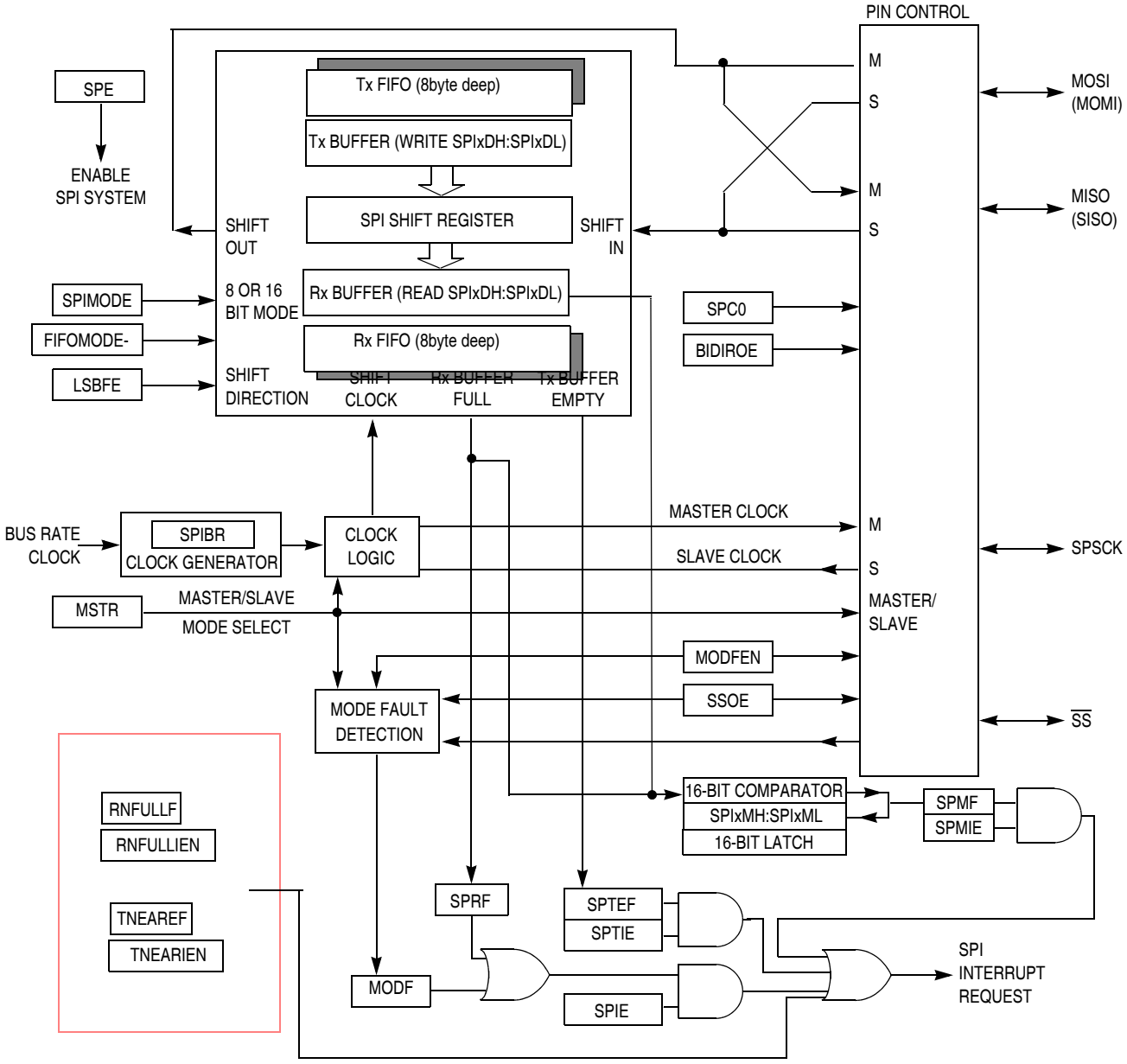


Figure 13-2. SPI Module Block Diagram

13.2 External Signal Description

The SPI optionally shares four port pins. The function of these pins depends on the settings of SPI control bits. When the SPI is disabled (SPE = 0), these four pins revert to being general-purpose port I/O pins that are not controlled by the SPI.

13.2.1 SPSCK — SPI Serial Clock

When the SPI is enabled as a slave, this pin is the serial clock input. When the SPI is enabled as a master, this pin is the serial clock output.

13.2.2 MOSI — Master Data Out, Slave Data In

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data output. When the SPI is enabled as a slave and SPC0 = 0, this pin is the serial data input. If SPC0 = 1 to select single-wire bidirectional mode, and master mode is selected, this pin becomes the bidirectional data I/O pin (MOMI). Also, the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE = 0) or an output (BIDIROE = 1). If SPC0 = 1 and slave mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

13.2.3 MISO — Master Data In, Slave Data Out

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data input. When the SPI is enabled as a slave and SPC0 = 0, this pin is the serial data output. If SPC0 = 1 to select single-wire bidirectional mode, and slave mode is selected, this pin becomes the bidirectional data I/O pin (SISO) and the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE = 0) or an output (BIDIROE = 1). If SPC0 = 1 and master mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

13.2.4 $\overline{SS}$ — Slave Select

When the SPI is enabled as a slave, this pin is the low-true slave select input. When the SPI is enabled as a master and mode fault enable is off (MODFEN = 0), this pin is not used by the SPI and reverts to being a general-purpose port I/O pin. When the SPI is enabled as a master and MODFEN = 1, the slave select output enable bit determines whether this pin acts as the mode fault input (SSOE = 0) or as the slave select output (SSOE = 1).

13.3 Register Definition

The SPI has eight 8-bit registers to select SPI options, control baud rate, report SPI status, hold an SPI data match value, and for transmit/receive data.

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all SPI registers. This section refers to registers and control bits only by their names, and a Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

13.3.1 SPI Control Register 1 (SPIxC1)

This read/write register includes the SPI enable control, interrupt enables, and configuration options.

	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 13-3. SPI Control Register 1 (SPIxC1)

Table 13-1. SPIxC1 Field Descriptions

Field	Description
7 SPIE	**FIFOMODE=0** **SPI Interrupt Enable (for SPRF and MODF)** — This is the interrupt enable for SPI receive buffer full (SPRF) and mode fault (MODF) events. 0 Interrupts from SPRF and MODF inhibited (use polling) 1 When SPRF or MODF is 1, request a hardware interrupt **FIFOMODE=1** **SPI Read FIFO Full Interrupt Enable** — This bit when set enables the SPI to interrupt the CPU when the Receive FIFO is full. An interrupt will occur when SPRF flag is set or MODF is set. 0 Read FIFO Full Interrupts are disabled 1 Read FIFO Full Interrupts are enabled
6 SPE	**SPI System Enable** — This bit enables the SPI system and dedicates the SPI port pins to SPI system functions. If SPE is cleared, SPI is disabled and forced into idle state, and all status bits in the SPIxS register are reset. 0 SPI system inactive 1 SPI system enabled
5 SPTIE	**SPI Transmit Interrupt Enable** — **FIFOMODE=0** This is the interrupt enable bit for SPI transmit buffer empty (SPTEF). An interrupt occurs when the SPI transmit buffer is empty (SPTEF is set) **FIFOMODE=1** This is the interrupt enable bit for SPI transmit FIFO empty (SPTEF). An interrupt occurs when the SPI transmit FIFO is empty (SPTEF is set) 0 Interrupts from SPTEF inhibited (use polling) 1 When SPTEF is 1, hardware interrupt requested
4 MSTR	**Master/Slave Mode Select —** This bit selects master or slave mode operation. 0 SPI module configured as a slave SPI device 1 SPI module configured as a master SPI device
3 CPOL	**Clock Polarity** — This bit selects an inverted or non-inverted SPI clock. To transmit data between SPI modules, the SPI modules must have identical CPOL values. This bit effectively places an inverter in series with the clock signal from a master SPI or to a slave SPI device. Refer to Section 13.4.6, “SPI Clock Formats” for more details. 0 Active-high SPI clock (idles low) 1 Active-low SPI clock (idles high)
2 CPHA	**Clock Phase** — This bit selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to Section 13.4.6, “SPI Clock Formats” for more details. 0 First edge on SPSCK occurs at the middle of the first cycle of a data transfer 1 First edge on SPSCK occurs at the start of the first cycle of a data transfer

Table 13-1. SPIxC1 Field Descriptions (continued)

Field	Description
1 SSOE	**Slave Select Output Enable** — This bit is used in combination with the mode fault enable (MODFEN) bit in SPIxC2 and the master/slave (MSTR) control bit to determine the function of the $\overline{SS}$ pin as shown in Table 13-2.
0 LSBFE	**LSB First (Shifter Direction) —** This bit does not affect the position of the MSB and LSB in the data register. Reads and writes of the data register always have the MSB in bit 7 (or bit 15 in 16-bit mode). 0 SPI serial data transfers start with most significant bit 1 SPI serial data transfers start with least significant bit

Table 13-2. $\overline{SS}$ Pin Function

MODFEN	SSOE	Master Mode	Slave Mode
0	0	General-purpose I/O (not SPI)	Slave select input
0	1	General-purpose I/O (not SPI)	Slave select input
1	0	$\overline{SS}$ input for mode fault	Slave select input
1	1	Automatic $\overline{SS}$ output	Slave select input

13.3.2 SPI Control Register 2 (SPIxC2)

This read/write register is used to control optional features of the SPI system. Bits 6 and 5 are not implemented and always read 0.

	7	6	5	4	3	2	1	0
R	SPMIE	SPIMODE	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 13-4. SPI Control Register 2 (SPIxC2)

Table 13-3. SPIxC2 Register Field Descriptions

Field	Description
7 SPMIE	**SPI Match Interrupt Enable** — This is the interrupt enable for the SPI receive data buffer hardware match (SPMF) function. 0 Interrupts from SPMF inhibited (use polling). 1 When SPMF = 1, requests a hardware interrupt.
6 SPIMODE	**SPI 8- or 16-bit Mode** — This bit allows the user to select either an 8-bit or 16-bit SPI data transmission length. In master mode, a change of this bit will abort a transmission in progress, force the SPI system into idle state, and reset all status bits in the SPIxS register. Refer to section Section 13.4.4, “SPI FIFO MODE,” for details. 0 8-bit SPI shift register, match register, and buffers. 1 16-bit SPI shift register, match register, and buffers.

Table 13-3. SPIxC2 Register Field Descriptions (continued)

Field	Description
4 MODFEN	**Master Mode-Fault Function Enable** — When the SPI is configured for slave mode, this bit has no meaning or effect. (The $\overline{SS}$ pin is the slave select input.) In master mode, this bit determines how the $\overline{SS}$ pin is used (refer to Table 13-2 for details) 0 Mode fault function disabled, master $\overline{SS}$ pin reverts to general-purpose I/O not controlled by SPI 1 Mode fault function enabled, master $\overline{SS}$ pin acts as the mode fault input or the slave select output
3 BIDIROE	**Bidirectional Mode Output Enable** — When bidirectional mode is enabled by SPI pin control 0 (SPC0) = 1, BIDIROE determines whether the SPI data output driver is enabled to the single bidirectional SPI I/O pin. Depending on whether the SPI is configured as a master or a slave, it uses either the MOSI (MOMI) or MISO (SISO) pin, respectively, as the single SPI data I/O pin. When SPC0 = 0, BIDIROE has no meaning or effect. 0 Output driver disabled so SPI data I/O pin acts as an input 1 SPI I/O pin enabled as an output
1 SPISWAI	**SPI Stop in Wait Mode —** This bit is used for power conservation while in wait. 0 SPI clocks continue to operate in wait mode 1 SPI clocks stop when the MCU enters wait mode
0 SPC0	**SPI Pin Control 0** — This bit enables bidirectional pin configurations as shown in Table 13-4. 0 SPI uses separate pins for data input and data output. 1 SPI configured for single-wire bidirectional operation.

Table 13-4. 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	SlaveIn
Bidirectional	1	0	Slave In	MOSI not used by SPI
		1	Slave I/O	

13.3.3 SPI Baud Rate Register (SPIxBR)

This register is used to set the prescaler and bit rate divisor for an SPI master. This register may be read or written at any time.

	7	6	5	4	3	2	1	0
R	0	SPPR2	SPPR1	SPPR0	SPR3	SPR2	SPR1	SPR0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 13-5. SPI Baud Rate Register (SPIxBR)

Table 13-5. SPIxBR Register Field Descriptions

Field	Description
6:4 SPPR[2:0]	**SPI Baud Rate Prescale Divisor** — This 3-bit field selects one of eight divisors for the SPI baud rate prescaler as shown in Table 13-6. The input to this prescaler is the bus rate clock (BUSCLK). The output of this prescaler drives the input of the SPI baud rate divider (see Figure 13-18). See Section 13.4.7, "SPI Baud Rate Generation," for details.
2:0 SPR[3:0]	**SPI Baud Rate Divisor** — This 4-bit field selects one of nine divisors for the SPI baud rate divider as shown in Table 13-7. The input to this divider comes from the SPI baud rate prescaler (see Figure 13-18). See Section 13.4.7, "SPI Baud Rate Generation," for details.

Table 13-6. SPI Baud Rate Prescaler Divisor

SPPR2:SPPR1:SPPR0	Prescaler Divisor
0:0:0	1
0:0:1	2
0:1:0	3
0:1:1	4
1:0:0	5
1:0:1	6
1:1:0	7
1:1:1	8

Table 13-7. SPI Baud Rate Divisor

SPR3:SPR2:SPR1:SPR0	Rate Divisor
0:0:0:0	2
0:0:0:1	4
0:0:1:0	8
0:0:1:1	16
0:1:0:0	32
0:1:0:1	64
0:1:1:0	128
0:1:1:1	256
1:0:0:0	512
All other combinations	Reserved

13.3.4 SPI Status Register (SPIxS)

This register has eight read-only status bits. Writes have no meaning or effect. This register has 4additional flags RNFULLF, TNEARF, TXFULLF and RFIFOEF which provide mechanisms to support an 8-byte FIFO mode. When in 8byte FIFO mode the function of SPRF and SPTEF differs slightly from the normal buffered modes, mainly in how these flags are cleared by the amount available in the transmit and receive FIFOs

	7	6	5	4	3	2	1	0
R	SPRF	SPMF	SPTEF	MODF	RNFULLF	TNEAREF	TXFULLF	RFIFOEF
W								
Reset	0	0	1	0	0	0[1]	0	0[1]

= Unimplemented or Reserved

Figure 13-6. SPI Status Register (SPIxS)

[1] Note: PoR values of TNEAREF and RFIFOEF is 0. If status register is reset due to change of SPIMODE, FIFOMODE or SPE than, if FIFOMODE = 1, TNEAREF and RFIFOEF resets to 1 else if FIFOMODE = 0, TNEAREF and RFIFOEF resets to 0

Table 13-8. SPIxS Register Field Descriptions

Field	Description
7 SPRF	**SPI Read Buffer Full Flag** — SPRF is set at the completion of an SPI transfer to indicate that received data may be read from the SPI data register (SPIxDH:SPIxDL). SPRF is cleared by reading SPRF while it is set, then reading the SPI data register. 0 No data available in the receive data buffer. 1 Data available in the receive data buffer. **FIFOMODE=1** **SPI Read FIFO FULL Flag** — This bit indicates the status of the Read FIFO when FIFOMODE enabled. The SPRF is set when the read FIFO has received 64bits (4 words or 8bytes) of data from the shifter and there has been no CPU reads of SPIxDH:SPIxDL. SPRF is cleard by reading the SPI Data Register, which empties the FIFO, assuming another SPI message is not received. 0 Read FIFO is not Full 1 Read FIFO is Full.
6 SPMF	**SPI Match Flag** — SPMF is set after SPRF = 1 when the value in the receive data buffer matches the value in SPIMH:SPIML. To clear the flag, read SPMF when it is set, then write a 1 to it. 0 Value in the receive data buffer does not match the value in SPIxMH:SPIxML registers. 1 Value in the receive data buffer matches the value in SPIxMH:SPIxML registers.

Table 13-8. SPIxS Register Field Descriptions

Field	Description
5 SPTEF	**SPI Transmit Buffer Empty Flag** — This bit is set when the transmit data buffer is empty. It is cleared by reading SPIxS with SPTEF set, followed by writing a data value to the transmit buffer at SPIxDH:SPIxDL. SPIxS must be read with SPTEF = 1 before writing data to SPIxDH:SPIxDL or the SPIxDH:SPIxDL write will be ignored. SPTEF is automatically set when all data from the transmit buffer transfers into the transmit shift register. For an idle SPI, data written to SPIxDH:SPIxDL is transferred to the shifter almost immediately so SPTEF is set within two bus cycles allowing a second data to be queued into the transmit buffer. After completion of the transfer of the data in the shift register, the queued data from the transmit buffer will automatically move to the shifter and SPTEF will be set to indicate there is room for new data in the transmit buffer. If no new data is waiting in the transmit buffer, SPTEF simply remains set and no data moves from the buffer to the shifter. 0 SPI transmit buffer not empty 1 SPI transmit buffer empty **FIFOMODE=1** **SPI Transmit FIFO Empty Flag** — *This bit when in FIFOMODE now changed to provide status of the FIFO rather than an 8or16-bit buffer.* This bit is set when the Transmit FIFO is empty. It is cleared by writing a data value to the transmit FIFO at SPIxDH:SPIxDL. SPTEF is automatically set when all data from transmit FIFO transfers into the transmit shift register. For an idle SPI, data written to SPIxDH:SPIxDL is transferred to the shifter almost immediately so SPTEF is set within two bus cycles, a second write of data to the SPIxDH:SPIxDL will clear ths SPTEF flag. After completion of the transfer of the data in the shift register, the queued data from the transmit FIFO will automatically move to the shifter and SPTEF will be set only when all data written to the transmit FIFO has been transfered to the shifter. If no new data is waiting in the transmit FIFO, SPTEF simply remains set and no data moves from the buffer to the shifter. 0 SPI FIFO not empty 1 SPI FIFO empty
4 MODF	**Master Mode Fault Flag** — MODF is set if the SPI is configured as a master and the slave select input goes low, indicating some other SPI device is also configured as a master. The $\overline{SS}$ pin acts as a mode fault error input only when MSTR = 1, MODFEN = 1, and SSOE = 0; otherwise, MODF will never be set. MODF is cleared by reading MODF while it is 1, then writing to SPI control register 1 (SPIxC1). 0 No mode fault error 1 Mode fault error detected
3 RNFULLF	**Receive FIFO Nearly Full Flag** — This flag is set when more than three16bit word or six 8bit bytes of data remain in the receive FIFO provided SPIxC3[4] = 0 or when more than two 16bit word or four 8bit bytes of data remain in the receive FIFO provided SPIxC3[4] = 1. It has no function if FIFOMODE=0. 0 Receive FIFO has received less than 48bits/32bits (See SPIxC3[4]). 1 Receive FIFO has receieved 48bits/32bits(See SPIxC3[4]) or more.
2 TNEAREF	**Transmit FIFO Nearly Empty Flag** — This flag is set when only one 16bit word or 2 8bit bytes of data remain in the transmit FIFO provided SPIxC3[5] = 0 or when only two 16bit word or 4 8bit bytes of data remain in the transmit FIFO provided SPIxC3[5] =1. If FIFOMODE is not enabled this bit should be ignored. 0 Transmit FIFO has more than 16bits/32bits (See SPIxC3[5]) left to transmit. 1 Transmit FIFO has 16bits/32 bits(See SPIxC3[5])or less left to transmit
1 TXFULLF	**Transmit FIFO Full Flag** - This bit indicates status of transmit fifo when fifomode is enabled. This flag is set when there are 8 bytes in transmit fifo. If FIFOMODE is not enabled this bit should be ignored. 0 Transmit FIFO has less than 8 bytes. 1 Transmit FIFO has 8 bytes of data.
0 RFIFOEF	**SPI Read FIFO Empty Flag** — This bit indicates the status of the Read FIFO when FIFOMODE enabled. If FIFOMODE is not enabled this bit should be ignored. 0 Read FIFO has data. Reads of the SPIxDH:SPIxDL registers in 16-bit mode or SPIxDL register in 8-bit mode will empty the Read FIFO. 1 Read FIFO is empty.

For FIFO management there are two other important flags that are used to help make the operation more efficient when transfering large amounts of data. These are the Receive FIFO Nearly Full Flag (RNFULLF) and the Transmit FIFO Nearly Empty Flag (TNEAREF). Both these flags provide a “watermark” feature of the FIFOs to allow continuous transmissions of data when running at high speed.

The RNFULLF flag can generate an interrupt if the RNFULLIEN bit in the SPIxC3 Register is set which allows the CPU to start emptying the Receive FIFO without delaying the reception of subsequent bytes. The user can also determine if all data in Receive FIFO has been read by monitoring the RFIFOEF flag.

The TNEAREF flag can generate an interrupt if the TNEARIEN bit n the SPIxC3 Register is set which allows the CPU to start filling the Transmit FIFO before it is empty and thus provide a mechanism to have no breaks in SPI transmission.

NOTE

SPIxS and both TX and RX fifos gets reset due to change in SPIMODE, FIFOMODE or SPE. PoR values of SPIxS are show in Figure 13-7.

Figure 13-7 and Figure 13-8 shows the reset values due to change of modes after PoR.

	7	6	5	4	3	2	1	0
R	SPRF	SPMF	SPTEF	MODF	RNFULLF	TNEAREF	TXFULLF	RFIFOEF
W								
Reset	0	0	1	0	0	0	0	0

Figure 13-7. Reset values of SPIxS after PoR with FIFOMODE = 0

	7	6	5	4	3	2	1	0
R	SPRF	SPMF	SPTEF	MODF	RNFULLF	TNEAREF	TXFULLF	RFIFOEF
W								
Reset	0	0	1	0	0	1	0	1

Figure 13-8. Reset values of SPIxS after PoR with FIFOMODE = 1

13.3.5 SPI Data Registers (SPIxDH:SPIxDL)

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

Figure 13-9. SPI Data Register High (SPIxDH)

	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 13-10. SPI Data Register Low (SPIxDL)

The SPI data registers (SPIxDH:SPIxDL) are both the input and output register for SPI data. A write to these registers writes to the transmit data buffer, allowing data to be queued and transmitted.

When the SPI is configured as a master, data queued in the transmit data buffer is transmitted immediately after the previous transmission has completed.

The SPI transmit buffer empty flag (SPTEF) in the SPIxS register indicates when the transmit data buffer is ready to accept new data. SPIxS must be read when SPTEF is set before writing to the SPI data registers, or the write will be ignored.

Data may be read from SPIxDH:SPIxDL any time after SPRF is set and before another transfer is finished. Failure to read the data out of the receive data buffer before a new transfer ends causes a receive overrun condition and the data from the new transfer is lost.

In 8-bit mode, only SPIxDL is available. Reads of SPIxDH will return all 0s. Writes to SPIxDH will be ignored.

In 16-bit mode, reading either byte (SPIxDH or SPIxDL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (SPIxDH or SPIxDL) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent 16-bit value into the transmit data buffer.

13.3.6 SPI Match Registers (SPIxMH:SPIxML)

These read/write registers contain the hardware compare value, which sets the SPI match flag (SPMF) when the value received in the SPI receive data buffer equals the value in the SPIxMH:SPIxML registers.

In 8-bit mode, only SPIxML is available. Reads of SPIxMH will return all 0s. Writes to SPIxMH will be ignored.

In 16-bit mode, reading either byte (SPIxMH or SPIxML) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (SPIxMH or SPIxML) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent value into the SPI match registers.

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

Figure 13-11. SPI Match Register High (SPIxMH)

	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 13-12. SPI Match Register Low (SPIxML)

13.3.7 SPI Control Register 3 (SPIxC3) — Enable FIFO Feature

The SPI Control Register 3 introduces a 64bit FIFO function on both transmit and receive buffers to be utilised on the SPI. Utilising this FIFO feature allows the SPI to provide high speed transfers of large amounts of data without consuming large amounts of the CPU bandwidth.

Enabling this FIFO function will effect the behaviour of some of the Read/Write Buffer flags in the SPIxS register namely:

The SPRF of the SPIxS register will be set when the Receive FIFO is filled and will interrupt the CPU if the SPIE in the SPIxC1 register is set.

The SPTEF of the SPIxS register will be set when the Transmit FIFO is empty, and will interrupt the CPU if the SPITIE bit is set in the SPIxC1 register. See SPIxC1 and SPIxS registers.

FIFO mode is enabled by setting the FIFOMODE bit, and provides the SPI with an 8-byte receive FIFO and an 8-byte transmit FIFO to reduce the amount of CPU interrupts for high speed/high volume data transfers.

Two interrupt enable bits TNEARIEN and RNFULLIEN provide CPU interrupts based on the “watermark” feature of the TNEARF and RNFULLF flags of the SPIxS register.

Note: This register has sixread/write control bits. Bits 7 thro’ 6are not implemented and always read 0. Writes have no meaning or effect. Write to this register happens only when FIFOMODE bit is 1.

	7	6	5	4	3	2	1	0
R W			TNEAREF MARK	RNFULL MARK	INTCLR	TNEARIEN	RNFULLIEN	FIFOMODE
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 13-13. SPI Status Register (SPIxC3)

Table 13-9. SPIxC3 Register Field Descriptions

Field	Description
5 TNEAREF MARK	**Transmit FIFO Nearly Empty Water Mark** - This bit selects the mark after which TNEAREF flag is asserted. 0 TNEAREF is set when Transmit FIFO has16bits or less. 1 TNEAREF is set when Transmit FIFO has 32bits or less.
4 RNFULLF MARK	**Receive FIFO Nearly Full Water Mark** - This bit selects the mark for which RNFULLF flag is asserted 0 RNFULLF is set when Receive FIFO has 48bits or more 1 RNFULLF is set when Receive FIFO has 32bits or more.

Table 13-9. SPIxC3 Register Field Descriptions

Field	Description
3 INTCLR	**Interrupt Clearing Mechanism Select** - This bit selects the mechanism by which SPRF, SPTEF, TNEAREF, RNFULLF interrupts gets cleared. 0 Interrupts gets cleared when respective flags gets cleared depending on the state of FIFOs 1 Interrupts gets cleared by writing to the SPIxCI respective bits.
2 TNEARIEN	**Transmit FIFO Nearly Empty Interrupt Enable** — Writing to this bit enables the SPI to interrupt the CPU when the TNEAREF flag is set. This is an additional interrupt on the SPI and will only interrupt the CPU if SPTIE in the SPIxC1 register is also set. This bit is ignored and has no function if FIFOMODE=0. 0 No interrupt on Transmit FIFO Nearly Empty Flag being set. 1 Enable interrupts on Transmit FIFO Nearly Empty Flag being set.
1 RNFULLIEN	**Receive FIFO Nearly Full Interrupt Enable** — Writing to this bit enables the SPI to interrupt the CPU when the RNEARFF flag is set. This is an additional interrupt on the SPI and will only interrupt the CPU if SPIE in the SPIxC1 register is also set. This bit is ignored and has no function if FIFOMODE = 0. 0 No interrupt on RNEARFF being set. 1 Enable interrupts on RNEARFF being set.
0 FIFOMODE	**SPI FIFO Mode Enable** — This bit enables the SPI to utilise a 64-bit FIFO (8bytes 4 16-bit words) for both transmit and receive buffers. 0 Buffer mode disabled. 1 Data available in the receive data buffer.

13.3.8 SPI Clear Interrupt Register (SPIxCI)

The SPI Clear Interrupt register has 4 bits dedicated for clearing the interrupts. Writing 1 to these bits clears the respective interrupts if INTCLR bit in SPIxCR3 is set.

It also have 2 bits to indicate the transmit fifo and receive fifo overrun conditions. When receive fifo is full and a data is received RXFOF flag is set. Similarily when transmit fifo is full and write happens to SPIDR TXFOF is set. These flags gets cleared when a read happens to this register with the flags set.

There are two more bits to indicate the error flags. These flags gets set when due to some spurious reasons entries in fifo becomes greater than 8. At this point all the flags in status register gets reset and entries in FIFO are flushed with respective error flags set. These flags are cleared when a read happen at SPIxCI with the error flags set.

Note: Bits [7:4] are readonly bits. These bits gets cleared when a read happens to this register with the flags set. Bits [3:0] are clear interrupts bits which clears the interrupts by writing 1 to respective bits. Reading these bits always return 0.

	7	6	5	4	3	2	1	0
R	TXFERR	RXFERR	TXFOF	RXFOF	TNEAREFCI	RNFULLFCI	SPTEFCI	SPRFCI
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Table 13-10. SPIxCI Register Field Descriptions

Field	Description
7 TXFERR	**Transmit FIFO ErrorFlag**- This flag indicates that TX FIFO error occured because entries in fifo goes above 8.. 0 No TX Fifo Error Occured 1 TX Fifo error occured.
6 RXFERR	**Receive FIFO Error Flag**- This flag indicates that RX FIFO error occured because entries in fifo goes above 8. 0 No RX Fifo Error Occured 1 RX Fifo error occured.
5 TXFOF	**TX FIFO Overflow Flag**- This Flag indicates that TX FIFO overflow condition has occured.. 0 TX FIFO overflow condition has not occured. 1 TX FIFO overflow condition occured.
4 RXFOF	**RX FIFO Overflow Flag** - This Flag indicates that RX FIFO overflow condition has occured.. 0 RX FIFO overflow condition has not occured. 1 RX FIFO overflow condition occured.
3 TNEAREFCI	**Transmit FIFO Nearly Empty Flag Clear Interrupt Register** - Write of 1 clears the TNEAREF interrupt provided SPIxC3[3] is set.
2 RNFULLFCI	**Receive FIFO Nearly Full Flag Clear Interrupt Register** - Write of 1 clears the RNFULLF interrupt provided SPIxC3[3] is set.
1 SPTEFCI	**Transmit FIFO Empty Flag Clear Interrupt Register** - Write of 1 clears the SPTEF interrupt provided SPIxC3[3] is set.
0 SPRFCI	**Receive FIFO Full Flag Clear Interrupt Register** - Write of 1 clears the TNEAREF interrupt provided SPIxC3[3] is set.

13.4 Functional Description

13.4.1 General

The SPI system is enabled by setting the SPI enable (SPE) bit in SPI Control Register 1. While the SPE bit is set, the four associated SPI port pins are dedicated to the SPI function as:

- Slave select ($\overline{SS}$)
- Serial clock (SPSCK)
- Master out/slave in (MOSI)
- Master in/slave out (MISO)

An SPI transfer is initiated in the master SPI device by reading the SPI status register (SPIxS) when SPTEF = 1 and then writing data to the transmit data buffer (write to SPIxDH:SPIxDL). When a transfer is complete, received data is moved into the receive data buffer. The SPIxDH:SPIxDL registers act as the SPI receive data buffer for reads and as the SPI transmit data buffer for writes.

The clock phase control bit (CPHA) and a clock polarity control bit (CPOL) in the SPI Control Register 1 (SPIxC1) select one of four possible clock formats to be used by the SPI system. The CPOL bit simply selects a non-inverted or inverted clock. The CPHA bit is used to accommodate two fundamentally different protocols by sampling data on odd numbered SPSCK edges or on even numbered SPSCK edges.

The SPI can be configured to operate as a master or as a slave. When the MSTR bit in SPI control register 1 is set, master mode is selected, when the MSTR bit is clear, slave mode is selected.

13.4.2 Master Mode

The SPI operates in master mode when the MSTR bit is set. Only a master SPI module can initiate transmissions. A transmission begins by reading the SPIxS register while SPTEF = 1 and writing to the master SPI data registers. If the shift register is empty, the byte immediately transfers to the shift register. The data begins shifting out on the MOSI pin under the control of the serial clock.

- SPSCK

The SPR3, SPR2, SPR1, and SPR0 baud rate selection bits in conjunction with the SPPR2, SPPR1, and SPPR0 baud rate preselection bits in the SPI Baud Rate register control the baud rate generator and determine the speed of the transmission. The SPSCK pin is the SPI clock output. Through the SPSCK pin, the baud rate generator of the master controls the shift register of the slave peripheral.

- MOSI, MISO pin

In master mode, the function of the serial data output pin (MOSI) and the serial data input pin (MISO) is determined by the SPC0 and BIDIROE control bits.

- $\overline{SS}$ pin

If MODFEN and SSOE bit are set, the $\overline{SS}$ pin is configured as slave select output. The $\overline{SS}$ output becomes low during each transmission and is high when the SPI is in idle state.

If MODFEN is set and SSOE is cleared, the $\overline{SS}$ pin is configured as input for detecting mode fault error. If the $\overline{SS}$ input becomes low this indicates a mode fault error where another master tries to drive the MOSI and SPSCK lines. In this case, the SPI immediately switches to slave mode, by clearing the MSTR bit and also disables the slave output buffer MISO (or SISO in bidirectional mode). So the result is that all outputs are disabled and SPSCK, MOSI and MISO are inputs. If a transmission is in progress when the mode fault occurs, the transmission is aborted and the SPI is forced into idle state.

This mode fault error also sets the mode fault (MODF) flag in the SPI Status Register (SPIxS). If the SPI interrupt enable bit (SPIE) is set when the MODF flag gets set, then an SPI interrupt sequence is also requested.

When a write to the SPI Data Register in the master occurs, there is a half SPSCK-cycle delay. After the delay, SPSCK is started within the master. The rest of the transfer operation differs slightly, depending on the clock format specified by the SPI clock phase bit, CPHA, in SPI Control Register 1 (see Section 13.4.6, “SPI Clock Formats”).

NOTE

A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0, BIDIROE with SPC0 set, SPIMODE, FIFOMODE, SPPR2-SPPR0 and SPR3-SPR0 in master mode will abort a transmission in progress and force the SPI into idle state. The remote slave cannot detect this, therefore the master has to ensure that the remote slave is set back to idle state.

13.4.3 Slave Mode

The SPI operates in slave mode when the MSTR bit in SPI Control Register1 is clear.

- SPSCK

In slave mode, SPSCK is the SPI clock input from the master.

- MISO, MOSI pin

In slave mode, the function of the serial data output pin (MISO) and serial data input pin (MOSI) is determined by the SPC0 bit and BIDIROE bit in SPI Control Register 2.

- $\overline{SS}$ pin

The $\overline{SS}$ pin is the slave select input. Before a data transmission occurs, the $\overline{SS}$ pin of the slave SPI must be low. $\overline{SS}$ must remain low until the transmission is complete. If $\overline{SS}$ goes high, the SPI is forced into idle state.

The $\overline{SS}$ input also controls the serial data output pin, if $\overline{SS}$ is high (not selected), the serial data output pin is high impedance, and, if $\overline{SS}$ 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 ($\overline{SS}$ is high), then the SPSCK input is ignored and no internal shifting of the SPI shift register takes place.

Although the SPI is capable of duplex operation, some SPI peripherals are capable of only receiving SPI data in a slave mode. For these simpler devices, there is no serial data out pin.

NOTE

When peripherals with duplex capability are used, take care not to simultaneously enable two receivers whose serial outputs drive the same system slave's serial data output line.

As long as no more than one slave device drives the system slave's serial data output line, it is possible for several slaves to receive the same transmission from a master, although the master would not receive return information from all of the receiving slaves.

If the CPHA bit in SPI Control Register 1 is clear, odd numbered edges on the SPSCK input cause the data at the serial data input pin to be latched. Even numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

If the CPHA bit is set, even numbered edges on the SPSCK input cause the data at the serial data input pin to be latched. Odd numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

When CPHA is set, the first edge is used to get the first data bit onto the serial data output pin. When CPHA is clear and the $\overline{SS}$ input is low (slave selected), the first bit of the SPI data is driven out of the serial data output pin. After the eighth (SPIMODE = 0) or sixteenth (SPIMODE = 1) shift, the transfer is considered complete and the received data is transferred into the SPI data registers. To indicate transfer is complete, the SPRF flag in the SPI Status Register is set.

NOTE

A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0 and BIDIROE with SPC0 set FIFOMODE and SPIMODE in slave mode will corrupt a transmission in progress and has to be avoided.

13.4.4 SPI FIFO MODE

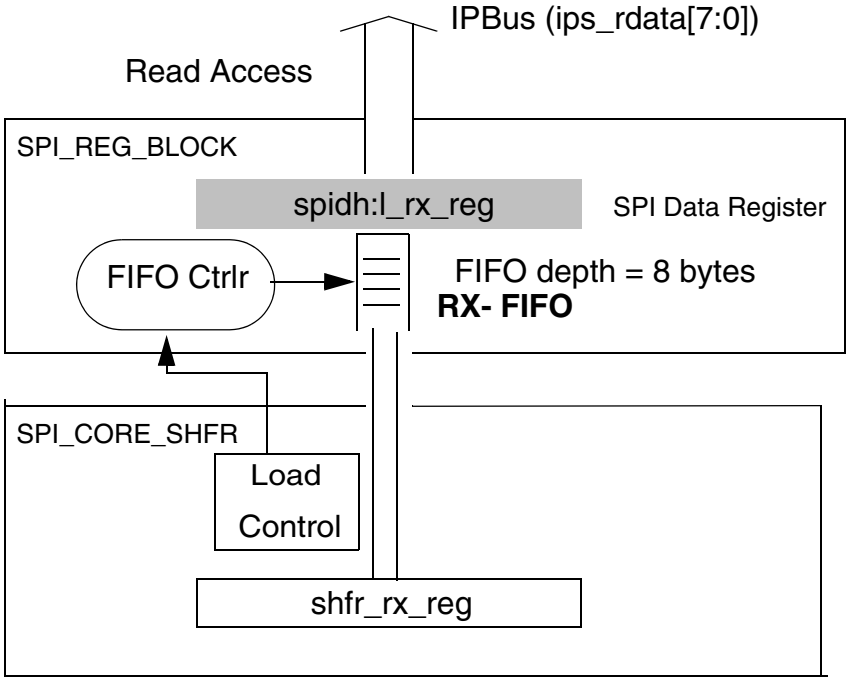


Figure 13-14. SPIDH:L read side structural overview in FIFO mode

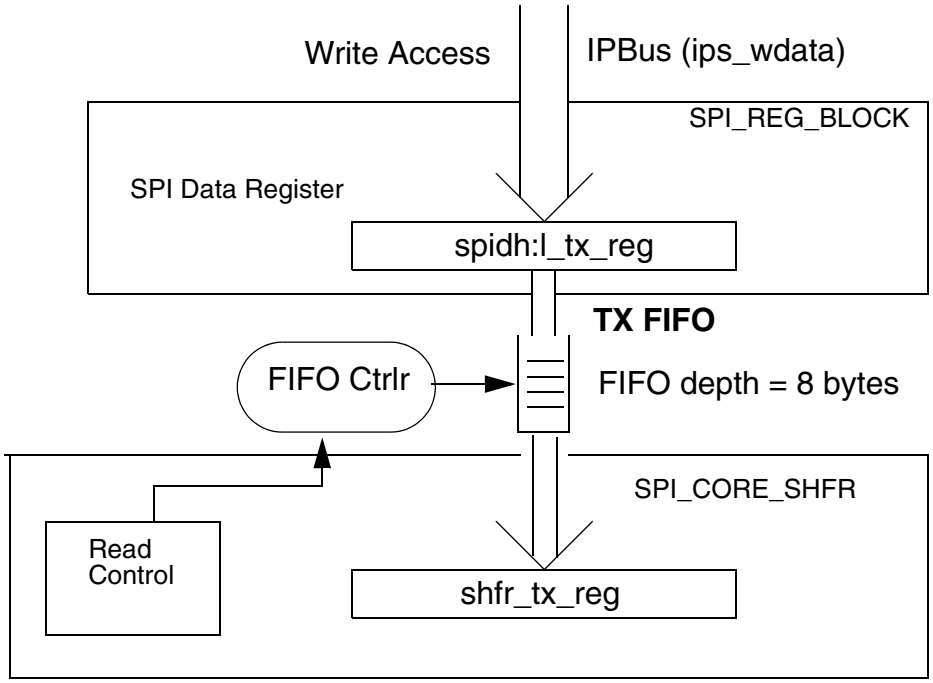


Figure 13-15. SPIDH:L write side structural overview in FIFO mode

SPI works in FIFO mode when SPIxC3[0] bit is set. When in FIFO mode SPI RX buffer and SPI TX buffer is replaced by a 8 byte deep fifo as shown in figure above.

13.4.5 Data Transmission Length

The SPI can support data lengths of 8 or 16 bits. The length can be configured with the SPIMODE bit in the SPIxC2 register.

In 8-bit mode (SPIMODE = 0), the SPI Data Register is comprised of one byte: SPIxDL. The SPI Match Register is also comprised of only one byte: SPIxML. Reads of SPIxDH and SPIxMH will return zero. Writes to SPIxDH and SPIxMH will be ignored.

In 16-bit mode (SPIMODE = 1), the SPI Data Register is comprised of two bytes: SPIxDH and SPIxDL. Reading either byte (SPIxDH or SPIxDL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (SPIxDH or SPIxDL) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent 16-bit value into the transmit data buffer.

In 16-bit mode, the SPI Match Register is also comprised of two bytes: SPIxMH and SPIxML. Reading either byte (SPIxMH or SPIxML) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (SPIxMH or SPIxML) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent 16-bit value into the transmit data buffer.

Any switching between 8- and 16-bit data transmission length (controlled by SPIMODE bit) in master mode will abort a transmission in progress, force the SPI system into idle state, and reset all status bits in the SPIxS register. To initiate a transfer after writing to SPIMODE, the SPIxS register must be read with SPTEF = 1, and data must be written to SPIxDH:SPIxDL in 16-bit mode (SPIMODE = 1) or SPIxDL in 8-bit mode (SPIMODE = 0).

In slave mode, user software should write to SPIMODE only once to prevent corrupting a transmission in progress.

NOTE

Data can be lost if the data length is not the same for both master and slave devices.

13.4.6 SPI Clock Formats

To accommodate a wide variety of synchronous serial peripherals from different manufacturers, the SPI system has a clock polarity (CPOL) bit and a clock phase (CPHA) control bit to select one of four clock formats for data transfers. CPOL selectively inserts an inverter in series with the clock. CPHA chooses between two different clock phase relationships between the clock and data.

Figure 13-16 shows the clock formats when SPIMODE = 0 (8-bit mode) and CPHA = 1. At the top of the figure, the eight bit times are shown for reference with bit 1 starting at the first SPSCK edge and bit 8 ending one-half SPSCK cycle after the sixteenth SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{SS}$ OUT waveform applies to the slave select output from a master (provided

MODFEN and SSOE = 1). The master $\overline{SS}$ output goes to active low one-half SPSCK cycle before the start of the transfer and goes back high at the end of the eighth bit time of the transfer. The $\overline{SS}$ IN waveform applies to the slave select input of a slave.

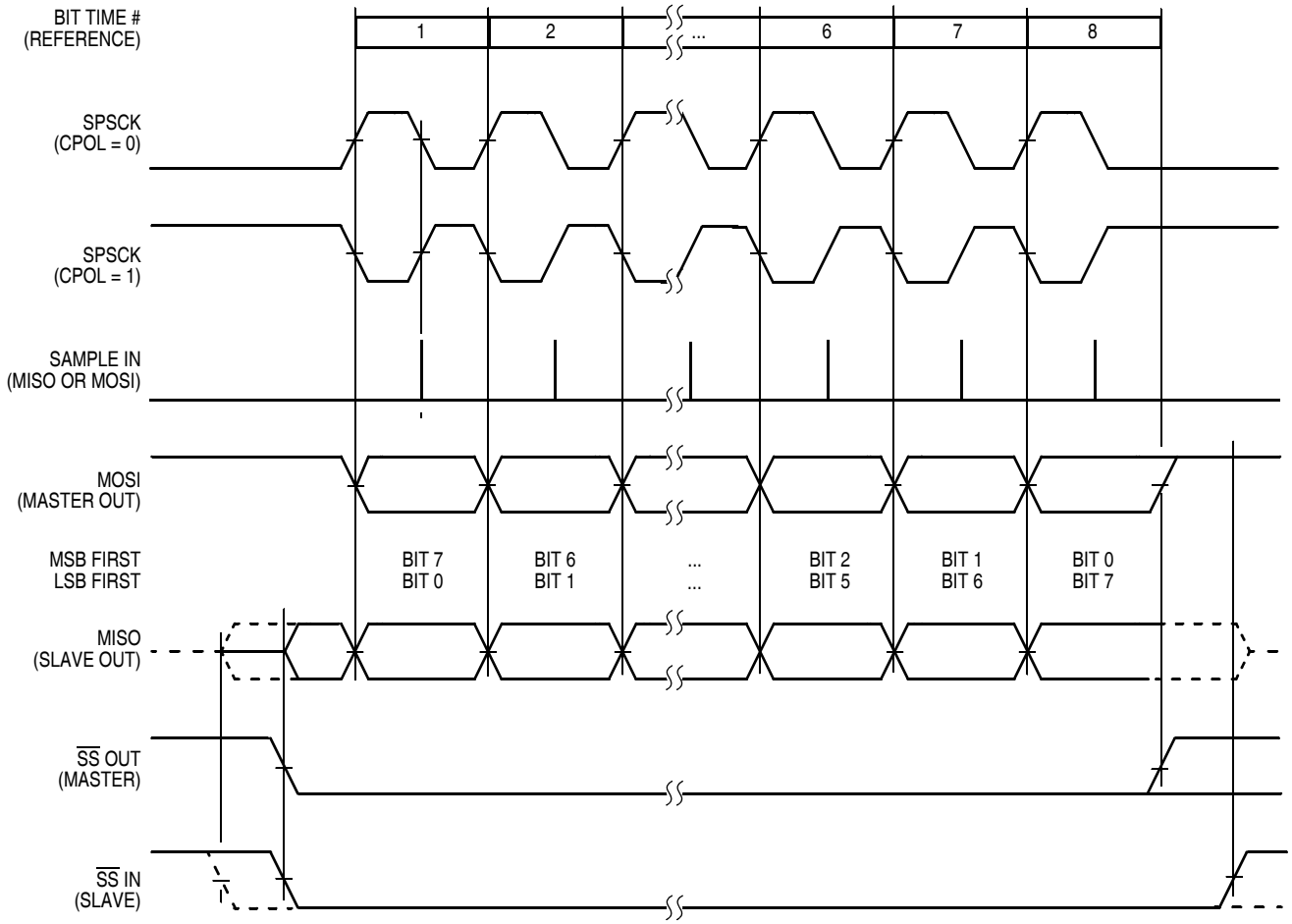


Figure 13-16. SPI Clock Formats (CPHA = 1)

When CPHA = 1, the slave begins to drive its MISO output when $\overline{SS}$ goes to active low, but the data is not defined until the first SPSCK edge. The first SPSCK edge shifts the first bit of data from the shifter onto the MOSI output of the master and the MISO output of the slave. The next SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled, and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CPHA = 1, the slave's $\overline{SS}$ input is not required to go to its inactive high level between transfers.

Figure 13-17 shows the clock formats when SPIMODE = 0 and CPHA = 0. At the top of the figure, the eight bit times are shown for reference with bit 1 starting as the slave is selected ($\overline{SS}$ IN goes low), and bit 8 ends at the last SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the

MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{SS}$ OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master $\overline{SS}$ output goes to active low at the start of the first bit time of the transfer and goes back high one-half SPSCK cycle after the end of the eighth bit time of the transfer. The $\overline{SS}$ IN waveform applies to the slave select input of a slave.

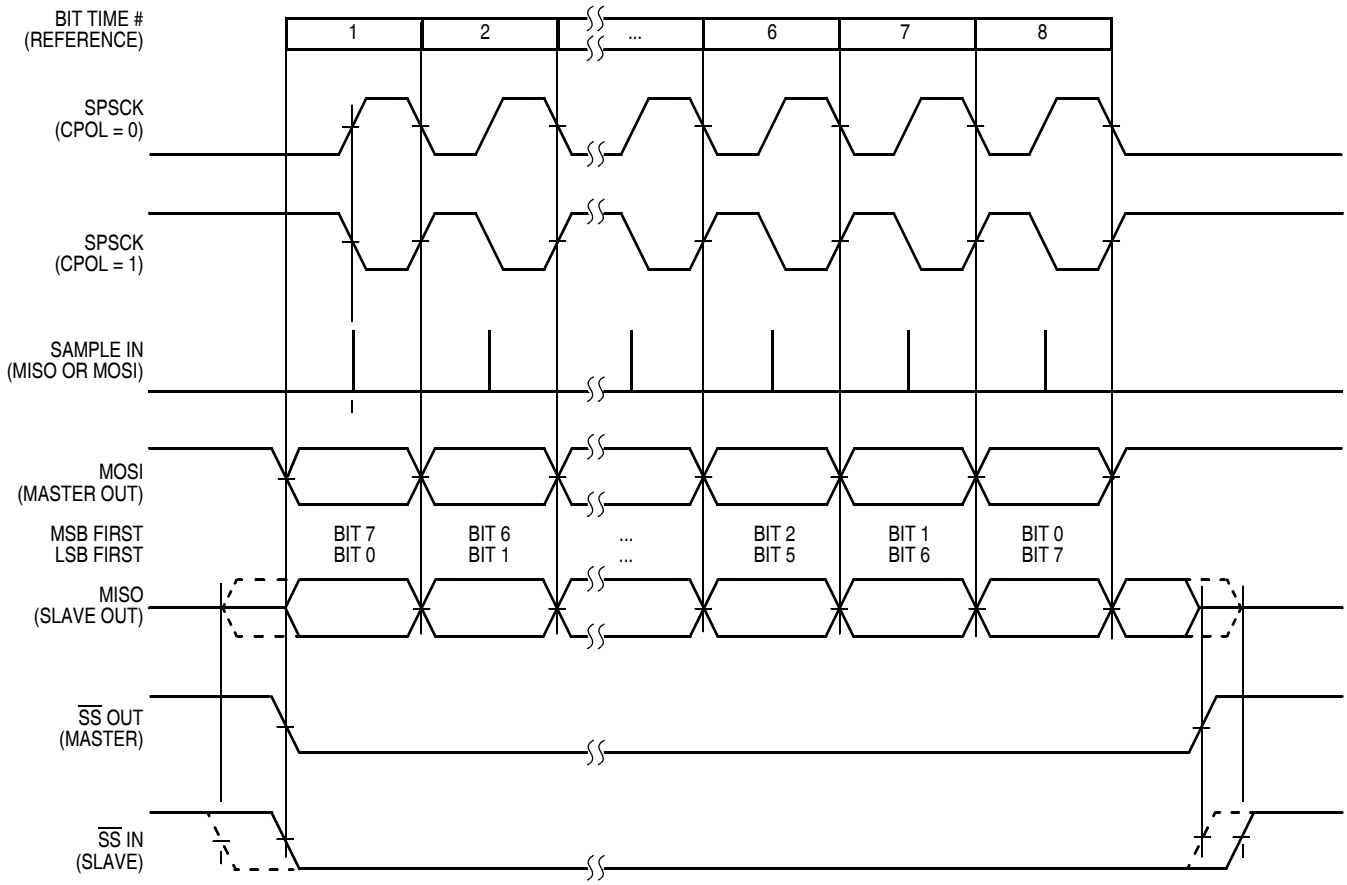


Figure 13-17. SPI Clock Formats (CPHA = 0)

When CPHA = 0, the slave begins to drive its MISO output with the first data bit value (MSB or LSB depending on LSBFE) when $\overline{SS}$ goes to active low. The first SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the second SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CPHA = 0, the slave's $\overline{SS}$ input must go to its inactive high level between transfers.

13.4.7 SPI Baud Rate Generation

As shown in Figure 13-18, the clock source for the SPI baud rate generator is the bus clock. The three prescale bits (SPPR2:SPPR1:SPPR0) choose a prescale divisor of 1, 2, 3, 4, 5, 6, 7, or 8. The three rate

select bits (SPR3:SPR2:SPR1:SPR0) divide the output of the prescaler stage by 2, 4, 8, 16, 32, 64, 128, 256 or 512 to get the internal SPI master mode bit-rate clock.

The baud rate generator is activated only when the SPI is in the master mode and a serial transfer is taking place. In the other cases, the divider is disabled to decrease I_{DD} current.

The baud rate divisor equation is as follows except those reserved combinations in Table 13-7 :

$$BaudRateDivisor = (SPPR + 1) \bullet 2^{(SPR + 1)}$$

The baud rate can be calculated with the following equation:

$$Baud\ Rate = BusClock / BaudRateDivisor$$

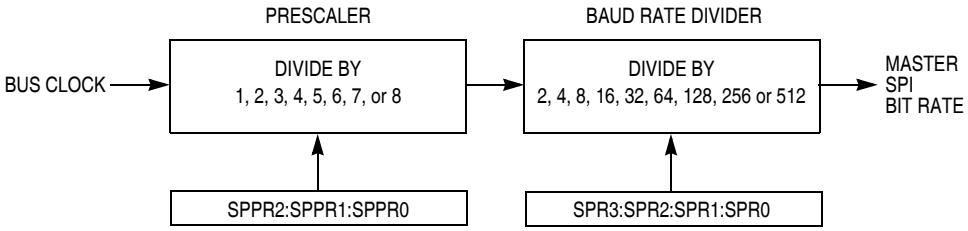


Figure 13-18. SPI Baud Rate Generation

13.4.8 Special Features

13.4.8.1 $\overline{SS}$ Output

The $\overline{SS}$ output feature automatically drives the $\overline{SS}$ pin low during transmission to select external devices and drives it high during idle to deselect external devices. When $\overline{SS}$ output is selected, the $\overline{SS}$ output pin is connected to the $\overline{SS}$ input pin of the external device.

The $\overline{SS}$ output is available only in master mode during normal SPI operation by asserting the SSOE and MODFEN bits as shown in Table 13-2.

The mode fault feature is disabled while $\overline{SS}$ output is enabled.

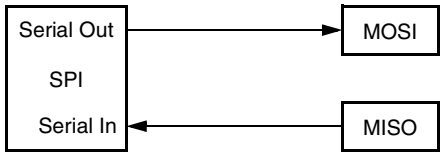
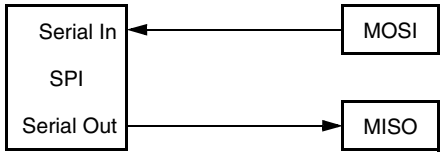
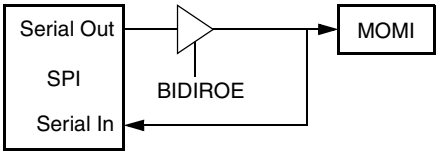
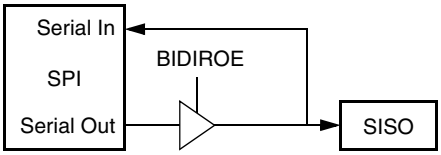
NOTE

Care must be taken when using the $\overline{SS}$ output feature in a multi-master system since the mode fault feature is not available for detecting system errors between masters.

13.4.8.2 Bidirectional Mode (MOMI or SISO)

The bidirectional mode is selected when the SPC0 bit is set in SPI Control Register 2 (see Table 13-11). 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 13-11. Normal Mode and Bidirectional Mode

When SPE = 1	Master Mode MSTR = 1	Slave Mode MSTR = 0
Normal Mode SPC0 = 0		
Bidirectional Mode SPC0 = 1		

The direction of each serial I/O pin depends on the BIDIROE bit. If the pin is configured as an output, serial data from the shift register is driven out on the pin. The same pin is also the serial input to the shift register.

The SPSCK is output for the master mode and input for the slave mode.

The $\overline{SS}$ is the input or output for the master mode, and it is always the input for the slave mode.

The bidirectional mode does not affect SPSCK and $\overline{SS}$ functions.

NOTE

In bidirectional master mode, with 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 has to be considered, if the MISO pin is used for another purpose.

13.4.9 Error Conditions

The SPI has one error condition:

- Mode fault error

13.4.9.1 Mode Fault Error

If the $\overline{SS}$ input becomes low while the SPI is configured as a master, it indicates a system error where more than one master may be trying to drive the MOSI and SPSCK lines simultaneously. This condition is not

permitted in normal operation, and 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 SPSCK, MISO and MOSI pins are forced to be high impedance inputs to avoid any possibility of conflict with another output driver. A transmission in progress is aborted and the SPI is forced into idle state.

If the mode fault error occurs in the bidirectional mode for 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 the SPI system configured in slave mode.

The mode fault flag is cleared automatically by a read of the SPI Status Register (with MODF set) followed by a write to SPI Control Register 1. If the mode fault flag is cleared, the SPI becomes a normal master or slave again.

13.4.10 Low Power Mode Options

13.4.10.1 SPI in Run Mode

In run mode with the SPI system enable (SPE) bit in the SPI control register clear, the SPI system is in a low-power, disabled state. SPI registers can still be accessed, but clocks to the core of this module are disabled.

13.4.10.2 SPI in Wait Mode

SPI operation in wait mode depends upon the state of the SPISWAI bit in SPI Control Register 2.

- If SPISWAI is clear, the SPI operates normally when the CPU is in wait mode
- If SPISWAI is set, SPI clock generation ceases and the SPI module enters a power conservation state when the CPU is in wait mode.
  - — If SPISWAI is set and the SPI is configured for master, any transmission and reception in progress stops at wait mode entry. The transmission and reception resumes when the SPI exits wait mode.
  - – If SPISWAI is set and the SPI is configured as a slave, any transmission and reception in progress continues if the SPSCK continues to be driven from the master. This keeps the slave synchronized to the master and the SPSCK.

    If the master transmits data while the slave is in wait mode, the slave will continue to send out data consistent with the operation mode at the start of wait mode (i.e., if the slave is currently sending its SPIxDH:SPIxDL to the master, it will continue to send the same byte. Otherwise, if the slave is currently sending the last data received byte from the master, it will continue to send each previously receive data from the master byte).

NOTE

Care must be taken when expecting data from a master while the slave is in wait or stop3 mode. Even though the shift register will continue to operate, the rest of the SPI is shut down (i.e. a SPRF interrupt will not be generated until exiting stop or wait mode). Also, the data from the shift register will not be copied into the SPIxDH:SPIxDL registers until after the slave SPI has exited wait or stop mode. A SPRF flag and SPIxDH:SPIxDL copy is only generated 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 SPRF nor a SPIxDH:SPIxDL copy will occur.

13.4.10.3 SPI in Stop Mode

Stop3 mode is dependent on the SPI system. Upon entry to stop3 mode, the SPI module clock is disabled (held high or low). If the SPI is in master mode and exchanging data when the CPU enters 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.

In all other stop modes, the SPI module is completely disabled. After stop, all registers are reset to their default values, and the SPI module must be re-initialized.

13.4.10.4 Reset

The reset values of registers and signals are described in Section 13.3, “Register Definition.” which details the registers and their bit-fields.

- If a data transmission occurs in slave mode after reset without a write to SPIxDH:SPIxDL, it will transmit garbage, or the data last received from the master before the reset.
- Reading from the SPIxDH:SPIxDL after reset will always read zeros.

13.4.10.5 Interrupts

The SPI only originates interrupt requests when the SPI is enabled (SPE bit in SPIxC1 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.

13.4.11 SPI Interrupts

There are four flag bits, three interrupt mask bits, and one interrupt vector associated with the SPI system. The SPI interrupt enable mask (SPIE) enables interrupts from the SPI receiver full flag (SPRF) and mode fault flag (MODF). The SPI transmit interrupt enable mask (SPTIE) enables interrupts from the SPI transmit buffer empty flag (SPTEF). The SPI match interrupt enable mask bit (SPIMIE) enables interrupts from the SPI match flag (SPMF). When one of the flag bits is set, and the associated interrupt mask bit is set, a hardware interrupt request is sent to the CPU. If the interrupt mask bits are cleared, software can poll the associated flag bits instead of using interrupts. The SPI interrupt service routine (ISR) should check

the flag bits to determine what event caused the interrupt. The service routine should also clear the flag bit(s) before returning from the ISR (usually near the beginning of the ISR).

13.4.11.1 MODF

MODF occurs when the master detects an error on the $\overline{SS}$ pin. The master SPI must be configured for the MODF feature (see Table 13-2). Once MODF is set, the current transfer is aborted and the following bit is changed:

- MSTR=0, The master bit in SPIxC1 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 13.3.4, “SPI Status Register (SPIxS).”

13.4.11.2 SPRF

SPRF occurs when new data has been received and copied to the SPI receive data buffer. In 8-bit mode, SPRF is set only after all 8 bits have been shifted out of the shift register and into SPIxDL. In 16-bit mode, SPRF is set only after all 16 bits have been shifted out of the shift register and into SPIxDH:SPIxDL.

Once SPRF is set, it does not clear until it is serviced. SPRF has an automatic clearing process which is described in Section 13.3.4, “SPI Status Register (SPIxS).” In the event that the SPRF is not serviced before the end of the next transfer (i.e. SPRF remains active throughout another transfer), the latter transfers will be ignored and no new data will be copied into the SPIxDH:SPIxDL.

13.4.11.3 SPTEF

SPTEF occurs when the SPI transmit buffer is ready to accept new data. In 8-bit mode, SPTEF is set only after all 8 bits have been moved from SPIxDL into the shifter. In 16-bit mode, SPTEF is set only after all 16 bits have been moved from SPIxDH:SPIxDL into the shifter.

Once SPTEF is set, it does not clear until it is serviced. SPTEF has an automatic clearing process which is described in Section 13.3.4, “SPI Status Register (SPIxS).”

13.4.11.4 SPMF

SPMF occurs when the data in the receive data buffer is equal to the data in the SPI match register. In 8-bit mode, SPMF is set only after bits 8–0 in the receive data buffer are determined to be equivalent to the value in SPIxML. In 16-bit mode, SPMF is set after bits 15–0 in the receive data buffer are determined to be equivalent to the value in SPIxMH:SPIxML.

13.4.11.5 TNEAREF

TNEAREF flag is set when only one 16-bit word or 2 8-bit bytes of data remain in the transmit FIFO provided SPIxC3[5] = 0 or when only two 16bit word or 4 8-bit bytes of data remain in the transmit FIFO provided SPIxC3[5] =1. If FIFOMODE is not enabled this bit should be ignored.

Clearing of this interrupts depends on state of SPIxC3[3] and the status of TNEAREF as described Section 13.3.4, “SPI Status Register (SPIxS).”

13.4.11.6 RNFULLF

RNFULLF is set when more than three16bit word or six 8bit bytes of data remain in the receive FIFO provided SPIxC3[4] = 0 or when more than two 16bit word or four 8bit bytes of data remain in the receive FIFO provided SPIxC3[4] = 1.

Clearing of this interrupts depends on state of SPIxC3[3] and the status of RNFULLF as described Section 13.3.4, “SPI Status Register (SPIxS).”

13.5 Initialization/Application Information

13.5.1 SPI Module Initialization Example

13.5.1.1 Initialization Sequence

Before the SPI module can be used for communication, an initialization procedure must be carried out, as follows:

1. Update control register 1 (SPIxC1) to enable the SPI and to control interrupt enables. This register also sets the SPI as master or slave, determines clock phase and polarity, and configures the main SPI options.
2. Update control register 2 (SPIxC2) to enable additional SPI functions such as the SPI match interrupt feature, the master mode-fault function, and bidirectional mode output. 8- or 16-bit mode select and other optional features are controlled here as well.
3. Update the baud rate register (SPIxBR) to set the prescaler and bit rate divisor for an SPI master.
4. Update the hardware match register (SPIxMH:SPIxML) with the value to be compared to the receive data register for triggering an interrupt if hardware match interrupts are enabled.
5. In the master, read SPIxS while SPTEF = 1, and then write to the transmit data register (SPIxDH:SPIxDL) to begin transfer.

13.5.1.2 Pseudo—Code Example

In this example, the SPI module will be set up for master mode with only hardware match interrupts enabled. The SPI will run in 16-bit mode at a maximum baud rate of bus clock divided by 2. Clock phase and polarity will be set for an active-high SPI clock where the first edge on SPSCK occurs at the start of the first cycle of a data transfer.

SPIxC1=0x54(%01010100)

Bit 7	SPIE	= 0	Disables receive and mode fault interrupts
Bit 6	SPE	= 1	Enables the SPI system
Bit 5	SPTIE	= 0	Disables SPI transmit interrupts
Bit 4	MSTR	= 1	Sets the SPI module as a master SPI device
Bit 3	CPOL	= 0	Configures SPI clock as active-high
Bit 2	CPHA	= 1	First edge on SPSCK at start of first data transfer cycle
Bit 1	SSOE	= 0	Determines $\overline{SS}$ pin function when mode fault enabled
Bit 0	LSBFE	= 0	SPI serial data transfers start with most significant bit

SPIxC2 = 0xC0(%11000000)

Bit 7	SPMIE	= 1	SPI hardware match interrupt enabled
Bit 6	SPIMODE	= 1	Configures SPI for 16-bit mode
Bit 5		= 0	Unimplemented
Bit 4	MODFEN	= 0	Disables mode fault function
Bit 3	BIDIROE	= 0	SPI data I/O pin acts as input
Bit 2		= 0	Unimplemented
Bit 1	SPISWAI	= 0	SPI clocks operate in wait mode
Bit 0	SPC0	= 0	uses separate pins for data input and output

SPIxBR = 0x00(%00000000)

Bit 7	= 0	Unimplemented
Bit 6:4	= 000	Sets prescale divisor to 1
Bit 3:0	= 0000	Sets baud rate divisor to 2

SPIxS = 0x00(%00000000)

Bit 7	SPRF	= 0	Flag is set when receive data buffer is full
Bit 6	SPMF	= 0	Flag is set when SPIMH/L = receive data buffer
Bit 5	SPTEF	= 0	Flag is set when transmit data buffer is empty
Bit 4	MODF	= 0	Mode fault flag for master mode
Bit 3:0		= 0	Unimplemented

SPIxMH = 0xXX

In 16-bit mode, this register holds bits 8–15 of the hardware match buffer. In 8-bit mode, writes to this register will be ignored.

SPIxML = 0xXX

Holds bits 0–7 of the hardware match buffer.

SPIxDH = 0xxx

In 16-bit mode, this register holds bits 8–15 of the data to be transmitted by the transmit buffer and received by the receive buffer.

SPIxDL = 0xxx

Holds bits 0–7 of the data to be transmitted by the transmit buffer and received by the receive buffer.

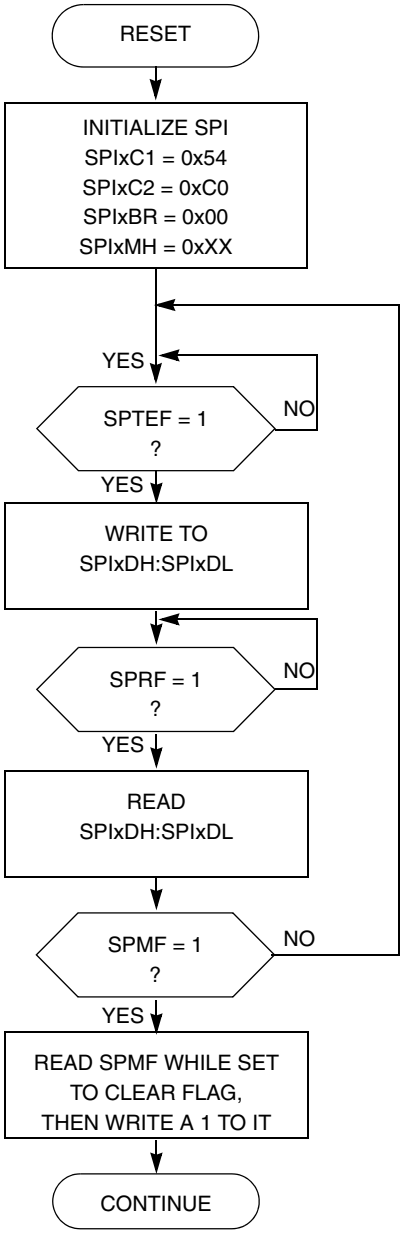


Figure 13-19. Initialization Flowchart Example for SPI Master Device in 16-Bit Mode for FIFOMODE = 0

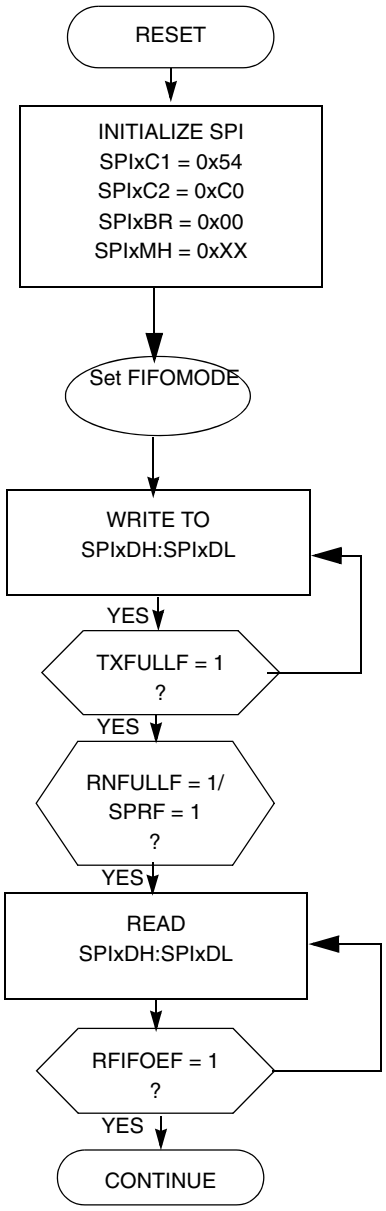


Figure 13-20. Initialization Flowchart Example for SPI Master Device in 16-Bit Mode for FIFOMODE

Chapter 14
Serial Communication Interface (SCI)

14.1 Introduction

The SCI allows asynchronous serial communications with peripheral devices and other CPUs.

NOTE

The SCI module connects to the TPM and PRACMP in order to facilitate IR communication modulation/demodulation. For more information, refer to Section 2.2.1, “Interfacing the SCIs to Off-Chip Circuits.”

NOTE

Ignore any references to stop1 low-power mode in this chapter, because this device does not support it.

For details on low-power mode operation, refer to Table 6-4 in Chapter 6, “Modes of Operation”.

14.1.1 Module Block Diagram

Figure 14-1 shows a block diagram of the SCI module.

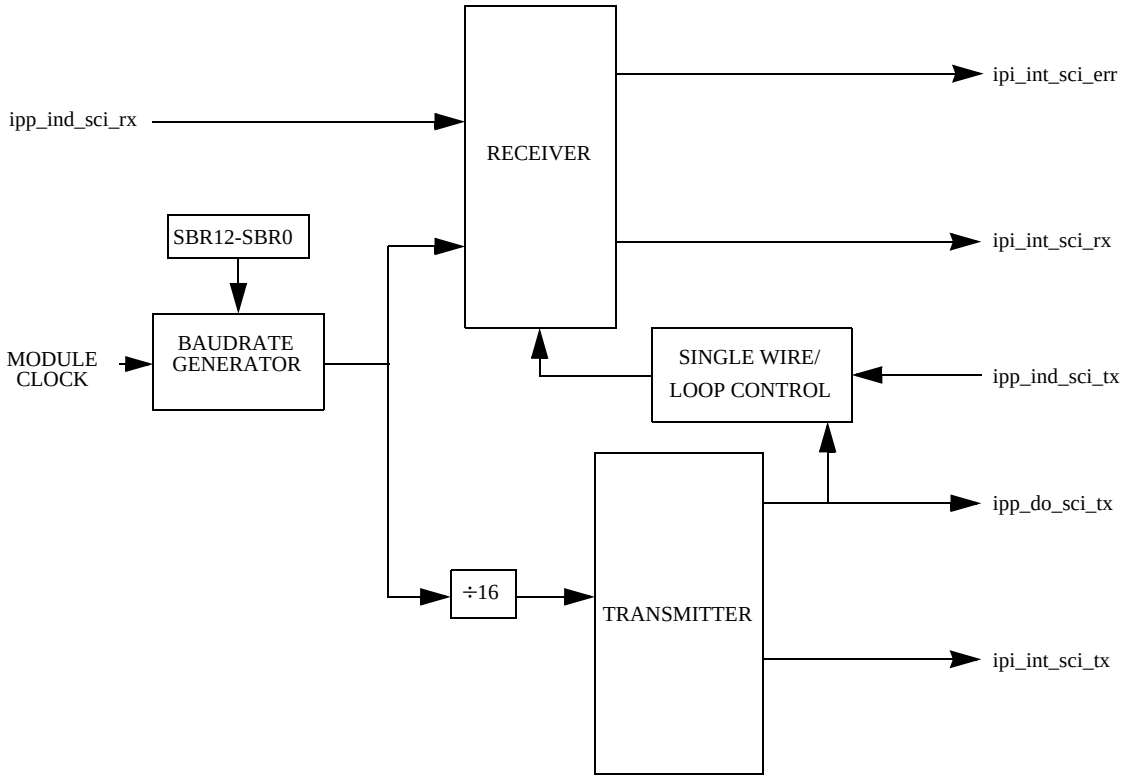


Figure 14-1. SCI Module Block Diagram

14.1.2 Features

Features of SCI module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Double-buffered transmitter and receiver with separate enables
- Programmable baud rates (13-bit modulo divider)
- Interrupt-driven or polled operation:
  - Transmit data register empty and transmission complete
  - Receive data register full
  - Receive overrun, parity error, framing error, and noise error
  - Idle receiver detect
  - Active edge on receive pin
  - Break detect supporting LIN
- Hardware parity generation and checking
- Programmable 8-bit or 9-bit character length
- Receiver wakeup by idle-line or address-mark
- Optional 13-bit break character generation / 11-bit break character detection
- Selectable transmitter output polarity

14.1.3 Modes of Operation

See Section 14.3, “Functional Description,” for details concerning SCI operation in these modes:

- 8- and 9-bit data modes
- Stop mode operation
- Loop mode
- Single-wire mode

14.1.4 Block Diagram

Figure 14-2 shows the transmitter portion of the SCI.

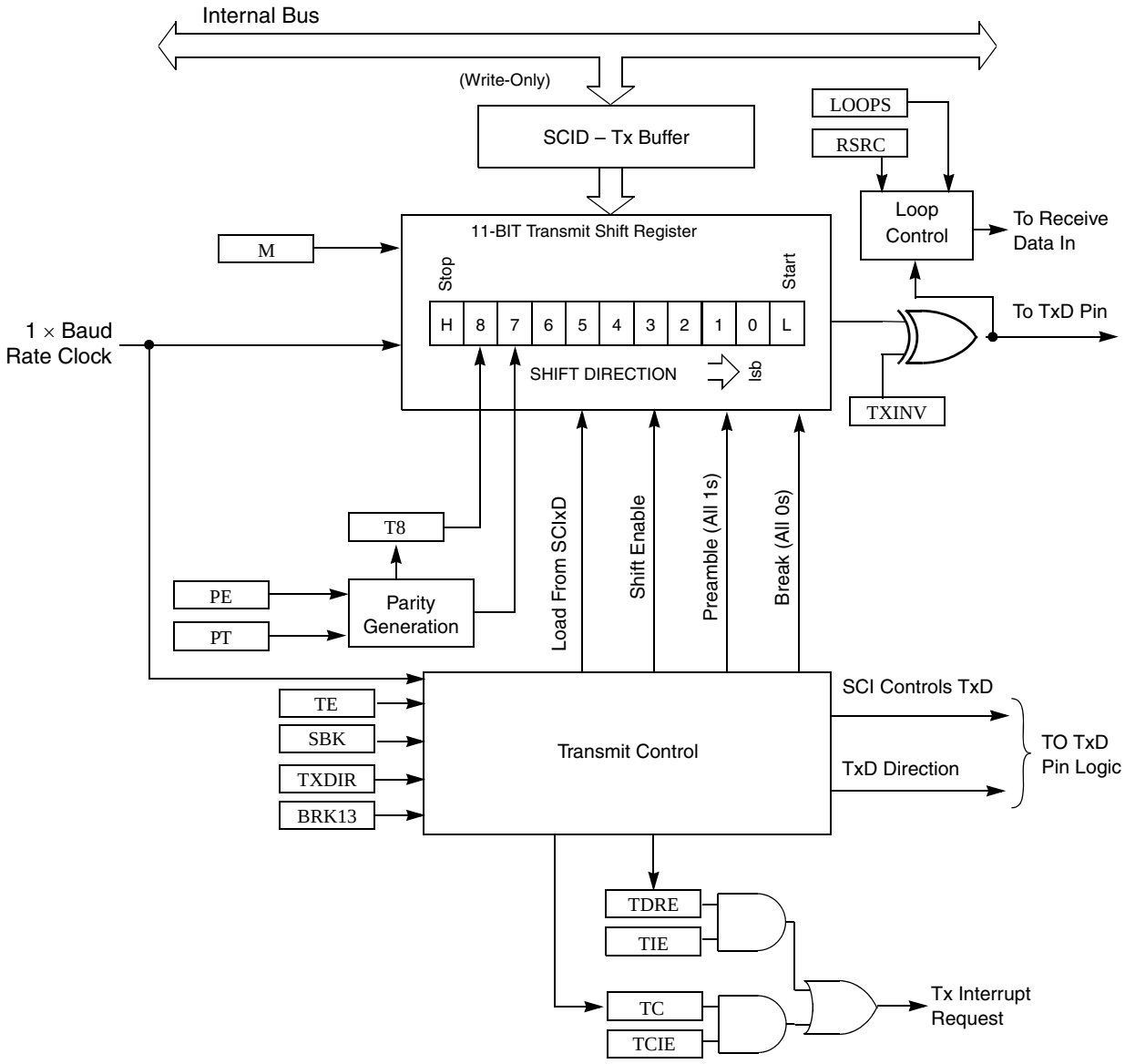


Figure 14-2. SCI Transmitter Block Diagram

Figure 14-3 shows the receiver portion of the SCI.

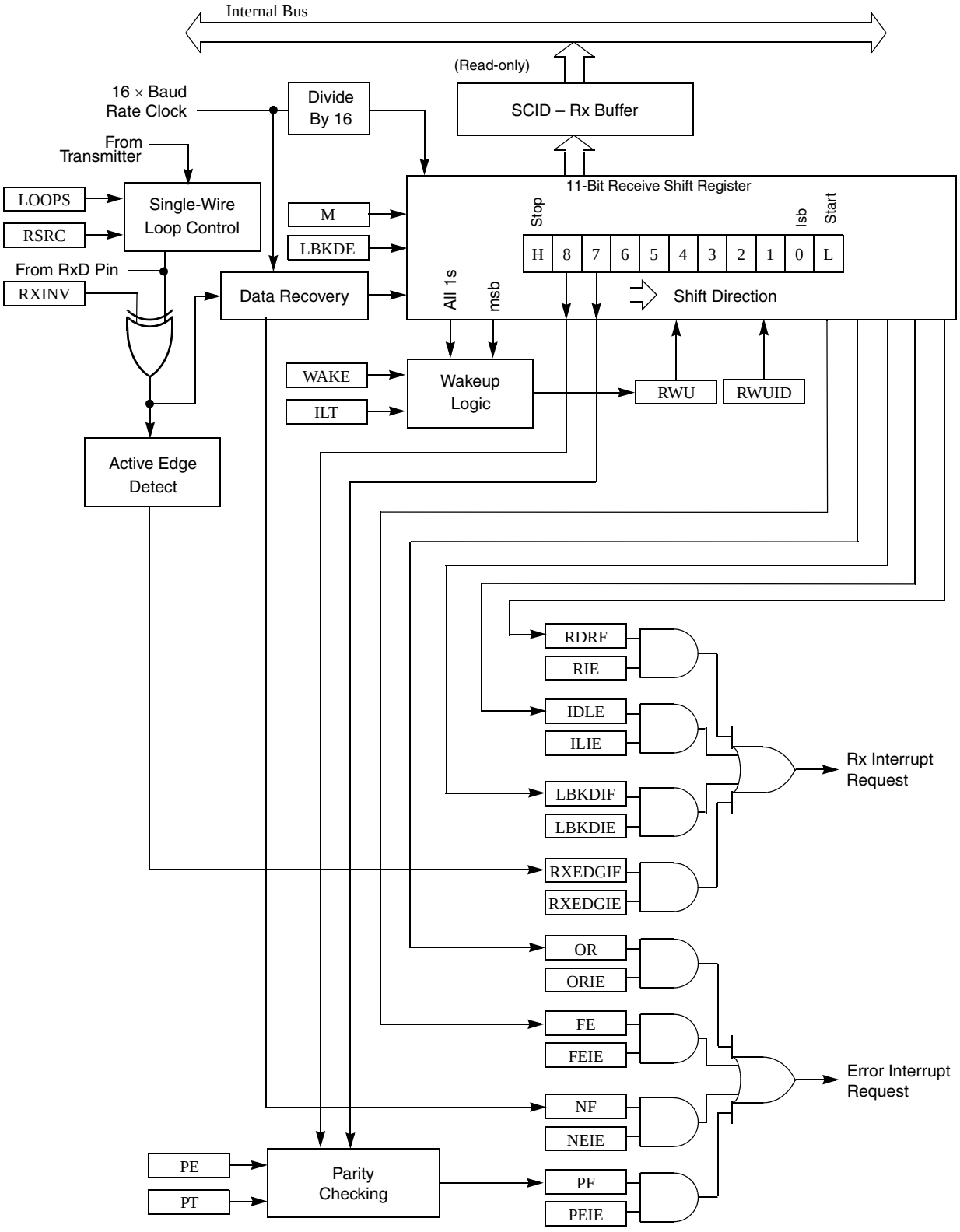


Figure 14-3. SCI Receiver Block Diagram

14.2 Register Definition

The SCI has eight 8-bit registers to control baud rate, select SCI options, report SCI status, and for transmit/receive data.

Refer to the direct-page register summary in the memory chapter of this document or the absolute address assignments for all SCI registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

14.2.1 SCI Baud Rate Registers (SCIxBDH, SCIxBDL)

This pair of registers controls the prescale divisor for SCI baud rate generation. To update the 13-bit baud rate setting [SBR12:SBR0], first write to SCIxBDH to buffer the high half of the new value and then write to SCIxBDL. The working value in SCIxBDH does not change until SCIxBDL is written.

SCIxBDL is reset to a non-zero value, so after reset the baud rate generator remains disabled until the first time the receiver or transmitter is enabled (RE or TE bits in SCIxC2 are written to 1).

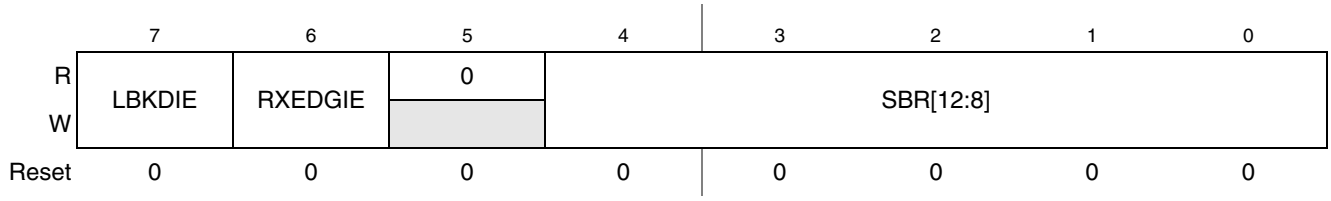


Figure 14-4. SCI Baud Rate Register (SCIxBDH)

Table 14-1. SCIxBDH Field Descriptions

Field	Description
7 LBKDIE	LIN Break Detect Interrupt Enable (for LBKDIF) 0 Hardware interrupts from LBKDIF disabled (use polling). 1 Hardware interrupt requested when LBKDIF flag is 1.
6 RXEDGIE	RxD Input Active Edge Interrupt Enable (for RXEDGIF) 0 Hardware interrupts from RXEDGIF disabled (use polling). 1 Hardware interrupt requested when RXEDGIF flag is 1.
4–0 SBR[12:8]	Baud Rate Modulo Divisor. The 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR is cleared, the SCI baud rate generator is disabled to reduce supply current. When BR is 1 – 8191, the SCI baud rate equals BUSCLK/(16×BR). See also BR bits in Table 14-2.

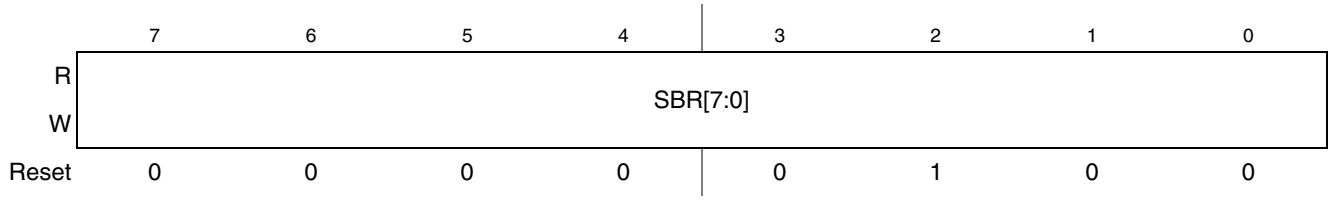


Figure 14-5. SCI Baud Rate Register (SCIxBDL)

Table 14-2. SCIxBDL Field Descriptions

Field	Description
7–0 SBR[7:0]	Baud Rate Modulo Divisor. These 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR is cleared, the SCI baud rate generator is disabled to reduce supply current. When BR is 1 – 8191, the SCI baud rate equals BUSCLK/(16×BR). See also BR bits in Table 14-1.

14.2.2 SCI Control Register 1 (SCIxC1)

This read/write register controls various optional features of the SCI system.

	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 14-6. SCI Control Register 1 (SCIxC1)

Table 14-3. SCIxC1 Field Descriptions

Field	Description
7 LOOPS	Loop Mode Select. Selects between loop back modes and normal 2-pin full-duplex modes. When LOOPS is set, the transmitter output is internally connected to the receiver input. 0 Normal operation — RxD and TxD use separate pins. 1 Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input. (See RSRC bit.) RxD pin is not used by SCI.
6 SCISWAI	SCI Stops in Wait Mode 0 SCI clocks continue to run in wait mode so the SCI can be the source of an interrupt that wakes up the CPU. 1 SCI clocks freeze while CPU is in wait mode.
5 RSRC	Receiver Source Select. This bit has no meaning or effect unless the LOOPS bit is set to 1. When LOOPS is set, the receiver input is internally connected to the TxD pin and RSRC determines whether this connection is also connected to the transmitter output. 0 Provided LOOPS is set, RSRC is cleared, selects internal loop back mode and the SCI does not use the RxD pins. 1 Single-wire SCI mode where the TxD pin is connected to the transmitter output and receiver input.
4 M	9-Bit or 8-Bit Mode Select 0 Normal — start + 8 data bits (lsb first) + stop. 1 Receiver and transmitter use 9-bit data characters start + 8 data bits (lsb first) + 9th data bit + stop.
3 WAKE	Receiver Wakeup Method Select. Refer to Section 14.3.3.2, “Receiver Wakeup Operation” for more information. 0 Idle-line wakeup. 1 Address-mark wakeup.
2 ILT	Idle Line Type Select. Setting this bit to 1 ensures that the stop bit and logic 1 bits at the end of a character do not count toward the 10 or 11 bit times of logic high level needed by the idle line detection logic. Refer to Section 14.3.3.2.1, “Idle-Line Wakeup” for more information. 0 Idle character bit count starts after start bit. 1 Idle character bit count starts after stop bit.

Table 14-3. SCIxC1 Field Descriptions (continued)

Field	Description
1 PE	Parity Enable. Enables hardware parity generation and checking. When parity is enabled, the most significant bit (msb) of the data character (eighth or ninth data bit) is treated as the parity bit. 0 No hardware parity generation or checking. 1 Parity enabled.
0 PT	Parity Type. Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even. 0 Even parity. 1 Odd parity.

14.2.3 SCI Control Register 2 (SCIxC2)

This register can be read or written at any time.

	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 14-7. SCI Control Register 2 (SCIxC2)

Table 14-4. SCIxC2 Field Descriptions

Field	Description
7 TIE	Transmit Interrupt Enable (for TDRE) 0 Hardware interrupts from TDRE disabled (use polling). 1 Hardware interrupt requested when TDRE flag is 1.
6 TCIE	Transmission Complete Interrupt Enable (for TC) 0 Hardware interrupts from TC disabled (use polling). 1 Hardware interrupt requested when TC flag is 1.
5 RIE	Receiver Interrupt Enable (for RDRF) 0 Hardware interrupts from RDRF disabled (use polling). 1 Hardware interrupt requested when RDRF flag is 1.
4 ILIE	Idle Line Interrupt Enable (for IDLE) 0 Hardware interrupts from IDLE disabled (use polling). 1 Hardware interrupt requested when IDLE flag is 1.

Table 14-4. SCIxC2 Field Descriptions (continued)

Field	Description
3 TE	Transmitter Enable 0 Transmitter off. 1 Transmitter on. TE must be 1 to use the SCI transmitter. When TE is set, the SCI forces the TxD pin to act as an output for the SCI system. When the SCI is configured for single-wire operation (LOOPS = RSRC = 1), TXDIR controls the direction of traffic on the single SCI communication line (TxD pin). TE can also queue an idle character by clearing TE then setting TE while a transmission is in progress. Refer to Section 14.3.2.1, “Send Break and Queued Idle” for more details. When TE is written to 0, the transmitter keeps control of the port TxD pin until any data, queued idle, or queued break character finishes transmitting before allowing the pin to revert to a general-purpose I/O pin.
2 RE	Receiver Enable. When the SCI receiver is off, the RxD pin reverts to being a general-purpose port I/O pin. If LOOPS is set the RxD pin reverts to being a general-purpose I/O pin even if RE is set. 0 Receiver off. 1 Receiver on.
1 RWU	Receiver Wakeup Control. This bit can be written to 1 to place the SCI receiver in a standby state where it waits for automatic hardware detection of a selected wakeup condition. The wakeup condition is an idle line between messages (WAKE = 0, idle-line wakeup) or a logic 1 in the most significant data bit in a character (WAKE = 1, address-mark wakeup). Application software sets RWU and (normally) a selected hardware condition automatically clears RWU. Refer to Section 14.3.3.2, “Receiver Wakeup Operation,” for more details. 0 Normal SCI receiver operation. 1 SCI receiver in standby waiting for wakeup condition.
0 SBK	Send Break. Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 10 or 11 (13 or 14 if BRK13 = 1) bit times of logic 0 are queued as long as SBK is set. Depending on the timing of the set and clear of SBK relative to the information currently being transmitted, a second break character may be queued before software clears SBK. Refer to Section 14.3.2.1, “Send Break and Queued Idle” for more details. 0 Normal transmitter operation. 1 Queue break character(s) to be sent.

14.2.4 SCI Status Register 1 (SCIxS1)

This register has eight read-only status flags. Writes have no effect. Special software sequences (which do not involve writing to this register) clear these status flags.

	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

Figure 14-8. SCI Status Register 1 (SCIxS1)

Table 14-5. SCIxS1 Field Descriptions

Field	Description
7 TDRE	Transmit Data Register Empty Flag. TDRE is set out of reset and when a transmit data value transfers from the transmit data buffer to the transmit shifter, leaving room for a new character in the buffer. To clear TDRE, read SCIxS1 with TDRE set and then write to the SCI data register (SCIxD). 0 Transmit data register (buffer) full. 1 Transmit data register (buffer) empty.
6 TC	Transmission Complete Flag. TC is set out of reset and when TDRE is set and no data, preamble, or break character is being transmitted. 0 Transmitter active (sending data, a preamble, or a break). 1 Transmitter idle (transmission activity complete). TC is cleared automatically by reading SCIxS1 with TC set and then doing one of the following: • Write to the SCI data register (SCIxD) to transmit new data • Queue a preamble by changing TE from 0 to 1 • Queue a break character by writing 1 to SBK in SCIxC2
5 RDRF	Receive Data Register Full Flag. RDRF becomes set when a character transfers from the receive shifter into the receive data register (SCIxD). To clear RDRF, read SCIxS1 with RDRF set and then read the SCI data register (SCIxD). 0 Receive data register empty. 1 Receive data register full.
4 IDLE	Idle Line Flag. IDLE is set when the SCI receive line becomes idle for a full character time after a period of activity. When ILT is cleared, the receiver starts counting idle bit times after the start bit. If the receive character is all 1s, these bit times and the stop bit time count toward the full character time of logic high (10 or 11 bit times depending on the M control bit) needed for the receiver to detect an idle line. When ILT is set, the receiver doesn't start counting idle bit times until after the stop bit. The stop bit and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line. To clear IDLE, read SCIxS1 with IDLE set and then read the SCI data register (SCIxD). After IDLE has been cleared, it cannot become set again until after a new character has been received and RDRF has been set. IDLE is set only once even if the receive line remains idle for an extended period. 0 No idle line detected. 1 Idle line was detected.
3 OR	Receiver Overrun Flag. OR is set when a new serial character is ready to be transferred to the receive data register (buffer), but the previously received character has not been read from SCIxD yet. In this case, the new character (and all associated error information) is lost because there is no room to move it into SCIxD. To clear OR, read SCIxS1 with OR set and then read the SCI data register (SCIxD). 0 No overrun. 1 Receive overrun (new SCI data lost).
2 NF	Noise Flag. The advanced sampling technique used in the receiver takes seven samples during the start bit and three samples in each data bit and the stop bit. If any of these samples disagrees with the rest of the samples within any bit time in the frame, the flag NF is set at the same time as RDRF is set for the character. To clear NF, read SCIxS1 and then read the SCI data register (SCIxD). 0 No noise detected. 1 Noise detected in the received character in SCIxD.

Table 14-5. SCIxS1 Field Descriptions (continued)

Field	Description
1 FE	Framing Error Flag. FE is set at the same time as RDRF when the receiver detects a logic 0 where the stop bit was expected. This suggests the receiver was not properly aligned to a character frame. To clear FE, read SCIxS1 with FE set and then read the SCI data register (SCIxD). 0 No framing error detected. This does not guarantee the framing is correct. 1 Framing error.
0 PF	Parity Error Flag. PF is set at the same time as RDRF when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, read SCIxS1 and then read the SCI data register (SCIxD). 0 No parity error. 1 Parity error.

14.2.5 SCI Status Register 2 (SCIxS2)

This register contains one read-only status flag.

	7	6	5	4	3	2	1	0
R	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
W								
Reset	0	0	0	0	0	0	0	0

Figure 14-9. SCI Status Register 2 (SCIxS2)

Table 14-6. SCIxS2 Field Descriptions

Field	Description
7 LBKDIF	LIN Break Detect Interrupt Flag. LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a 1 to it. 0 No LIN break character has been detected. 1 LIN break character has been detected.
6 RXEDGIF	RxD Pin Active Edge Interrupt Flag. RXEDGIF is set when an active edge (falling if RXINV = 0, rising if RXINV=1) on the RxD pin occurs. RXEDGIF is cleared by writing a 1 to it. 0 No active edge on the receive pin has occurred. 1 An active edge on the receive pin has occurred.
4 RXINV[1]	Receive Data Inversion. Setting this bit reverses the polarity of the received data input. 0 Receive data not inverted 1 Receive data inverted
3 RWUID	Receive Wake Up Idle Detect. RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. 0 During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. 1 During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character.
2 BRK13	Break Character Generation Length. BRK13 selects a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. 0 Break character is transmitted with length of 10 bit times (11 if M = 1) 1 Break character is transmitted with length of 13 bit times (14 if M = 1)

Table 14-6. SCIxS2 Field Descriptions (continued)

Field	Description
1 LBKDE	LIN Break Detection Enable. LBKDE selects a longer break character detection length. While LBKDE is set, framing error (FE) and receive data register full (RDRF) flags are prevented from setting. 0 Break character is detected at length of 10 bit times (11 if M = 1). 1 Break character is detected at length of 11 bit times (12 if M = 1).
0 RAF	Receiver Active Flag. RAF is set when the SCI receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line. This status flag can be used to check whether an SCI character is being received before instructing the MCU to go to stop mode. 0 SCI receiver idle waiting for a start bit. 1 SCI receiver active (RxD input not idle).

[1] Setting RXINV inverts the RxD input for all cases: data bits, start and stop bits, break, and idle.

When using an internal oscillator in a LIN system, it is necessary to raise the break detection threshold one bit time. Under the worst case timing conditions allowed in LIN, it is possible that a 0x00 data character can appear to be 10.26 bit times long at a slave running 14% faster than the master. This would trigger normal break detection circuitry designed to detect a 10-bit break symbol. When the LBKDE bit is set, framing errors are inhibited and the break detection threshold changes from 10 bits to 11 bits, preventing false detection of a 0x00 data character as a LIN break symbol.

14.2.6 SCI Control Register 3 (SCIxC3)

	7	6	5	4	3	2	1	0
R	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
W								
Reset	0	0	0	0	0	0	0	0

Figure 14-10. SCI Control Register 3 (SCIxC3)

Table 14-7. SCIxC3 Field Descriptions

Field	Description
7 R8	Ninth Data Bit for Receiver. When the SCI is configured for 9-bit data (M = 1), R8 can be thought of as a ninth receive data bit to the left of the msb of the buffered data in the SCIxD register. When reading 9-bit data, read R8 before reading SCIxD because reading SCIxD completes automatic flag clearing sequences that could allow R8 and SCIxD to be overwritten with new data.
6 T8	Ninth Data Bit for Transmitter. When the SCI is configured for 9-bit data (M = 1), T8 may be thought of as a ninth transmit data bit to the left of the msb of the data in the SCIxD register. When writing 9-bit data, the entire 9-bit value is transferred to the SCI shift register after SCIxD is written so T8 should be written (if it needs to change from its previous value) before SCIxD is written. If T8 does not need to change in the new value (such as when it is used to generate mark or space parity), it need not be written each time SCIxD is written.
5 TXDIR	TxD Pin Direction in Single-Wire Mode. When the SCI is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TxD pin. 0 TxD pin is an input in single-wire mode. 1 TxD pin is an output in single-wire mode.

Table 14-7. SCIxC3 Field Descriptions (continued)

Field	Description
4 TXINV[1]	Transmit Data Inversion. Setting this bit reverses the polarity of the transmitted data output. 0 Transmit data not inverted 1 Transmit data inverted
3 ORIE	Overrun Interrupt Enable. This bit enables the overrun flag (OR) to generate hardware interrupt requests. 0 OR interrupts disabled (use polling). 1 Hardware interrupt requested when OR is set.
2 NEIE	Noise Error Interrupt Enable. This bit enables the noise flag (NF) to generate hardware interrupt requests. 0 NF interrupts disabled (use polling). 1 Hardware interrupt requested when NF is set.
1 FEIE	Framing Error Interrupt Enable. This bit enables the framing error flag (FE) to generate hardware interrupt requests. 0 FE interrupts disabled (use polling). 1 Hardware interrupt requested when FE is set.
0 PEIE	Parity Error Interrupt Enable. This bit enables the parity error flag (PF) to generate hardware interrupt requests. 0 PF interrupts disabled (use polling). 1 Hardware interrupt requested when PF is set.

[1] Setting TXINV inverts the TxD output for all cases: data bits, start and stop bits, break, and idle.

14.2.7 SCI Data Register (SCIxD)

This register is actually two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer. Reads and writes of this register are also involved in the automatic flag clearing mechanisms for the SCI status flags.

	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 14-11. SCI Data Register (SCIxD)

14.3 Functional Description

The SCI allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs. The SCI comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. During normal operation, the MCU monitors the status of the SCI, writes the data to be transmitted, and processes received data. The following describes each of the blocks of the SCI.

14.3.1 Baud Rate Generation

As shown in Figure 14-12, the clock source for the SCI baud rate generator is the bus-rate clock.

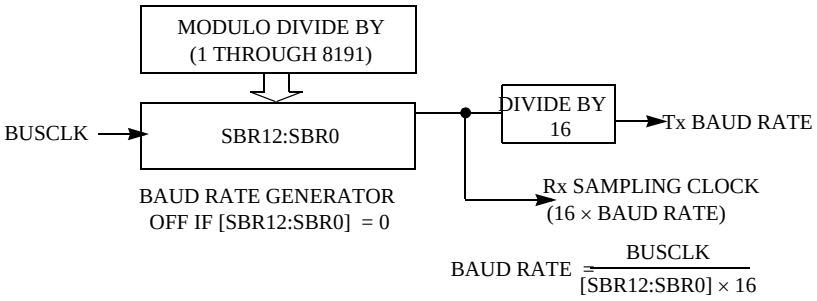


Figure 14-12. SCI Baud Rate Generation

SCI communications require the transmitter and receiver (which typically derive baud rates from independent clock sources) to use the same baud rate. Allowed tolerance on this baud frequency depends on the details of how the receiver synchronizes to the leading edge of the start bit and how bit sampling is performed.

The MCU resynchronizes to bit boundaries on every high-to-low transition. In the worst case, there are no such transitions in the full 10- or 11-bit time character frame so any mismatch in baud rate is accumulated for the whole character time. For a Freescale Semiconductor SCI system whose bus frequency is driven by a crystal, the allowed baud rate mismatch is about ±4.5 percent for 8-bit data format and about ±4 percent for 9-bit data format. Although baud rate modulo divider settings do not always produce baud rates that exactly match standard rates, it is normally possible to get within a few percent, which is acceptable for reliable communications.

14.3.2 Transmitter Functional Description

This section describes the overall block diagram for the SCI transmitter, as well as specialized functions for sending break and idle characters. The transmitter block diagram is shown in Figure 14-2.

The transmitter output (TxD) idle state defaults to logic high (TXINV is cleared following reset). The transmitter output is inverted by setting TXINV. The transmitter is enabled by setting the TE bit in SCIxC2. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the SCI data register (SCIxD).

The central element of the SCI transmitter is the transmit shift register that is 10 or 11 bits long depending on the setting in the M control bit. For the remainder of this section, assume M is cleared, selecting the normal 8-bit data mode. In 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new SCI character, the value waiting in the transmit data register is transferred to the shift register (synchronized with the baud rate clock) and the transmit data register empty (TDRE) status flag is set to indicate another character may be written to the transmit data buffer at SCIxD.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TxD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TxD high, waiting for more characters to transmit.

Writing 0 to TE does not immediately release the pin to be a general-purpose I/O pin. Any transmit activity in progress must first be completed. This includes data characters in progress, queued idle characters, and queued break characters.

14.3.2.1 Send Break and Queued Idle

The SBK control bit in SCIxC2 sends break characters originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0 (10 bit times including the start and stop bits). A longer break of 13 bit times can be enabled by setting BRK13. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 1 and then write 0 to the SBK bit. This action queues a break character to be sent as soon as the shifter is available. If SBK remains 1 when the queued break moves into the shifter (synchronized to the baud rate clock), an additional break character is queued. If the receiving device is another Freescale Semiconductor SCI, the break characters are received as 0s in all eight data bits and a framing error (FE = 1) occurs.

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the TE bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while TE is cleared, the SCI transmitter never actually releases control of the TxD pin. If there is a possibility of the shifter finishing while TE is cleard, set the general-purpose I/O controls so the pin shared with TxD is an output driving a logic 1. This ensures that the TxD line looks like a normal idle line even if the SCI loses control of the port pin between writing 0 and then 1 to TE.

The length of the break character is affected by the BRK13 and M bits as shown below.

Table 14-8. Break Character Length

BRK13	M	Break Character Length
0	0	10 bit times
0	1	11 bit times
1	0	13 bit times
1	1	14 bit times

14.3.3 Receiver Functional Description

In this section, the receiver block diagram (Figure 14-3) is a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, two variations of the receiver wakeup function are explained.

The receiver input is inverted by setting RXINV. The receiver is enabled by setting the RE bit in SCIxC2. Character frames consist of a start bit of logic 0, eight (or nine) data bits (lsb first), and a stop bit of logic 1. For information about 9-bit data mode, refer to Section •, “8- and 9-bit data modes”. For the remainder of this discussion, assume the SCI is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (RDRF)

status flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program has one full character time after RDRF is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full (RDRF = 1), it gets the data from the receive data register by reading SCIxD. The RDRF flag is cleared automatically by a two-step sequence normally satisfied in the course of the user's program that manages receive data. Refer to Section 14.3.4, "Interrupts and Status Flags," for more details about flag clearing.

14.3.3.1 Data Sampling Technique

The SCI receiver uses a 16× baud rate clock for sampling. The receiver starts by taking logic level samples at 16 times the baud rate to search for a falling edge on the RxD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The 16× baud rate clock divides the bit time into 16 segments labeled RT1 through RT16. When a falling edge is located, three more samples are taken at RT3, RT5, and RT7 to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a receive character.

The receiver then samples each bit time, including the start and stop bits, at RT8, RT9, and RT10 to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. In the case of the start bit, the bit is assumed to be 0 if at least two of the samples at RT3, RT5, and RT7 are 0 even if one or all of the samples taken at RT8, RT9, and RT10 are 1s. If any sample in any bit time (including the start and stop bits) in a character frame fails to agree with the logic level for that bit, the noise flag (NF) is set when the received character is transferred to the receive data buffer.

The falling edge detection logic continuously looks for falling edges. If an edge is detected, the sample clock is resynchronized to bit times. This improves the reliability of the receiver in the presence of noise or mismatched baud rates. It does not improve worst case analysis because some characters do not have any extra falling edges anywhere in the character frame.

In the case of a framing error, provided the received character was not a break character, the sampling logic that searches for a falling edge is filled with three logic 1 samples so that a new start bit can be detected almost immediately.

In the case of a framing error, the receiver is inhibited from receiving any new characters until the framing error flag is cleared. The receive shift register continues to function, but a complete character cannot transfer to the receive data buffer if FE remains set.

14.3.3.2 Receiver Wakeup Operation

Receiver wakeup is a hardware mechanism that allows an SCI receiver to ignore the characters in a message intended for a different SCI receiver. In such a system, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up (RWU) control bit in SCIxC2. When RWU bit is set, the status flags associated with the receiver (with the exception of the idle bit, IDLE, when RWUID bit is set) are inhibited from setting, thus eliminating the software overhead for handling the unimportant message characters. At

the end of a message, or at the beginning of the next message, all receivers automatically force RWU to 0 so all receivers wake up in time to look at the first character(s) of the next message.

14.3.3.2.1 Idle-Line Wakeup

When wake is cleared, the receiver is configured for idle-line wakeup. In this mode, RWU is cleared automatically when the receiver detects a full character time of the idle-line level. The M control bit selects 8-bit or 9-bit data mode that determines how many bit times of idle are needed to constitute a full character time (10 or 11 bit times because of the start and stop bits).

When RWU is one and RWUID is zero, the idle condition that wakes up the receiver does not set the IDLE flag. The receiver wakes up and waits for the first data character of the next message that sets the RDRF flag and generates an interrupt if enabled. When RWUID is one, any idle condition sets the IDLE flag and generates an interrupt if enabled, regardless of whether RWU is zero or one.

The idle-line type (ILT) control bit selects one of two ways to detect an idle line. When ILT is cleared, the idle bit counter starts after the start bit so the stop bit and any logic 1s at the end of a character count toward the full character time of idle. When ILT is set, the idle bit counter does not start until after a stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

14.3.3.2.2 Address-Mark Wakeup

When wake is set, the receiver is configured for address-mark wakeup. In this mode, RWU is cleared automatically when the receiver detects a logic 1 in the most significant bit of a received character (eighth bit when M is cleared and ninth bit when M is set).

Address-mark wakeup allows messages to contain idle characters, but requires the msb be reserved for use in address frames. The logic 1 msb of an address frame clears the RWU bit before the stop bit is received and sets the RDRF flag. In this case, the character with the msb set is received even though the receiver was sleeping during most of this character time.

14.3.4 Interrupts and Status Flags

The SCI system has three separate interrupt vectors to reduce the amount of software needed to isolate the cause of the interrupt. One interrupt vector is associated with the transmitter for TDRE and TC events. Another interrupt vector is associated with the receiver for RDRF, IDLE, RXEDGIF, and LBKDIF events. A third vector is used for OR, NF, FE, and PF error conditions. Each of these ten interrupt sources can be separately masked by local interrupt enable masks. The flags can be polled by software when the local masks are cleared to disable generation of hardware interrupt requests.

The SCI transmitter has two status flags that can optionally generate hardware interrupt requests. Transmit data register empty (TDRE) indicates when there is room in the transmit data buffer to write another transmit character to SCIxD. If the transmit interrupt enable (TIE) bit is set, a hardware interrupt is requested when TDRE is set. Transmit complete (TC) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TxD at the inactive level. This flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (TCIE) bit is set, a hardware interrupt is requested when TC is set. Instead of hardware

interrupts, software polling may be used to monitor the TDRE and TC status flags if the corresponding TIE or TCIE local interrupt masks are cleared.

When a program detects that the receive data register is full (RDRF = 1), it gets the data from the receive data register by reading SCIxD. The RDRF flag is cleared by reading SCIxS1 while RDRF is set and then reading SCIxD.

When polling is used, this sequence is naturally satisfied in the normal course of the user program. If hardware interrupts are used, SCIxS1 must be read in the interrupt service routine (ISR). Normally, this is done in the ISR anyway to check for receive errors, so the sequence is automatically satisfied.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the RxD line remains idle for an extended period of time. IDLE is cleared by reading SCIxS1 while IDLE is set and then reading SCIxD. After IDLE has been cleared, it cannot become set again until the receiver has received at least one new character and has set RDRF.

If the associated error was detected in the received character that caused RDRF to be set, the error flags — noise flag (NF), framing error (FE), and parity error flag (PF) — are set at the same time as RDRF. These flags are not set in overrun cases.

If RDRF was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (OR) flag is set instead of the data along with any associated NF, FE, or PF condition is lost.

At any time, an active edge on the RxD serial data input pin causes the RXEDGIF flag to set. The RXEDGIF flag is cleared by writing a 1 to it. This function does depend on the receiver being enabled (RE = 1).

14.3.5 Additional SCI Functions

The following sections describe additional SCI functions.

14.3.5.1 8- and 9-Bit Data Modes

The SCI system (transmitter and receiver) can be configured to operate in 9-bit data mode by setting the M control bit in SCIxC1. In 9-bit mode, there is a ninth data bit to the left of the msb of the SCI data register. For the transmit data buffer, this bit is stored in T8 in SCIxC3. For the receiver, the ninth bit is held in R8 in SCIxC3.

For coherent writes to the transmit data buffer, write to the T8 bit before writing to SCIxD.

If the bit value to be transmitted as the ninth bit of a new character is the same as for the previous character, it is not necessary to write to T8 again. When data is transferred from the transmit data buffer to the transmit shifter, the value in T8 is copied at the same time data is transferred from SCIxD to the shifter.

The 9-bit data mode is typically used with parity to allow eight bits of data plus the parity in the ninth bit, or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. In custom protocols, the ninth bit can also serve as a software-controlled marker.

14.3.5.2 Stop Mode Operation

During all stop modes, clocks to the SCI module are halted.

In stop1 and stop2 modes, all SCI register data is lost and must be re-initialized upon recovery from these two stop modes. No SCI module registers are affected in stop3 mode.

The receive input active edge detect circuit remains active in stop3 mode, but not in stop2. An active edge on the receive input brings the CPU out of stop3 mode if the interrupt is not masked (RXEDGIE = 1).

Because the clocks are halted, the SCI module resumes operation upon exit from stop (only in stop3 mode). Software should ensure stop mode is not entered while there is a character being transmitted out of or received into the SCI module.

14.3.5.3 Loop Mode

When LOOPS is set, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Loop mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and the RxD pin is not used by the SCI, so it reverts to a general-purpose port I/O pin.

14.3.5.4 Single-Wire Operation

When LOOPS is set, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Single-wire mode implements a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TxD pin. The RxD pin is not used and reverts to a general-purpose port I/O pin.

In single-wire mode, the TXDIR bit in SCIxC3 controls the direction of serial data on the TxD pin. When TXDIR is cleared, the TxD pin is an input to the SCI receiver and the transmitter is temporarily disconnected from the TxD pin so an external device can send serial data to the receiver. When TXDIR is set, the TxD pin is an output driven by the transmitter. In single-wire mode, the internal loop back connection from the transmitter to the receiver causes the receiver to receive characters that are sent out by the transmitter.

Chapter 15
Inter-Integrated Circuit (IIC)

15.1 Introduction

The inter-integrated circuit (IIC) provides a method of communication between a number of devices. 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. Support System Management Bus Specification(SMBus), version2.

For additional detail, please refer to volume 1 of the *HCS08 Reference Manual*, (Freescale Semiconductor document order number HCS08RMv1).

15.1.1 Features

The IIC includes these distinctive features:

- Compatible with IIC bus standard
- Multi-master operation
- Software programmable for one of 64 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- START and STOP signal generation/detection
- Repeated START signal generation/detection
- Acknowledge bit generation/detection
- Bus busy detection
- General call recognition
- 10-bit address extension
- Support System Management Bus Specification(SMBus), version2
- Programmable glitch input filter

15.1.2 Modes of Operation

A brief description of the IIC in the various MCU modes is given here.

- **Run mode** — This is the basic mode of operation. To conserve power in this mode, disable the module.
- **Wait mode** — The module will continue to operate while the MCU is in wait mode and can provide a wake-up interrupt.
- **Stop mode** — The IIC is inactive in stop3 mode for reduced power consumption. The STOP instruction does not affect IIC register states. Stop2 will reset the register contents.

15.1.3 Block Diagram

Figure 15-1 is a block diagram of the IIC.

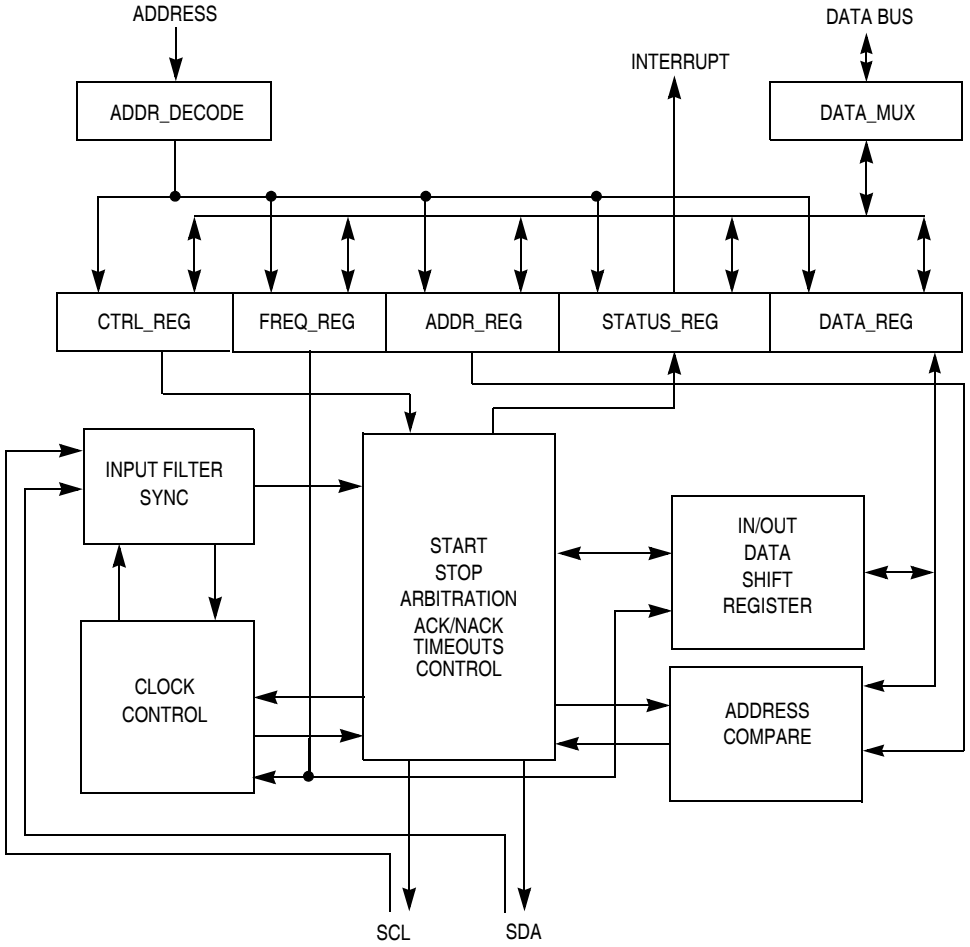


Figure 15-1. IIC Functional Block Diagram

15.2 External Signal Description

This section describes each user-accessible pin signal.

15.2.1 SCL — Serial Clock Line

The bidirectional SCL is the serial clock line of the IIC system.

15.2.2 SDA — Serial Data Line

The bidirectional SDA is the serial data line of the IIC system.

15.3 Register Definition

15.3.1 Module Memory Map

The IIC has ten 8-bit registers. The base address of the module is hardware programmable. The IIC register map is fixed and begins at the module's base address. Table 15-1 summarizes the IIC module's address space. The following section describes the bit-level arrangement and functionality of each register.

Table 15-1. Module Memory Map

Address	Use	Access
Base + $0000	IIC Address Register 1 (IICA1)	read/write
Base + $0001	IIC Frequency Divider Register (IICF)	read/write
Base + $0002	IIC Control Register 1 (IICC1)	read/write
Base + $0003	IIC Status Register (IICS)	read
Base + $0004	IIC Data IO Register (IICD)	read/write
Base + $0005	IIC Control Register 2 (IICC2)	read/write
Base + $0006	SMBUS IIC Control and Status Register (IICSMB)	read/write
Base + $0007	IIC Address Register 2 (IICA2)	read/write
Base + $0008	IIC SCL Low Time Out Register High (IICSLTH)	read/write
Base + $0009	IIC SCL Low Time Out Register Low (IICSLTL)	read/write
Base + $000A	IIC input programmable filter (IICFLT)	read/write

This section consists of the IIC register descriptions in address order.

Refer to the direct-page register summary in the Chapter 3, "Memory," for the absolute address assignments for all IIC registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

15.3.2 IIC Address Register 1 (IICA1)

	7	6	5	4	3	2	1	0
R	AD7	AD6	AD5	AD4	AD3	AD2	AD1	0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 15-2. IIC Address Register 1 (IICA1)

Table 15-2. IICA1 Field Descriptions

Field	Description
7:1 AD[7:1]	**Slave Address 1**— The AD[7:1] field contains the slave address to be used by the IIC module. This field is used on the 7-bit address scheme and the lower seven bits of the 10-bit address scheme.

15.3.3 IIC Frequency Divider Register (IICF)

	7	6	5	4	3	2	1	0
R W	MULT				ICR			
Reset	0	0	0	0	0	0	0	0

Figure 15-3. IIC Frequency Divider Register (IICF)

Table 15-3. IICF Field Descriptions

Field	Description
7:6 MULT	**IIC Multiplier Factor** — The MULT bits define the multiplier factor mul. This factor is used along with the SCL divider to generate the IIC baud rate. The multiplier factor mul as defined by the MULT bits is provided below. 00 mul = 01 01 mul = 02 10 mul = 04 11 Reserved
5:0 ICR	**IIC Clock Rate** — The ICR bits are used to prescale the bus clock for bit rate selection. These bits and the MULT bits are used to determine the IIC baud rate, the SDA hold time, the SCL Start hold time and the SCL Stop hold time. Table 15-4 provides the SCL divider and hold values for corresponding values of the ICR. The SCL divider multiplied by multiplier factor mul is used to generate IIC baud rate. **IIC baud rate = bus speed (Hz)/(mul * SCL divider)** ***Eqn. 15-1*** SDA hold time is the delay from the falling edge of SCL (IIC clock) to the changing of SDA (IIC data). **SDA hold time = bus period (s) * mul * SDA hold value** ***Eqn. 15-2*** SCL Start hold time is the delay from the falling edge of SDA (IIC data) while SCL is high (Start condition) to the falling edge of SCL (IIC clock). **SCL Start hold time = bus period (s) * mul * SCL Start hold value** ***Eqn. 15-3*** SCL Stop hold time is the delay from the rising edge of SCL (IIC clock) to the rising edge of SDA SDA (IIC data) while SCL is high (Stop condition). **SCL Stop hold time = bus period (s) * mul * SCL Stop hold value** ***Eqn. 15-4***

For example if the bus speed is 8MHz, the table below shows the possible hold time values with different ICR and MULT selections to achieve an IIC baud rate of 100kbps.

MULT	ICR	Hold times (μs)		
		SDA	SCL Start	SCL Stop
0x2	0x00	3.500	3.000	5.500
0x1	0x07	2.500	4.000	5.250
0x1	0x0B	2.250	4.000	5.250
0x0	0x14	2.125	4.250	5.125
0x0	0x18	1.125	4.750	5.125

Table 15-4. IIC Divider and Hold Values

ICR (hex)	SCL Divider	SDA Hold Value	SCL Hold (Start) Value	SDA Hold (Stop) Value
00	20	7	6	11
01	22	7	7	12
02	24	8	8	13
03	26	8	9	14
04	28	9	10	15
05	30	9	11	16
06	34	10	13	18
07	40	10	16	21
08	28	7	10	15
09	32	7	12	17
0A	36	9	14	19
0B	40	9	16	21
0C	44	11	18	23
0D	48	11	20	25
0E	56	13	24	29
0F	68	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

ICR (hex)	SCL Divider	SDA Hold Value	SCL Hold (Start) Value	SCL Hold (Stop) Value
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
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

15.3.4 IIC Control Register (IICC1)

	7	6	5	4	3	2	1	0
R W	IICEN	IICIE	MST	TX	TXAK	RSTA		
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 15-4. IIC Control Register (IICC1)

Table 15-5. IICC1 Field Descriptions

Field	Description
7 IICEN	**IIC Enable** — The IICEN bit determines whether the IIC module is enabled. 0 IIC is not enabled. 1 IIC is enabled.
6 IICIE	**IIC Interrupt Enable** — The IICIE bit determines whether an IIC interrupt is requested. 0 IIC interrupt request not enabled. 1 IIC interrupt request enabled.
5 MST	**Master Mode Select** — When the MST bit is changed from a 0 to a 1, a START signal is generated on the bus and master mode is selected. When this bit changes from a 1 to a 0 a STOP signal is generated and the mode of operation changes from master to slave. 0 Slave mode. 1 Master mode.
4 TX	**Transmit Mode Select** — The TX bit selects the direction of master and slave transfers. 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. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. 0 Receive. 1 Transmit.
3 TXAK	**Transmit Acknowledge Enable** — This bit specifies the value driven onto the SDA during data acknowledge cycles for both master and slave receivers. There are two conditions will effect NAK/ACK generation. If FACK (fast NACK/ACK) is cleared, 0 An acknowledge signal will be sent out to the bus on the following receiving data byte. 1 No acknowledge signal response is sent to the bus on the following receiving data byte. If FASK bit is set. no ACK or NACK is sent out after receiving one data byte until this TXAK bit is written 0 An acknowledge signal will be sent out to the bus on the current receiving data byte 1 No acknowledge signal response is sent to the bus on the current receiving data byte Note: SCL is held to low until TXAK is written.
2 RSTA (Write Only read always 0)	**Repeat START** — Writing a 1 to this bit will generate a repeated START condition provided it is the current master. Attempting a repeat at the wrong time will result in loss of arbitration. 0 No repeat start detected in bus operation. 1 Repeat start generated.

15.3.5 IIC Status Register (IICS)

	7	6	5	4	3	2	1	0
R	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
W								
Reset	1	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 15-5. IIC Status Register (IICS)

Table 15-6. IICS Field Descriptions

Field	Description
7 TCF	**Transfer Complete Flag** — This bit is set on the completion of a byte and acknowledge bit transfer. Note that this bit is only valid during or immediately following a transfer to the IIC module or from the IIC module.The TCF bit is cleared by reading the IICD register in receive mode or writing to the IICD in transmit mode. 0 Transfer in progress. 1 Transfer complete.
6 IAAS	**Addressed as a Slave** — The IAAS bit is set when one of the following conditions is met 1) When the calling address matches the programmed slave address, 2) If the GCAEN bit is set and a general call is received. 3) If SIICAEN bit is set, when the calling address matches the 2nd programmed slave address 4) If ALERTEN bit is set and SMBus alert response address is received This bit is set before ACK bit. The CPU needs to check the SRW bit and set TX/RX bit accordingly. Writing the IICC1 register with any value clears this bit. 0 Not addressed. 1 Addressed as a slave.
5 BUSY	**Bus Busy** — The BUSY bit indicates the status of the bus regardless of slave or master mode. The BUSY bit is set when a START signal is detected and cleared when a STOP signal is detected. 0 Bus is idle. 1 Bus is busy.
4 ARBL	**Arbitration Lost** — This bit is set by hardware when the arbitration procedure is lost. The ARBL bit must be cleared by software, by writing a 1 to it. 0 Standard bus operation. 1 Loss of arbitration.
2 SRW	**Slave Read/Write** — When addressed as a slave the SRW bit indicates the value of the R/$\overline{W}$ command bit of the calling address sent to the master. 0 Slave receive, master writing to slave. 1 Slave transmit, master reading from slave.

Table 15-6. IICS Field Descriptions (continued)

Field	Description
1 IICIF	**IIC Interrupt Flag** — The IICIF bit is set when an interrupt is pending. This bit must be cleared by software, by writing a 1 to it in the interrupt routine. One of the following events can set the IICIF bit: • One byte transfer including ACK/NACK bit completes if FACK = 0 • One byte transfer including ACK/NACK bit completes if FACK = 1 and this byte is an address byte • One byte transfer excluding ACK/NCAK bit completes if FACK = 1 and this byte is a data byte. an ACK or NACK is sent out on the bus by writing 0 or 1 to TXAK after this bit is set. • Match of slave addresses to calling address (Primary Slave address, General Call address, Alert Response address, and Second Slave address) (Received address is stored in data register) • Arbitration lost • Timeouts in SMBus mode except high timeout 0 No interrupt pending. 1 Interrupt pending.
0 RXAK	**Receive Acknowledge** — When the RXAK bit is low, it indicates an acknowledge signal has been received after the completion of one byte of data transmission on the bus. If the RXAK bit is high it means that no acknowledge signal is detected. 0 Acknowledge received. 1 No acknowledge received.

15.3.6 IIC Data I/O Register (IICD)

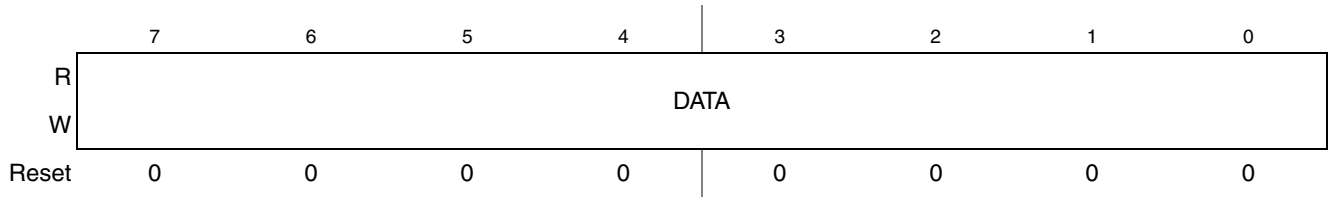


Figure 15-6. IIC Data I/O Register (IICD)

Table 15-7. IICD Field Descriptions

Field	Description
7:0 DATA	**Data** — In master transmit mode, when data is written to the IICD, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates receiving of the next byte of data.

NOTE

When transitioning out of master receive mode, the IIC mode should be switched before reading the IICD register to prevent an inadvertent initiation of a master receive data transfer.

In slave mode, the same functions are available after an address match has occurred.

Note that the TX bit in IICC 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 IICD will not initiate the receive.

Reading the IICD will return the last byte received while the IIC is configured in either master receive or slave receive modes. The IICD does not reflect every byte that is transmitted on the IIC bus, nor can software verify that a byte has been written to the IICD correctly by reading it back.

In master transmit mode, the first byte of data written to IICD following assertion of MST (Start bit) or assertion of RSTA bit (repeated Start) is used for the address transfer and should comprise of the calling address (in bit 7 to bit 1) concatenated with the required R/$\overline{W}$ bit (in position bit 0).

15.3.7 IIC Control Register 2 (IICC2)

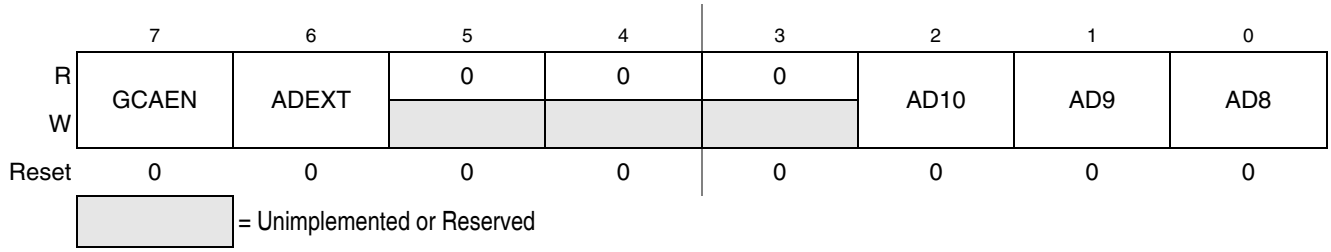


Figure 15-7. IIC Control Register (IICC2)

Table 15-8. IICC2 Field Descriptions

Field	Description
7 GCAEN	**General Call Address Enable** — The GCAEN bit enables or disables general call address. 0 General call address is disabled 1 General call address is enabled.
6 ADEXT	**Address Extension** — The ADEXT bit controls the number of bits used for the slave address. 0 7-bit address scheme 1 10-bit address scheme
2:0 AD[10:8]	**Slave Address** — The AD[10:8] field contains the upper three bits of the slave address in the 10-bit address scheme. This field is only valid when the ADEXT bit is set.

15.3.8 IIC SMBus Control and Status Register (IICSMB)

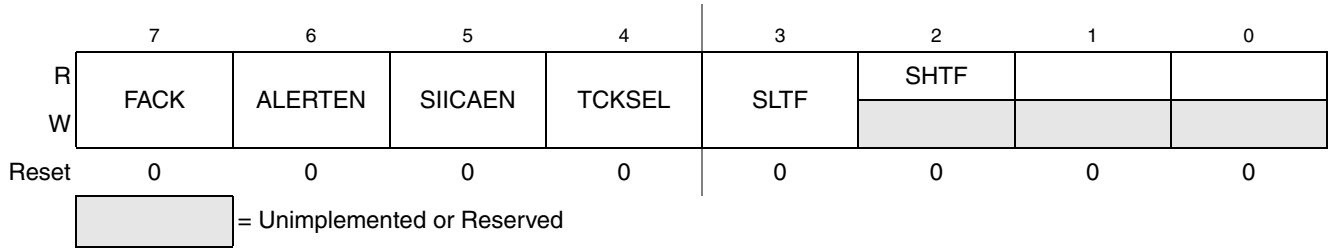


Figure 15-8. IIC SMBus Control and Status Register (IICSMB)

Table 15-9. IICSMB[NOTE] Field Descriptions

Field	Description
7 FACK	**Fast NACK/ACK enable** — For SMBus Packet Error Checking, CPU should be able to issue an ACK or NACK according to the result of receiving data byte. 0 ACK or NACK will be sent out on the following receiving data byte. 1 Writing an 0 to TXAK after receiving data byte will generate an ACK; Writing an 1 to TXAK after receiving data byte will generate a NACK
6 ALERTEN reserved	**SMBus Alert Response Address Enable** — The ALERTEN bit enables or disable SMBus alert response address. 0 SMBus alert response address matching is disabled 1 SMBus alert response address matching is enabled.
5 SIICAEN	**Second IIC Address Enable** — The SIICAEN bit enables or disable SMBus device default address. 0 IIC Address Register 2 matching is disabled. 1 IIC Address Register 2 matching is enabled.
4 TCKSEL	Time Out Counter Clock Select— This bit selects the clock sources of Time Out Counter 0 Time Out Counter counts at bus/64 frequency 1 Time Out Counter counts at the bus frequency
3 SLTF	SCL Low Timeout Flag — This read-only bit is set to logic 1 when IICSLT loaded non zero value (LoValue) and a SCL Low Time Out occurs. This bit is cleared by software, by writting a logic 1 to it 0 No LOW TIME OUT occurs. 1 A LOW TIME OUT occurs. Note: LOW TIME OUT function is disabled when IIC SCL LOW TIMER OUT register is set to zero
2 SHTF	SCL High Timeout Flag — This read-only bit is set to logic 1 when SCL and SDA are held high more than clock * LoValue/512, which indicates the Bus Free. This bit is cleared automatically. 0 No HIGH TIMEOUT occurs. 1 An HIGH TIMEOUT occurs.

NOTE

1. A master can assume that the bus is free if it detects that the clock and data signals have been high for greater than THIGH,MAX, however, the SHTF will rise in bus transmission process but bus idle state.

2. When TCKSEL=1 there is no meaning to monitor SHTF since the bus speed is too high to match the protocol of SMBus.

15.3.9 IIC Address Register 2 (IICA2)

	7	6	5	4	3	2	1	0
R	SAD7	SAD6	SAD5	SAD4	SAD3	SAD2	SAD1	0
W								
Reset	1	1	0	0	0	0	1	0

= Unimplemented or Reserved

Figure 15-9. IIC Address Register 2 (IICA2)

Table 15-10. IICA2 Field Descriptions

Field	Description
7:1 SAD[7:1]	**SMBUs Address** — The AD[7:1] field contains the slave address to be used by the SMBus. This field is used on the device default address or other related address

15.3.10 IIC SCL Low Time Out Register High (IICSLTH)

	7	6	5	4	3	2	1	0
R	SSLT15	SSLT14	SSLT13	SSLT12	SSLT11	SSLT10	SSLT9	SSLT8
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 15-10. IIC SCL Low Time Out Register High (IICSLTH)

Table 15-11. IICSLTH Field Descriptions

Field	Description
7:0 SSLT[15:8]	**The value in this register is the most significant byte of SCL low time out value that determines the time-out period of SCL low.**

15.3.11 IIC SCL LowTime Out register Low (IICSLTL)

	7	6	5	4	3	2	1	0
R	SSLT7	SSLT6	SSLT5	SSLT4	SSLT3	SSLT2	SSLT1	SSLT0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 15-11. IIC SCL LowTime Out register Low (IICSLTL)

Table 15-12. IICSLTL Field Descriptions

Field	Description
7:0 SSLT[7:0]	**The value in this register is the least significant byte of SCL low time out value that determines the time-out period of SCL low..**

15.3.12 IIC Programmable Input Glitch Filter (IICFLT)

	7	6	5	4	3	2	1	0
R	0	0	0	0	FLT3	FLT2	FLT1	FLT0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 15-12. IIC Programmable Input Glitch Filter (IICFLT)

Table 15-13. IICFLT Field Descriptions

Field	Description
3:0 FLT	IIC Programmable Filter Factor contain the programming controls for the width of glitch (in terms of bus clock cycles) the filter should absorb; in other words, the filter does not let glitches less than or equal to this width setting pass. FLT[3:0] 0000 No Filter / Bypass 0001 Filter glitches up to width of 1 (half) IPBUS clock cycle 0010 Filter glitches up to width of 2 (half) IPBUS clock cycles 0011 Filter glitches up to width of 3 (half) IPBUS clock cycles 0100 Filter glitches up to width of 4 (half) IPBUS clock cycles 0101 Filter glitches up to width of 5 (half) IPBUS clock cycles 0110 Filter glitches up to width of 6 (half) IPBUS clock cycles 0111 Filter glitches up to width of 7 (half) IPBUS clock cycles 1000 Filter glitches up to width of 8 (half) IPBUS clock cycles 1001 Filter glitches up to width of 9 (half) IPBUS clock cycles 1010 Filter glitches up to width of 10 (half) IPBUS clock cycles 1011 Filter glitches up to width of 11 (half) IPBUS clock cycles 1100 Filter glitches up to width of 12 (half) IPBUS clock cycles 1101 Filter glitches up to width of 13 (half) IPBUS clock cycles 1110 Filter glitches up to width of 14 (half) IPBUS clock cycles 1111 Filter glitches up to width of 15 (half) IPBUS clock cycles NOTE: If the filter input clock is connected to 2X IPBUS clock. for instance S08 core clock then the (half) will affect the width of glitches

15.4 Functional Description

This section provides a complete functional description of the IIC module.

15.4.1 IIC 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. A logic AND function is exercised on both lines with external pullup 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
- STOP signal

The STOP signal should not be confused with the CPU STOP instruction. The IIC bus system communication is described briefly in the following sections and illustrated in Figure 15-13.

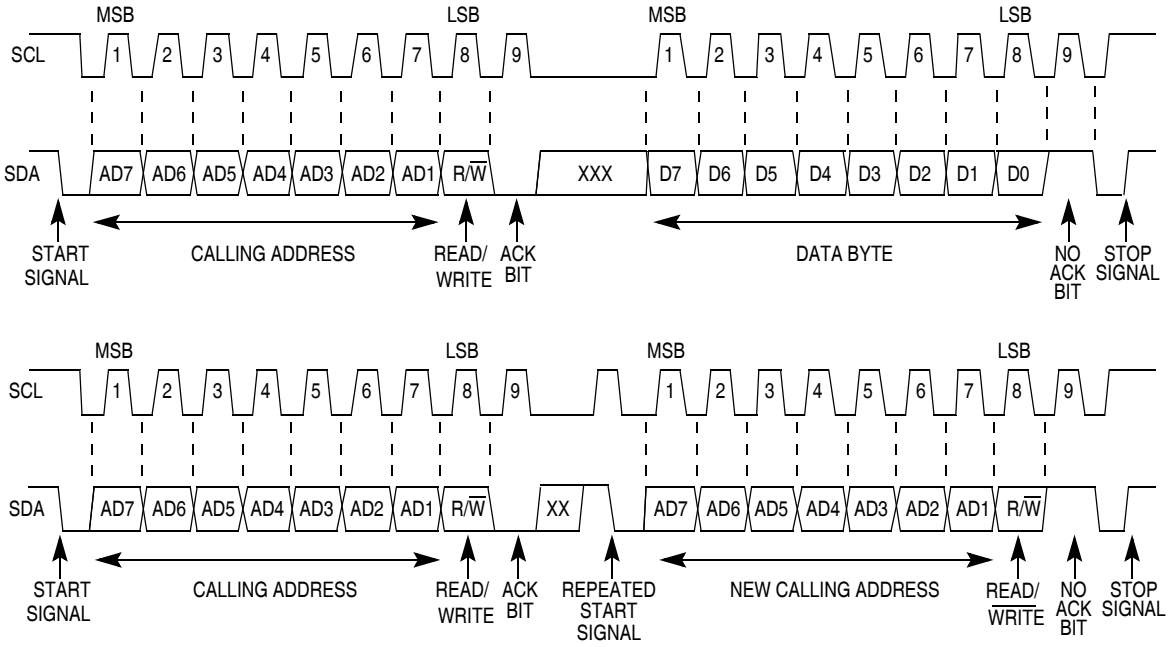


Figure 15-13. IIC Bus Transmission Signals

15.4.1.1 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 15-13, 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.

15.4.1.2 Slave Address Transmission

The first byte of data transferred immediately after the START signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/$\overline{W}$ bit. The R/$\overline{W}$ bit tells the slave the desired direction of data transfer.

1 = Read transfer, the slave transmits data to the master.
0 = Write transfer, the master transmits data to the slave.

Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see Figure 15-13).

No two slaves in the system may have the same address. If the IIC module is the master, it must not transmit an address that is equal to its own slave address. The IIC cannot be master and slave at the same time. However, if arbitration is lost during an address cycle, the IIC will revert to slave mode and operate correctly even if it is being addressed by another master.

15.4.1.3 Data Transfer

Before successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/$\overline{W}$ 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 15-13. There is one clock pulse on SCL for each data bit, the MSB being transferred first. Each data byte is followed by a 9th (acknowledge) bit, which is signalled from the receiving device. An acknowledge is signalled by pulling the SDA low at the ninth clock. In summary, one complete data transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master in the 9th bit time, the SDA line must be left high by the slave. The master interprets the failed acknowledge as an unsuccessful data transfer.

If the master receiver does not acknowledge the slave transmitter after a data byte transmission, the slave interprets this as an end of data transfer and releases the SDA line.

In either case, the data transfer is aborted and the master does one of two things:

- Relinquishes the bus by generating a STOP signal.
- Commences a new calling by generating a repeated START signal.

15.4.1.4 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 15-13).

The master can generate a STOP even if the slave has generated an acknowledge at which point the slave must release the bus.

15.4.1.5 Repeated START Signal

As shown in Figure 15-13, 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.

15.4.1.6 Arbitration Procedure

The IIC 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.

15.4.1.7 Clock Synchronization

Because wire-AND logic is performed on the SCL line, a high-to-low transition on the SCL line affects all the devices connected on the bus. The devices start counting their low period and after 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 still 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 15-14). 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.

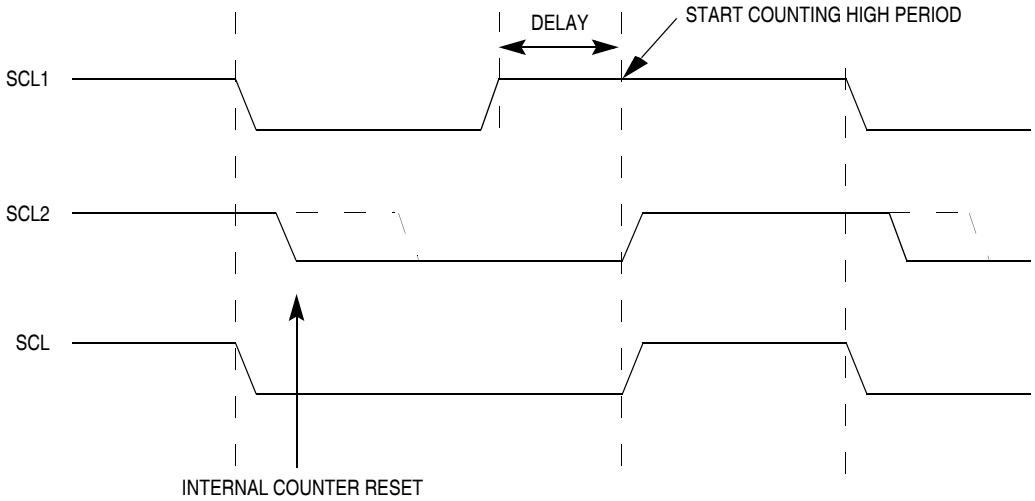


Figure 15-14. IIC Clock Synchronization

15.4.1.8 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.

15.4.1.9 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.

15.4.2 10-Bit Address

For 10-bit addressing, bit 11110 is used for the first 5 bits of the first address byte. Various combinations of read/write formats are possible within a transfer that includes 10-bit addressing.

15.4.2.1 Master-Transmitter Addresses a Slave-Receiver

The transfer direction is not changed (see Table 15-14). When a 10-bit address follows a START condition, each slave compares the first seven bits of the first byte of the slave address (11110XX) with its own address and tests whether the eighth bit (R/$\overline{W}$ direction bit) is 0. It is possible that more than one device will find a match and generate an acknowledge (A1). Each slave that finds a match will compare the eight bits of the second byte of the slave address with its own address, but only one slave will find a match and generate an acknowledge (A2). The matching slave will remain addressed by the master until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.

S	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 0	A1	Slave Address 2nd byte AD[8:1]	A2	Data	A	...	Data	A/A	P

Table 15-14. Master-Transmitter Addresses Slave-Receiver with a 10-bit Address

After the master-transmitter has sent the first byte of the 10-bit address, the slave-receiver will see an IIC interrupt. User software must ensure that for this interrupt, the contents of IICD are ignored and not treated as valid data.

15.4.2.2 Master-Receiver Addresses a Slave-Transmitter

The transfer direction is changed after the second R/$\overline{W}$ bit (see Table 15-15). Up to and including acknowledge bit A2, the procedure is the same as that described for a master-transmitter addressing a slave-receiver. After the repeated START condition (Sr), a matching slave remembers that it was addressed before. This slave then checks whether the first seven bits of the first byte of the slave address following Sr are the same as they were after the START condition (S), and tests whether the eighth (R/$\overline{W}$) bit is 1. If there is a match, the slave considers that it has been addressed as a transmitter and generates acknowledge A3. The slave-transmitter remains addressed until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.

After a repeated START condition (Sr), all other slave devices will also compare the first seven bits of the first byte of the slave address with their own addresses and test the eighth (R/$\overline{W}$) bit. However, none of them will be addressed because R/$\overline{W}$ = 1 (for 10-bit devices), or the 11110XX slave address (for 7-bit devices) does not match.

S	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 0	A1	Slave Address 2nd byte AD[8:1]	A2	Sr	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 1	A3	Data	A	...	Data	A	P

Table 15-15. Master-Receiver Addresses a Slave-Transmitter with a 10-bit Address

After the master-receiver has sent the first byte of the 10-bit address, the slave-transmitter will see an IIC interrupt. User software must ensure that for this interrupt, the contents of IICD are ignored and not treated as valid data.

15.4.3 Address Matching

All received Addresses can be requested in 7-bit or 10-bit address. IIC Address Register 1, which contains IIC primary slave address, always participates the address matching process. If the GCAEN bit is set, general call will participate the address matching process. If the ALERTEN bit is set, alert response will participate the address matching process. If SIICAEN bit is set, the IIC Address Register 2 will participate the address matching process.

When the IIC responds to one of above mentioned address, it acts as a slave-receiver and the IAAS bit is set after the address cycle. Software need to read the IICD register after the first byte transfer to determine which the address is matched.

15.4.4 System Management Bus Specification

SMBus provides a control bus for system and power management related tasks. A system may use SMBus to pass messages to and from devices instead of tripping individual control lines. Removing the individual control lines reduces pin count. Accepting messages ensures future expandability.With System Management Bus, a device can provide manufacturer information, tell the system what its model/part number is, save its state for a suspend event, report different types of errors, accept control parameters, and return its status.

15.4.4.1 Timeouts

The $T_{TIMEOUT,MIN}$ parameter allows a master or slave to conclude that a defective device is holding the clock low indefinitely or a master is intentionally trying to drive devices off the bus. It is highly recommended that a slave device release the bus (stop driving the bus and let SCL and SDA float high) when it detects any single clock held low longer than $T_{TIMEOUT,MIN}$. Devices that have detected this condition should reset their communication and be able to receive a new START condition in no later than $T_{TIMEOUT,MAX}$.

SMBus defines a clock low time-out, $T_{TIMEOUT}$ of 35 ms and specifies $T_{LOW:SEXT}$ as the cumulative clock low extend time for a slave device and specifies $T_{LOW:MEXT}$ as the cumulative clock low extend time for a master device.

15.4.4.1.1 SCL Low Timeout

If the SCL line is held low by a slave device on the bus, no further communication is possible. Furthermore, the master cannot force the SCL line high to correct the error condition. To solve this problem, the SMBus protocol specifies that devices participating in a transfer must detect any clock cycle held low longer than a “timeout” value condition. Devices that have detected the timeout condition must reset the communication. When active master, if the IIC detects that SMBCLK low has exceeded the value of $T_{TIMEOUT,MIN}$ it must generate a stop condition within or after the current data byte in the transfer process. When slave, upon detection of the $T_{TIMEOUT,MIN}$ condition, the IIC shall reset its communication and be able to receive a new START condition.

15.4.4.1.2 SCL High (SMBus Free) Timeout

The IIC shall assume that the bus is idle, when it has determined that the SMBCLK and SMBDAT signals have been high for at least $T_{HIGH:MAX}$.HIGH timeout can occur in two ways: 1) HIGH timeout detected after a STOP condition appears on the bus; 2) HIGH timeout detected after a START condition, but before a STOP condition appears on the bus.Any master detecting either scenario can assume the bus is free then SHTF rises. HIGH timeout occurred in scenario 2 if it ever detects that both the following is true:BUSY bit is high and SHTF is high.

15.4.4.1.3 CSMBCLK TIMEOUT MEXT

Figure1-10: Timeout measurement intervals illustrates the definition of the timeout intervals, $T_{LOW:SEXT}$ and $T_{LOW:MEXT}$. When master mode, the I2C must not cumulatively extend its clock cycles for a period greater than $T_{LOW:MEXT}$ within a byte, where each byte is defined as START-to-ACK, ACK-to-ACK, or ACK-to-STOP. When CSMBCLK TIMEOUT MEXT occurs SMBus MEXT will rise and also trigger the SLTF.

15.4.4.1.4 CSMBCLK TIMEOUT SEXT

A Master is allowed to abort the transaction in progress to any slave that violates the $T_{LOW:SEXT}$ or $T_{TIMEOUT,MIN}$ specifications. This can be accomplished by the Master issuing a STOP condition at the conclusion of the byte transfer in progress. When slave, the I2C must not cumulatively extend its clock cycles for a period greater than $T_{LOW:SEXT}$ during any message from the initial START to the STOP. When CSMBCLK TIMEOUT SEXT occurs SEXT will rise and also trigger SLTF.

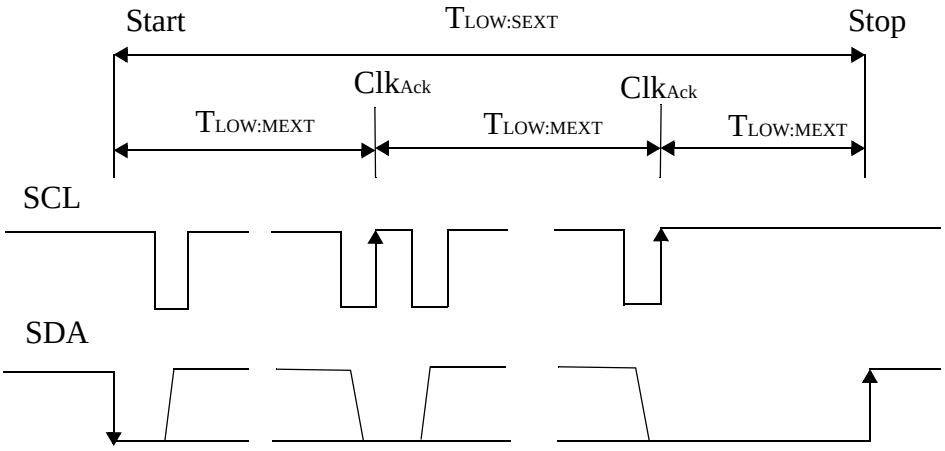


Figure 15-15. Timeout measurement intervals

NOTE

CSMBCLK TIMEOUT SEXT and MEXT are optional functions which will be implemented in second step.

15.4.4.2 FAST ACK and NACK

To improve reliability and communication robustness, implementation of Packet Error Checking (PEC) by SMBus devices is optional for SMBus devices but required for devices participating in and only during the Address Resolution Protocol (ARP) process. The PEC is a CRC-8 error checking byte, calculated on all the message bytes. The PEC is appended to the message by the device that supplied the last data byte. If the PEC is present but not correct, a NACK is issued by receiver. Otherwise an ACK will be issued. In order to calculate the CRC-8 by software, this module can hold SCL line to low after receiving eighth SCL (bit 8th) if this byte is a data byte. So software can determine whether an ACK or NACK should be sent out to the bus by setting or clearing TXAK bit if FASK (fast ACK/NACK enable bit) is enabled.

SMBus requires devices to acknowledge their own address always, as a mechanism to detect a removable devices presence on the bus (battery, docking station, etc.) Besides to indicate a slave device busy condition, SMBus is using the NACK mechanism also to indicate the reception of an invalid command or data. Since such a condition may occur on the last byte of the transfer, it is required that SMBus devices have the ability to generate the not acknowledge after the transfer of each byte and before the completion of the transaction. This is important because SMBus does not provide any other resend signaling. This difference in the use of the NACK signaling has implications on the specific implementation of the SMBus port, especially in devices that handle critical system data such as the SMBus host and the SBS components.

NOTE

In the last byte of master receive slave transmit mode, the master should send NACK to bus so FACK should be switched off before the last byte transmit.

15.5 Resets

The IIC is disabled after reset. The IIC cannot cause an MCU reset.

15.6 Interrupts

The IIC generates a single interrupt.

An interrupt from the IIC is generated when any of the events in Table 15-16 occur, provided the IICIE bit is set. The interrupt is driven by bit IICIF (of the IIC status register) and masked with bit IICIE (of the IIC control register). The IICIF bit must be cleared by software by writing a 1 to it in the interrupt routine. The user can determine the interrupt type by reading the status register. For SMBus timeouts interrupt, the interrupt is driven by SLTF and masked with bit IICIE. The SLTF bit must be cleared by software by writing a 1 to it in the interrupt routine. The user can determine the interrupt type by reading the status register.

Note: In Master receive mode the FACK should be set zero before the last byte transfer.

Table 15-16. Interrupt Summary

Interrupt Source	Status	Flag	Local Enable
Complete 1-byte transfer	TCF	IICIF	IICIE
Match of received calling address	IAAS	IICIF	IICIE
Arbitration Lost	ARBL	IICIF	IICIE
SMBus Timeout Interrupt Flag	SLTF	IICIF	IICIE

15.6.1 Byte Transfer Interrupt

The TCF (transfer complete flag) bit is set at the falling edge of the 9th clock to indicate the completion of byte transfer.

15.6.2 Address Detect Interrupt

When the calling address matches the programmed slave address (IIC address register) or when the GCAEN bit is set and a general call is received, the IAAS bit in the status register is set. The CPU is interrupted, provided the IICIE is set. The CPU must check the SRW bit and set its Tx mode accordingly.

15.6.3 Arbitration Lost Interrupt

The IIC 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, the relative priority of the contending masters is determined by a data arbitration procedure. The IIC module asserts this interrupt when it loses the data arbitration process and the ARBL bit in the status register is set.

Arbitration is lost in the following circumstances:

- SDA sampled as a low when the master drives a high during an address or data transmit cycle.
- SDA sampled as a low when the master drives a high during the acknowledge bit of a data receive cycle.
- A START cycle is attempted when the bus is busy.
- A repeated START cycle is requested in slave mode.
- A STOP condition is detected when the master did not request it.

This bit must be cleared by software by writing a 1 to it.

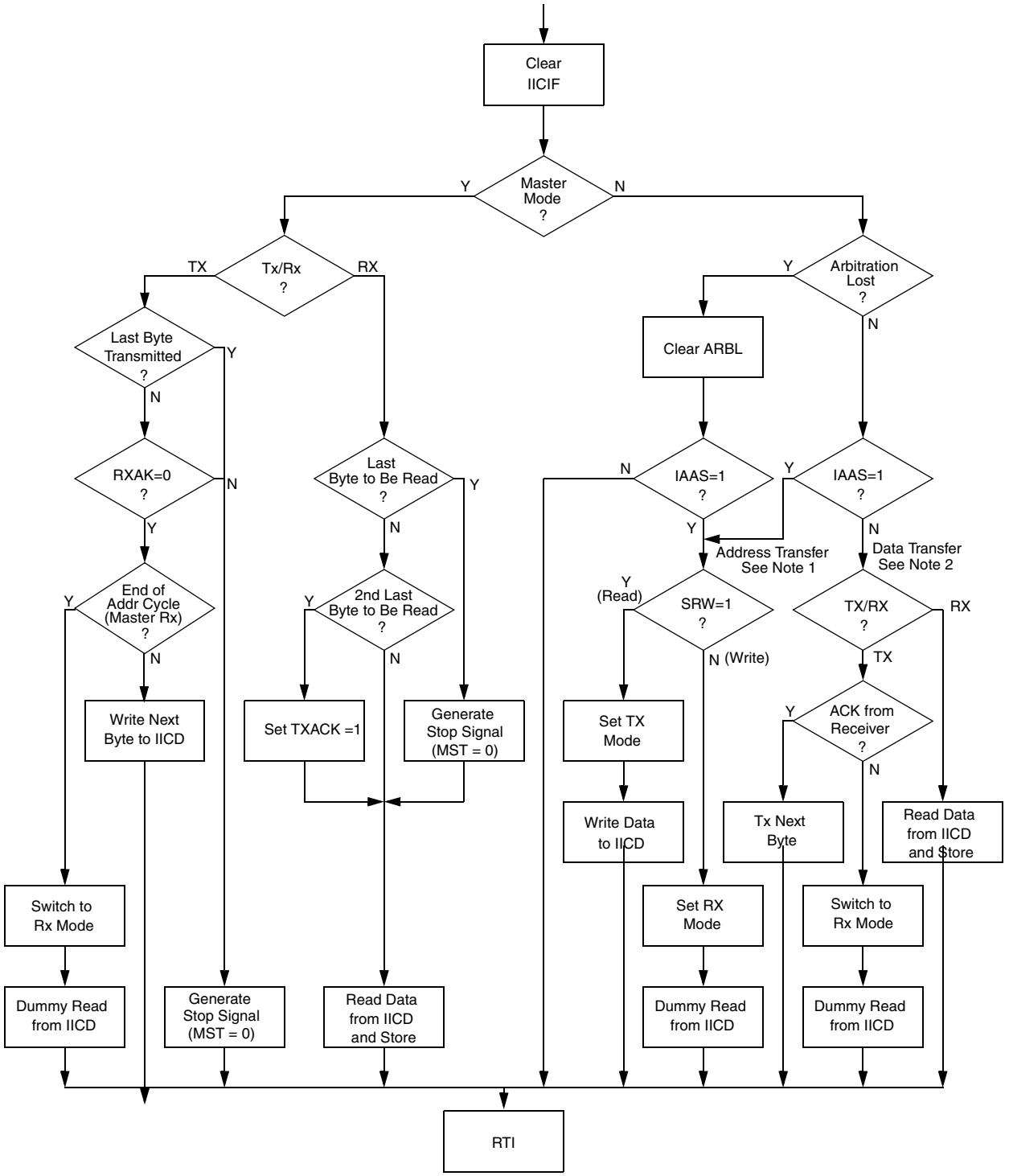
15.6.4 Timeouts Interrupt in SMbus

When IICIE is set, the IIC asserts a timeout interrupt output SLTF upon detection of any of the mentioned timeout conditions, with one exception. The HIGH TIMEOUT mechanism shall not be used to influence the timeout interrupt output, because the HIGH TIMEOUT indicates an idle condition on the bus.

15.6.5 Programmable input glitch filter

An IIC glitch filter has been added outside the IIC legacy modules, but within the IIC package. This filter can absorb glitches on the IIC clock and data lines for I2C module. The width of the glitch to absorb can be specified in terms of number of half bus clock cycles. A single glitch filter control register is provided as IICFLT. Effectively, any down-up-down or up-down-up transition on the data line that occurs within the number of clock cycles programmed here is ignored by the IIC.the programmer only needs to specify the size of glitch (in terms of bus clock cycles) for the filter to absorb and not pass.

15.7 Initialization/Application Information



1 If general call is enabled, a check must be done to determine whether the received address was a general call address (0x00). If the received address was a general call address, then the general call must be handled by user software.

2 When 10-bit addressing is used to address a slave, the slave will see an interrupt following the first byte of the extended address. User software must ensure that for this interrupt, the contents of IICD are ignored and not treated as a valid data transfer.

Figure 15-16. Typical IIC Interrupt Routine

Chapter 16
Timer/PWM Module(TPM)

16.1 Introduction

The TPM is a one-to-eight-channel timer system which supports traditional input capture, output compare, or edge-aligned PWM on each channel. A control bit allows the TPM to be configured such that all channels may be used for center-aligned PWM functions. Timing functions are based on a 16-bit counter with prescaler and modulo features to control frequency and range (period between overflows) of the time reference. This timing system is ideally suited for a wide range of control applications, and the center-aligned PWM capability extends the field of application to motor control in small appliances.

16.1.1 Features

The TPM includes these distinctive features:

- One to eight channels:
  - — Each channel may be input capture, output compare, or edge-aligned PWM
  - — Rising-Edge, falling-edge, or any-edge input capture trigger
  - — Set, clear, or toggle output compare action
  - — Selectable polarity on PWM outputs
- Module may be configured for buffered, center-aligned pulse-width-modulation (CPWM) on all channels
- Timer clock source selectable as prescaled bus clock, fixed system clock, or an external clock pin
  - — Prescale taps for divide-by 1, 2, 4, 8, 16, 32, 64, or 128
  - — Fixed system clock source are synchronized to the bus clock by an on-chip synchronization circuit
  - — External clock pin may be shared with any timer channel pin or a separated input pin
- 16-bit free-running or modulo up/down count operation
- Timer system enable
- One interrupt per channel plus terminal count interrupt

16.1.2 Modes of Operation

For details on low-power mode operation, refer to Table 6-3 in Chapter 6, “Modes of Operation”.

In general, TPM channels may be independently configured to operate in input capture, output compare, or edge-aligned PWM modes. A control bit allows the whole TPM (all channels) to switch to

center-aligned PWM mode. When center-aligned PWM mode is selected, input capture, output compare, and edge-aligned PWM functions are not available on any channels of this TPM module.

When the microcontroller is in active BDM background or BDM foreground mode, the TPM temporarily suspends all counting until the microcontroller returns to normal user operating mode. During stop mode, all system clocks, including the main oscillator, are stopped; therefore, the TPM is effectively disabled until clocks resume. During wait mode, the TPM continues to operate normally. Provided the TPM does not need to produce a real time reference or provide the interrupt source(s) needed to wake the MCU from wait mode, the user can save power by disabling TPM functions before entering wait mode.

- Input capture mode

  When a selected edge event occurs on the associated MCU pin, the current value of the 16-bit timer counter is captured into the channel value register and an interrupt flag bit is set. Rising edges, falling edges, any edge, or no edge (disable channel) may be selected as the active edge which triggers the input capture.
- Output compare mode

  When the value in the timer counter register matches the channel value register, an interrupt flag bit is set, and a selected output action is forced on the associated MCU pin. The output compare action may be selected to force the pin to zero, force the pin to one, toggle the pin, or ignore the pin (used for software timing functions).
- Edge-aligned PWM mode

  The value of a 16-bit modulo register plus 1 sets the period of the PWM output signal. The channel value register sets the duty cycle of the PWM output signal. The user may also choose the polarity of the PWM output signal. Interrupts are available at the end of the period and at the duty-cycle transition point. This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within a TPM.
- Center-aligned PWM mode

  Twice the value of a 16-bit modulo register sets the period of the PWM output, and the channel-value register sets the half-duty-cycle duration. The timer counter counts up until it reaches the modulo value and then counts down until it reaches zero. As the count matches the channel value register while counting down, the PWM output becomes active. When the count matches the channel value register while counting up, the PWM output becomes inactive. This type of PWM signal is called center-aligned because the centers of the active duty cycle periods for all channels are aligned with a count value of zero. This type of PWM is required for types of motors used in small appliances.

This is a high-level description only. Detailed descriptions of operating modes are in later sections.

16.1.3 Block Diagram

The TPM uses one input/output (I/O) pin per channel, TPMCHn (timer channel n) where n is the channel number (1-8). The TPM shares its I/O pins with general purpose I/O port pins (refer to I/O pin descriptions in full-chip specification for the specific chip implementation).

Figure 16-1 shows the TPM structure. The central component of the TPM is the 16-bit counter that can operate as a free-running counter or a modulo up/down counter. The TPM counter (when operating in normal up-counting mode) provides the timing reference for the input capture, output compare, and edge-aligned PWM functions. The timer counter modulo registers, TPMMODH:TPMMODL, control the modulo value of the counter (the values 0x0000 or 0xFFFF effectively make the counter free running). Software can read the counter value at any time without affecting the counting sequence. Any write to either half of the TPMCNT counter resets the counter, regardless of the data value written.

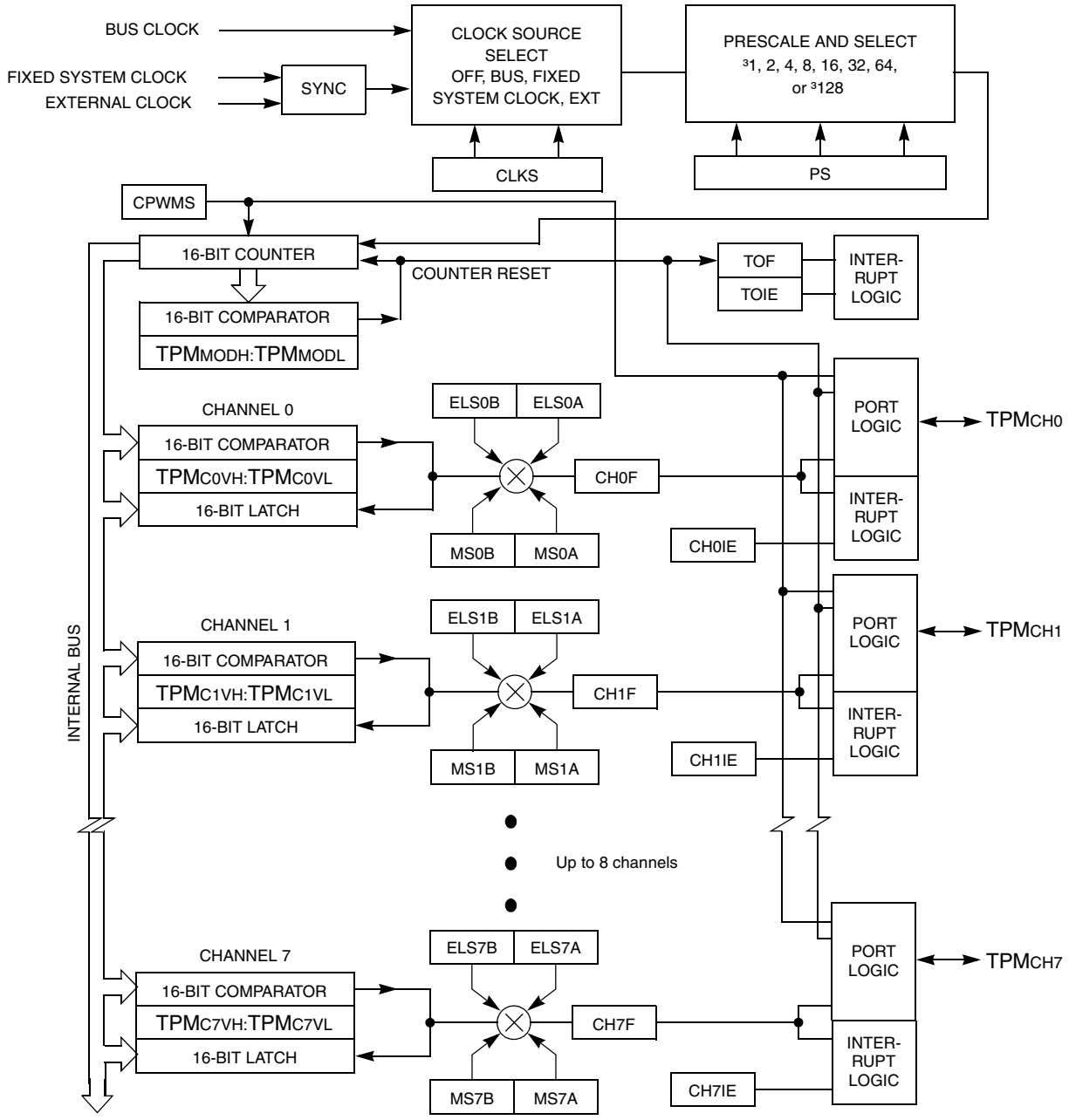


Figure 16-1. TPM Block Diagram

The TPM channels are programmable independently as input capture, output compare, or edge-aligned PWM channels. Alternately, the TPM can be configured to produce CPWM outputs on all channels. When the TPM is configured for CPWMs, the counter operates as an up/down counter; input capture, output compare, and EPWM functions are not practical.

If a channel is configured as input capture, an internal pullup device may be enabled for that channel. The details of how a module interacts with pin controls depends upon the chip implementation because the I/O pins and associated general purpose I/O controls are not part of the module. Refer to the discussion of the I/O port logic in a full-chip specification.

Because center-aligned PWMs are usually used to drive 3-phase AC-induction motors and brushless DC motors, they are typically used in sets of three or six channels.

16.2 Signal Description

Table 16-1 shows the user-accessible signals for the TPM. The number of channels may be varied from one to eight. When an external clock is included, it can be shared with the same pin as any TPM channel; however, it could be connected to a separate input pin. Refer to the I/O pin descriptions in full-chip specification for the specific chip implementation.

Table 16-1. Signal Properties

Name	Function
EXTCLK[1]	External clock source which may be selected to drive the TPM counter.
TPMCHn[2]	I/O pin associated with TPM channel n

1 When preset, this signal can share any channel pin; however depending upon full-chip implementation, this signal could be connected to a separate external pin.

2 n=channel number (1 to 8)

Refer to documentation for the full-chip for details about reset states, port connections, and whether there is any pullup device on these pins.

TPM channel pins can be associated with general purpose I/O pins and have passive pullup devices which can be enabled with a control bit when the TPM or general purpose I/O controls have configured the associated pin as an input. When no TPM function is enabled to use a corresponding pin, the pin reverts to being controlled by general purpose I/O controls, including the port-data and data-direction registers. Immediately after reset, no TPM functions are enabled, so all associated pins revert to general purpose I/O control.

16.2.1 Detailed Signal Descriptions

This section describes each user-accessible pin signal in detail. Although Table 16-1 grouped all channel pins together, any TPM pin can be shared with the external clock source signal. Since I/O pin logic is not part of the TPM, refer to full-chip documentation for a specific derivative for more details about the interaction of TPM pin functions and general purpose I/O controls including port data, data direction, and pullup controls.

16.2.1.1 EXTCLK — External Clock Source

Control bits in the timer status and control register allow the user to select nothing (timer disable), the bus-rate clock (the normal default source), a crystal-related clock, or an external clock as the clock which drives the TPM prescaler and subsequently the 16-bit TPM counter. The external clock source is synchronized in the TPM. The bus clock clocks the synchronizer; the frequency of the external source must be no more than one-fourth the frequency of the bus-rate clock, to meet Nyquist criteria and allowing for jitter.

The external clock signal shares the same pin as a channel I/O pin, so the channel pin will not be usable for channel I/O function when selected as the external clock source. It is the user's responsibility to avoid such settings. If this pin is used as an external clock source (CLKSB:CLKSA = 1:1), the channel can still be used in output compare mode as a software timer (ELSnB:ELSnA = 0:0).

16.2.1.2 TPMCHn — TPM Channel n I/O Pin(s)

Each TPM channel is associated with an I/O pin on the MCU. The function of this pin depends on the channel configuration. The TPM pins share with general purpose I/O pins, where each pin has a port data register bit, and a data direction control bit, and the port has optional passive pullups which may be enabled whenever a port pin is acting as an input.

The TPM channel does not control the I/O pin when (ELSnB:ELSnA = 0:0) or when (CLKS = 00) so it normally reverts to general purpose I/O control. When CPWMS = 1 (and ELSnB:ELSnA not = 0:0), all channels within the TPM are configured for center-aligned PWM and the TPMCHn pins are all controlled by the TPM system. When CPWMS=0, the MSnB:MSnA control bits determine whether the channel is configured for input capture, output compare, or edge-aligned PWM.

When a channel is configured for input capture (CPWMS=0, MSnB:MSnA = 0:0 and ELSnB:ELSnA not = 0:0), the TPMCHn pin is forced to act as an edge-sensitive input to the TPM. ELSnB:ELSnA control bits determine what polarity edge or edges will trigger input-capture events. A synchronizer based on the bus clock is used to synchronize input edges to the bus clock. This implies the minimum pulse width—that can be reliably detected—on an input capture pin is four bus clock periods (with ideal clock pulses as near as two bus clocks can be detected). TPM uses this pin as an input capture input to override the port data and data direction controls for the same pin.

When a channel is configured for output compare (CPWMS=0, MSnB:MSnA = 0:1 and ELSnB:ELSnA not = 0:0), the associated data direction control is overridden, the TPMCHn pin is considered an output controlled by the TPM, and the ELSnB:ELSnA control bits determine how the pin is controlled. The remaining three combinations of ELSnB:ELSnA determine whether the TPMCHn pin is toggled, cleared, or set each time the 16-bit channel value register matches the timer counter.

When the output compare toggle mode is initially selected, the previous value on the pin is driven out until the next output compare event—then the pin is toggled.

When a channel is configured for edge-aligned PWM (CPWMS=0, MSnB=1 and ELSnB:ELSnA not = 0:0), the data direction is overridden, the TPMCHn pin is forced to be an output controlled by the TPM, and ELSnA controls the polarity of the PWM output signal on the pin. When ELSnB:ELSnA=1:0, the TPMCHn pin is forced high at the start of each new period (TPMCNT=0x0000), and the pin is forced low when the channel value register matches the timer counter. When ELSnA=1, the TPMCHn pin is forced low at the start of each new period (TPMCNT=0x0000), and the pin is forced high when the channel value register matches the timer counter.

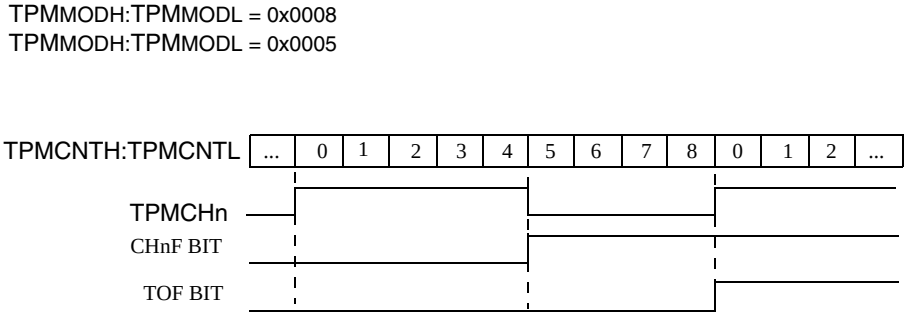


Figure 16-2. High-True Pulse of an Edge-Aligned PWM

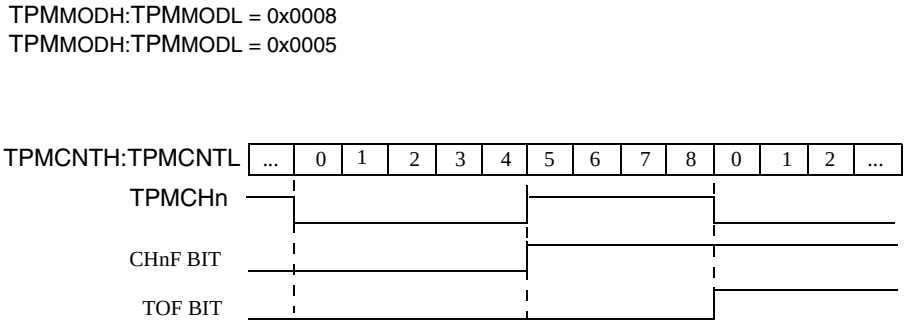


Figure 16-3. Low-True Pulse of an Edge-Aligned PWM

When the TPM is configured for center-aligned PWM (and ELSnB:ELSnA not = 0:0), the data direction for all channels in this TPM are overridden, the TPMCHn pins are forced to be outputs controlled by the TPM, and the ELSnA bits control the polarity of each TPMCHn output. If ELSnB:ELSnA=1:0, the corresponding TPMCHn pin is cleared when the timer counter is counting up, and the channel value register matches the timer counter; the TPMCHn pin is set when the timer counter is counting down, and the channel value register matches the timer counter. If ELSnA=1, the corresponding TPMCHn pin is set when the timer counter is counting up and the channel value register matches the timer counter; the TPMCHn pin is cleared when the timer counter is counting down and the channel value register matches the timer counter.

TPMMODH:TPMMODL = 0x0008
TPMMODH:TPMMODL = 0x0005

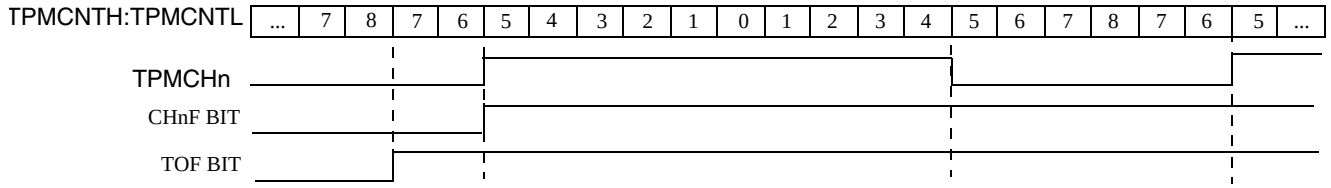


Figure 16-4. High-True Pulse of a Center-Aligned PWM

TPMMODH:TPMMODL = 0x0008
TPMMODH:TPMMODL = 0x0005

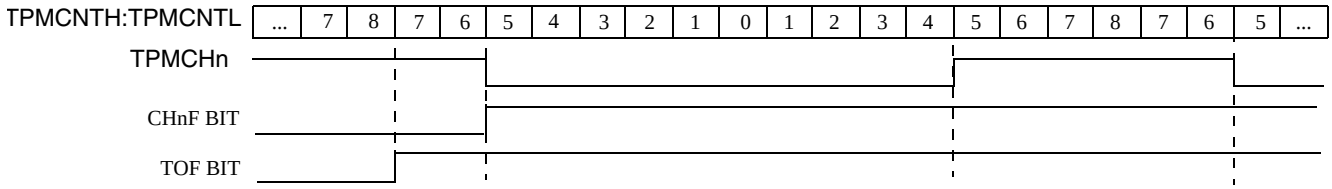


Figure 16-5. Low-True Pulse of a Center-Aligned PWM

16.3 Register Definition

This section consists of register descriptions in address order.

16.3.1 TPM Status and Control Register (TPMSC)

TPMSC contains the overflow status flag and control bits used to configure the interrupt enable, TPM configuration, clock source, and prescale factor. These controls relate to all channels within this timer module.

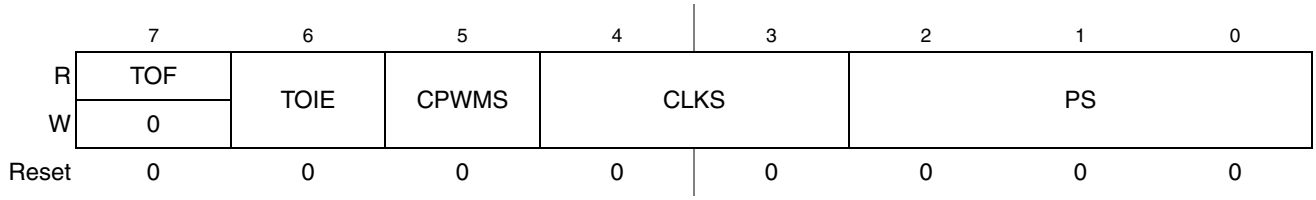


Figure 16-6. TPM Status and Control Register (TPMSC)

Table 16-2.

Field	Description
7 TOF	Timer overflow flag. This read/write flag is set when the TPM counter resets to 0x0000 after reaching the modulo value programmed in the TPM counter modulo registers. Clear TOF by reading the TPM status and control register when TOF is set and then writing a logic 0 to TOF. If another TPM overflow occurs before the clearing sequence is complete, the sequence is reset so TOF would remain set after the clear sequence was completed for the earlier TOF. This is done so a TOF interrupt request cannot be lost during the clearing sequence for a previous TOF. Reset clears TOF. Writing a logic 1 to TOF has no effect. 0 TPM counter has not reached modulo value or overflow 1 TPM counter has overflowed
6 TOIE	Timer overflow interrupt enable. This read/write bit enables TPM overflow interrupts. If TOIE is set, an interrupt is generated when TOF equals one. Reset clears TOIE. 0 TOF interrupts inhibited (use for software polling) 1 TOF interrupts enabled
5 CPWMS	Center-aligned PWM select. When present, this read/write bit selects CPWM operating mode. By default, the TPM operates in up-counting mode for input capture, output compare, and edge-aligned PWM functions. Setting CPWMS reconfigures the TPM to operate in up/down counting mode for CPWM functions. Reset clears CPWMS. 0 All channels operate as input capture, output compare, or edge-aligned PWM mode as selected by the MSnB:MSnA control bits in each channel's status and control register. 1 All channels operate in center-aligned PWM mode.
4–3 CLKS	Clock source selects. As shown in Table 16-3, this 2-bit field is used to disable the TPM system or select one of three clock sources to drive the counter prescaler. The fixed system clock source is only meaningful in systems with a PLL-based system clock. When there is no PLL, the fixed-system clock source is the same as the bus rate clock. The external source is synchronized to the bus clock by TPM module, and the fixed system clock source (when a PLL is present) is synchronized to the bus clock by an on-chip synchronization circuit. When a PLL is present but not enabled, the fixed-system clock source is the same as the bus-rate clock.
2–0 PS	Prescale factor select. This 3-bit field selects one of 8 division factors for the TPM clock input as shown in Table 16-4. This prescaler is located after any clock source synchronization or clock source selection so it affects the clock source selected to drive the TPM system. The new prescale factor will affect the clock source on the next system clock cycle after the new value is updated into the register bits.

Table 16-3. TPM-Clock-Source Selection

CLKS	TPM Clock Source to Prescaler Input
00	No clock selected (TPM counter disable)
01	Bus rate clock
10	Fixed system clock
11	External source

Table 16-4. Prescale Factor Selection

PS	TPM Clock Source Divided-by
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

16.3.2 TPM-Counter Registers (TPMCNTH:TPMCNTL)

The two read-only TPM counter registers contain the high and low bytes of the value in the TPM counter. Reading either byte (TPMCNTH or TPMCNTL) latches the contents of both bytes into a buffer where they remain latched until the other half is read. This allows coherent 16-bit reads in either big-endian or little-endian order which makes this more friendly to various compiler implementations. The coherency mechanism is automatically restarted by an MCU reset or any write to the timer status/control register (TPMSC).

Reset clears the TPM counter registers. Writing any value to TPMCNTH or TPMCNTL also clears the TPM counter (TPMCNTH:TPMCNTL) and resets the coherency mechanism, regardless of the data involved in the write.

	7	6	5	4	3	2	1	0
R	Bit 15	14	13	12	11	10	9	Bit 8
W	Any write to TPMCNTH clears the 16-bit counter							
Reset	0	0	0	0	0	0	0	0

Figure 16-7. TPM Counter Register High (TPMCNTH)

	7	6	5	4	3	2	1	0
R	Bit 7	6	5	4	3	2	1	Bit 0
W	Any write to TPMCNTL clears the 16-bit counter							
Reset	0	0	0	0	0	0	0	0

Figure 16-8. TPM Counter Register Low (TPMCNTL)

When BDM is active, the timer counter is frozen (this is the value that will be read by user); the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both counter halves are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it will read the appropriate value from the other half of the 16-bit value after returning to normal execution.

16.3.3 TPM Counter Modulo Registers (TPMMODH:TPMMODL)

The read/write TPM modulo registers contain the modulo value for the TPM counter. After the TPM counter reaches the modulo value, the TPM counter resumes counting from 0x0000 at the next clock, and the overflow flag (TOF) becomes set. Writing to TPMMODH or TPMMODL inhibits the TOF bit and overflow interrupts until the other byte is written. Reset sets the TPM counter modulo registers to 0x0000 which results in a free running timer counter (modulo disabled).

Writing to either byte (TPMMODH or TPMMODL) latches the value into a buffer and the registers are updated with the value of their write buffer according to the value of CLKS bits, so:

- If (CLKS[1:0] = 00), then the registers are updated when the second byte is written
- If (CLKS[1:0] not = 00), then the registers are updated after both bytes were written, and the TPM counter changes from (TPMMODH:TPMMODL - 1) to (TPMMODH:TPMMODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFE to 0xFFFF

The latching mechanism may be manually reset by writing to the TPMSC address (whether BDM is active or not).

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the modulo register are written while BDM is active. Any write to the modulo registers bypasses the buffer latches and directly writes to the modulo register while BDM is active.

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

Figure 16-9. TPM Counter Modulo Register High (TPMMODH)

	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

Reset the TPM counter before writing to the TPM modulo registers to avoid confusion about when the first counter overflow will occur.

16.3.4 TPM Channel n Status and Control Register (TPMCnSC)

TPMCnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

	7	6	5	4	3	2	1	0
R	CHnF	CHnIE	MSnB	MSnA	ELSnB	ELSnA	0	0
W	0							
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 16-11. TPM Channel n Status and Control Register (TPMCnSC)

Table 16-5.

Field	Description
7 CHnF	Channel n flag. When channel n is an input-capture channel, this read/write bit is set when an active edge occurs on the channel n pin. When channel n is an output compare or edge-aligned/center-aligned PWM channel, CHnF is set when the value in the TPM counter registers matches the value in the TPM channel n value registers. When channel n is an edge-aligned/center-aligned PWM channel and the duty cycle is set to 0% or 100%, CHnF will not be set even when the value in the TPM counter registers matches the value in the TPM channel n value registers. A corresponding interrupt is requested when CHnF is set and interrupts are enabled (CHnIE = 1). Clear CHnF by reading TPMCnSC while CHnF is set and then writing a logic 0 to CHnF. If another interrupt request occurs before the clearing sequence is complete, the sequence is reset so CHnF remains set after the clear sequence completed for the earlier CHnF. This is done so a CHnF interrupt request cannot be lost due to clearing a previous CHnF. Reset clears the CHnF bit. Writing a logic 1 to CHnF has no effect. 0 No input capture or output compare event occurred on channel n 1 Input capture or output compare event on channel n
6 CHnIE	Channel n interrupt enable. This read/write bit enables interrupts from channel n. Reset clears CHnIE. 0 Channel n interrupt requests disabled (use for software polling) 1 Channel n interrupt requests enabled
5 MSnB	Mode select B for TPM channel n. When CPWMS=0, MSnB=1 configures TPM channel n for edge-aligned PWM mode. Refer to the summary of channel mode and setup controls in Table 16-6.

Table 16-5.

Field	Description
4 MSnA	Mode select A for TPM channel n. When CPWMS=0 and MSnB=0, MSnA configures TPM channel n for input-capture mode or output compare mode. Refer to Table 16-6 for a summary of channel mode and setup controls. **Note:** If the associated port pin is not stable for at least two bus clock cycles before changing to input capture mode, it is possible to get an unexpected indication of an edge trigger.
3–2 ELSnB ELSnA	Edge/level select bits. Depending upon the operating mode for the timer channel as set by CPWMS:MSnB:MSnA and shown in Table 16-6, these bits select the polarity of the input edge that triggers an input capture event, select the level that will be driven in response to an output compare match, or select the polarity of the PWM output. Setting ELSnB:ELSnA to 0:0 configures the related timer pin as a general purpose I/O pin not related to any timer functions. This function is typically used to temporarily disable an input capture channel or to make the timer pin available as a general purpose I/O pin when the associated timer channel is set up as a software timer that does not require the use of a pin.

Table 16-6. Mode, Edge, and Level Selection

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	XX	00	Pin not used for TPM - revert to general purpose I/O or other peripheral control	
0	00	01	Input capture	Capture on rising edge only
		10		Capture on falling edge only
		11		Capture on rising or falling edge
	01	01	Output compare	Toggle output on compare
		10		Clear output on compare
		11		Set output on compare
	1X	10	Edge-aligned PWM	High-true pulses (clear output on compare)
		X1		Low-true pulses (set output on compare)
1	XX	10	Center-aligned PWM	High-true pulses (clear output on compare-up)
		X1		Low-true pulses (set output on compare-up)

16.3.5 TPM Channel Value Registers (TPMCnVH:TPMCnVL)

These read/write registers contain the captured TPM counter value of the input capture function or the output compare value for the output compare or PWM functions. The channel registers are cleared by reset.

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

Figure 16-12. TPM Channel Value Register High (TPMCnVH)

	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 16-13. TPM Channel Value Register Low (TPMCnVL)

In input capture mode, reading either byte (TPMCnVH or TPMCnVL) latches the contents of both bytes into a buffer where they remain latched until the other half is read. This latching mechanism also resets (becomes unlatched) when the TPMCnSC register is written (whether BDM mode is active or not). Any write to the channel registers will be ignored during the input capture mode.

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the channel register are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it will read the appropriate value from the other half of the 16-bit value after returning to normal execution. The value read from the TPMCnVH and TPMCnVL registers in BDM mode is the value of these registers and not the value of their read buffer.

In output compare or PWM modes, writing to either byte (TPMCnVH or TPMCnVL) latches the value into a buffer. After both bytes are written, they are transferred as a coherent 16-bit value into the timer-channel registers according to the value of CLKS bits and the selected mode, so:

- If (CLKS[1:0] = 00), then the registers are updated when the second byte is written.
- If (CLKS[1:0] not = 00 and in output compare mode) then the registers are updated after the second byte is written and on the next change of the TPM counter (end of the prescaler counting).
- If (CLKS[1:0] not = 00 and in EPWM or CPWM modes), then the registers are updated after the both bytes were written, and the TPM counter changes from (TPMMODH:TPMMODL - 1) to (TPMMODH:TPMMODL). If the TPM counter is a free-running counter then the update is made when the TPM counter changes from 0xFFFE to 0xFFFF.

The latching mechanism may be manually reset by writing to the TPMCnSC register (whether BDM mode is active or not). This latching mechanism allows coherent 16-bit writes in either big-endian or little-endian order which is friendly to various compiler implementations.

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active even if one or both halves of the channel register are written while BDM is active. Any write to the channel registers bypasses the buffer latches and directly write to the channel register while BDM is active. The values written to the channel register while BDM is active are used for PWM & output compare operation once normal execution resumes. Writes to the channel

registers while BDM is active do not interfere with partial completion of a coherency sequence. After the coherency mechanism has been fully exercised, the channel registers are updated using the buffered values written (while BDM was not active) by the user.

16.4 Functional Description

All TPM functions are associated with a central 16-bit counter which allows flexible selection of the clock source and prescale factor. There is also a 16-bit modulo register associated with the main counter.

The CPWMS control bit chooses between center-aligned PWM operation for all channels in the TPM (CPWMS=1) or general purpose timing functions (CPWMS=0) where each channel can independently be configured to operate in input capture, output compare, or edge-aligned PWM mode. The CPWMS control bit is located in the main TPM status and control register because it affects all channels within the TPM and influences the way the main counter operates. (In CPWM mode, the counter changes to an up/down mode rather than the up-counting mode used for general purpose timer functions.)

The following sections describe the main counter and each of the timer operating modes (input capture, output compare, edge-aligned PWM, and center-aligned PWM). Because details of pin operation and interrupt activity depend upon the operating mode, these topics will be covered in the associated mode explanation sections.

16.4.1 Counter

All timer functions are based on the main 16-bit counter (TPMCNTH:TPMCNTL). This section discusses selection of the clock source, end-of-count overflow, up-counting vs. up/down counting, and manual counter reset.

16.4.1.1 Counter Clock Source

The 2-bit field, CLKS, in the timer status and control register (TPMSC) selects one of three possible clock sources or OFF (which effectively disables the TPM). See Table 16-3. After any MCU reset, CLKS=00 so no clock source is selected, and the TPM is in a very low power state. These control bits may be read or written at any time and disabling the timer (writing 00 to the CLKS field) does not affect the values in the counter or other timer registers.

Table 16-7. TPM Clock Source Selection

CLKS	TPM Clock Source to Prescaler Input
00	No clock selected (TPM counter disabled)
01	Bus rate clock
10	Fixed system clock
11	External source

The bus rate clock is the main system bus clock for the MCU. This clock source requires no synchronization because it is the clock that is used for all internal MCU activities including operation of the CPU and buses.

In MCUs that have no PLL or the PLL is not engaged, the fixed system clock source is the same as the bus-rate-clock source, and it does not go through a synchronizer. When a PLL is present and engaged, a synchronizer is required between the crystal divided-by two clock source and the timer counter so counter transitions will be properly aligned to bus-clock transitions. A synchronizer will be used at chip level to synchronize the crystal-related source clock to the bus clock.

The external clock source may be connected to any TPM channel pin. This clock source always has to pass through a synchronizer to assure that counter transitions are properly aligned to bus clock transitions. The bus-rate clock drives the synchronizer; therefore, to meet Nyquist criteria even with jitter, the frequency of the external clock source must not be faster than the bus rate divided-by four. With ideal clocks the external clock can be as fast as bus clock divided by four.

When the external clock source shares the TPM channel pin, this pin should not be used for other channel timing functions. For example, it would be ambiguous to configure channel 0 for input capture when the TPM channel 0 pin was also being used as the timer external clock source. (It is the user's responsibility to avoid such settings.) The TPM channel could still be used in output compare mode for software timing functions (pin controls set not to affect the TPM channel pin).

16.4.1.2 Counter Overflow and Modulo Reset

An interrupt flag and enable are associated with the 16-bit main counter. The flag (TOF) is a software-accessible indication that the timer counter has overflowed. The enable signal selects between software polling (TOIE=0) where no hardware interrupt is generated, or interrupt-driven operation (TOIE=1) where a static hardware interrupt is generated whenever the TOF flag is equal to one.

The conditions causing TOF to become set depend on whether the TPM is configured for center-aligned PWM (CPWMS=1). In the simplest mode, there is no modulus limit and the TPM is not in CPWMS=1 mode. In this case, the 16-bit timer counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to 0x0000. When the TPM is in center-aligned PWM mode (CPWMS=1), the TOF flag gets set as the counter changes direction at the end of the count value set in the modulus register (that is, at the transition from the value set in the modulus register to the next lower count value). This corresponds to the end of a PWM period (the 0x0000 count value corresponds to the center of a period).

16.4.1.3 Counting Modes

The main timer counter has two counting modes. When center-aligned PWM is selected (CPWMS=1), the counter operates in up/down counting mode. Otherwise, the counter operates as a simple up counter. As an up counter, the timer counter counts from 0x0000 through its terminal count and then continues with 0x0000. The terminal count is 0xFFFF or a modulus value in TPMMODH:TPMMODL.

When center-aligned PWM operation is specified, the counter counts up from 0x0000 through its terminal count and then down to 0x0000 where it changes back to up counting. Both 0x0000 and the terminal count value are normal length counts (one timer clock period long). In this mode, the timer overflow flag (TOF) becomes set at the end of the terminal-count period (as the count changes to the next lower count value).

16.4.1.4 Manual Counter Reset

The main timer counter can be manually reset at any time by writing any value to either half of TPMCNTH or TPMCNTL. Resetting the counter in this manner also resets the coherency mechanism in case only half of the counter was read before resetting the count.

16.4.2 Channel Mode Selection

Provided CPWMS=0, the MSnB and MSnA control bits in the channel n status and control registers determine the basic mode of operation for the corresponding channel. Choices include input capture, output compare, and edge-aligned PWM.

16.4.2.1 Input Capture Mode

With the input-capture function, the TPM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input-capture channel, the TPM latches the contents of the TPM counter into the channel-value registers (TPMCnVH:TPMCnVL). Rising edges, falling edges, or any edge may be chosen as the active edge that triggers an input capture.

When either half of the 16-bit capture register is read, the other half is latched into a buffer to support coherent 16-bit accesses in big-endian or little-endian order. The coherency sequence can be manually reset by writing to the channel status/control register (TPMCnSC).

An input capture event sets a flag bit (CHnF) which may optionally generate a CPU interrupt request.

While in BDM, the input capture function works as configured by the user. When an external event occurs, the TPM latches the contents of the TPM counter (which is frozen because of the BDM mode) into the channel value registers and sets the flag bit.

16.4.2.2 Output Compare Mode

With the output-compare function, the TPM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter reaches the value in the channel-value registers of an output-compare channel, the TPM can set, clear, or toggle the channel pin.

In output compare mode, values are transferred to the corresponding timer channel registers only after both 8-bit halves of a 16-bit register have been written and according to the value of CLKS bits, so:

- If (CLKS[1:0] = 00), the registers are updated when the second byte is written
- If (CLKS[1:0] not = 00), the registers are updated at the next change of the TPM counter (end of the prescaler counting) after the second byte is written.

The coherency sequence can be manually reset by writing to the channel status/control register (TPMCnSC).

An output compare event sets a flag bit (CHnF) which may optionally generate a CPU-interrupt request.

16.4.2.3 Edge-Aligned PWM Mode

This type of PWM output uses the normal up-counting mode of the timer counter (CPWMS=0) and can be used when other channels in the same TPM are configured for input capture or output compare functions. The period of this PWM signal is determined by the value of the modulus register (TPMMODH:TPMMODL) plus 1. The duty cycle is determined by the setting in the timer channel register (TPMCnVH:TPMCnVL). The polarity of this PWM signal is determined by the setting in the ELSnA control bit. 0% and 100% duty cycle cases are possible.

The output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal (Figure 16-14). The time between the modulus overflow and the output compare is the pulse width. If ELSnA=0, the counter overflow forces the PWM signal high, and the output compare forces the PWM signal low. If ELSnA=1, the counter overflow forces the PWM signal low, and the output compare forces the PWM signal high.

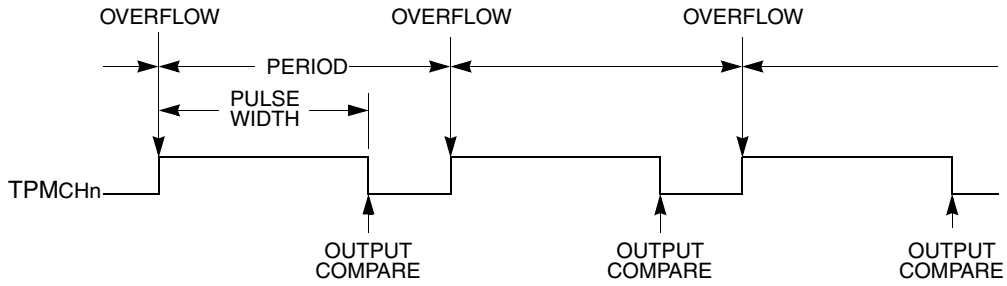


Figure 16-14. PWM Period and Pulse Width (ELSnA=0)

When the channel value register is set to 0x0000, the duty cycle is 0%. 100% duty cycle can be achieved by setting the timer-channel register (TPMCnVH:TPMCnVL) to a value greater than the modulus setting. This implies that the modulus setting must be less than 0xFFFF in order to get 100% duty cycle.

Because the TPM may be used in an 8-bit MCU, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPMCnVH and TPMCnVL, actually write to buffer registers. In edge-aligned PWM mode, values are transferred to the corresponding timer-channel registers according to the value of CLKS bits, so:

- If (CLKS[1:0] = 00), the registers are updated when the second byte is written
- If (CLKS[1:0] not = 00), the registers are updated after the both bytes were written, and the TPM counter changes from (TPMMODH:TPMMODL - 1) to (TPMMODH:TPMMODL). If the TPM counter is a free-running counter then the update is made when the TPM counter changes from 0xFFFE to 0xFFFF.

16.4.2.4 Center-Aligned PWM Mode

This type of PWM output uses the up/down counting mode of the timer counter (CPWMS=1). The output compare value in TPMCnVH:TPMCnVL determines the pulse width (duty cycle) of the PWM signal while the period is determined by the value in TPMMODH:TPMMODL. TPMMODH:TPMMODL should be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results. ELSnA will determine the polarity of the CPWM output.

pulse width = 2 x (TPMCnVH:TPMCnVL)

period = 2 x (TPMMODH:TPMMODL); TPMMODH:TPMMODL=0x0001-0x7FFF

If the channel-value register TPMCnVH:TPMCnVL is zero or negative (bit 15 set), the duty cycle will be 0%. If TPMCnVH:TPMCnVL is a positive value (bit 15 clear) and is greater than the (non-zero) modulus setting, the duty cycle will be 100% because the duty cycle compare will never occur. This implies the usable range of periods set by the modulus register is 0x0001 through 0x7FFE (0x7FF if you do not need to generate 100% duty cycle). This is not a significant limitation. The resulting period would be much longer than required for normal applications.

TPMMODH:TPMMODL=0x0000 is a special case that should not be used with center-aligned PWM mode. When CPWMS=0, this case corresponds to the counter running free from 0x0000 through 0xFFFF, but when CPWMS=1 the counter needs a valid match to the modulus register somewhere other than at 0x0000 in order to change directions from up-counting to down-counting.

The output compare value in the TPM channel registers (times 2) determines the pulse width (duty cycle) of the CPWM signal (Figure 16-15). If ELSnA=0, a compare occurred while counting up forces the CPWM output signal low and a compare occurred while counting down forces the output high. The counter counts up until it reaches the modulo setting in TPMMODH:TPMMODL, then counts down until it reaches zero. This sets the period equal to two times TPMMODH:TPMMODL.

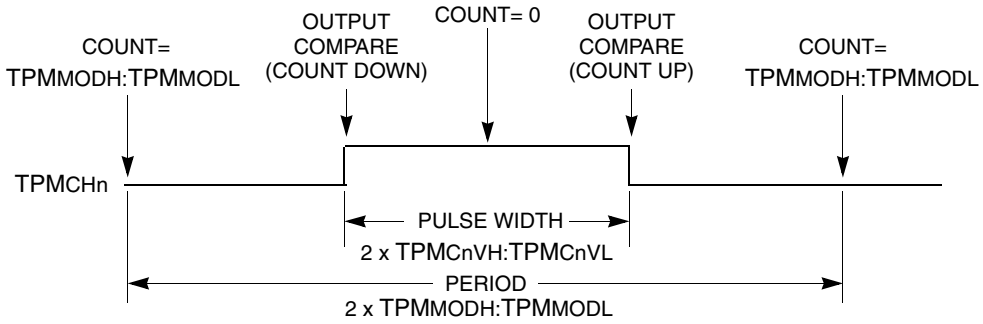


Figure 16-15. CPWM Period and Pulse Width (ELSnA=0)

Center-aligned PWM outputs typically produce less noise than edge-aligned PWMs because fewer I/O pin transitions are lined up at the same system clock edge. This type of PWM is also required for some types of motor drives.

Input capture, output compare, and edge-aligned PWM functions do not make sense when the counter is operating in up/down counting mode so this implies that all active channels within a TPM must be used in CPWM mode when CPWMS=1.

The TPM may be used in an 8-bit MCU. The settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPMMODH, TPMMODL, TPMCnVH, and TPMCnVL, actually write to buffer registers.

In center-aligned PWM mode, the TPMCnVH:L registers are updated with the value of their write buffer according to the value of CLKS bits, so:

- If (CLKS[1:0] = 00), the registers are updated when the second byte is written
- If (CLKS[1:0] not = 00), the registers are updated after the both bytes were written, and the TPM counter changes from (TPMMODH:TPMMODL - 1) to (TPMMODH:TPMMODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFE to 0xFFFF.

When TPMCNTH:TPMCNTL=TPMMODH:TPMMODL, the TPM can optionally generate a TOF interrupt (at the end of this count).

Writing to TPMSC cancels any values written to TPMMODH and/or TPMMODL and resets the coherency mechanism for the modulo registers. Writing to TPMCnSC cancels any values written to the channel value registers and resets the coherency mechanism for TPMCnVH:TPMCnVL.

16.5 Reset Overview

16.5.1 General

The TPM is reset whenever any MCU reset occurs.

16.5.2 Description of Reset Operation

Reset clears the TPMSC register which disables clocks to the TPM and disables timer overflow interrupts (TOIE=0). CPWMS, MSnB, MSnA, ELSnB, and ELSnA are all cleared which configures all TPM channels for input-capture operation with the associated pins disconnected from I/O pin logic (so all MCU pins related to the TPM revert to general purpose I/O pins).

16.6 Interrupts

16.6.1 General

The TPM generates an optional interrupt for the main counter overflow and an interrupt for each channel. The meaning of channel interrupts depends on each channel’s mode of operation. If the channel is configured for input capture, the interrupt flag is set each time the selected input capture edge is recognized. If the channel is configured for output compare or PWM modes, the interrupt flag is set each time the main timer counter matches the value in the 16-bit channel value register.

All TPM interrupts are listed in Table 16-8 which shows the interrupt name, the name of any local enable that can block the interrupt request from leaving the TPM and getting recognized by the separate interrupt processing logic.

Table 16-8. Interrupt Summary

Interrupt	Local Enable	Source	Description
TOF	TOIE	Counter overflow	Set each time the timer counter reaches its terminal count (at transition to next count value which is usually 0x0000)
CHnF	CHnIE	Channel event	An input capture or output compare event took place on channel n

The TPM module will provide a high-true interrupt signal. Vectors and priorities are determined at chip integration time in the interrupt module so refer to the user's guide for the interrupt module or to the chip's complete documentation for details.

16.6.2 Description of Interrupt Operation

For each interrupt source in the TPM, a flag bit is set upon recognition of the interrupt condition such as timer overflow, channel-input capture, or output-compare events. This flag may be read (polled) by software to determine that the action has occurred, or an associated enable bit (TOIE or CHnIE) can be set to enable hardware interrupt generation. While the interrupt enable bit is set, a static interrupt will generate whenever the associated interrupt flag equals one. The user's software must perform a sequence of steps to clear the interrupt flag before returning from the interrupt-service routine.

TPM interrupt flags are cleared by a two-step process including a read of the flag bit while it is set (1) followed by a write of zero (0) to the bit. If a new event is detected between these two steps, the sequence is reset and the interrupt flag remains set after the second step to avoid the possibility of missing the new event.

16.6.2.1 Timer Overflow Interrupt (TOF) Description

The meaning and details of operation for TOF interrupts varies slightly depending upon the mode of operation of the TPM system (general purpose timing functions versus center-aligned PWM operation). The flag is cleared by the two step sequence described above.

16.6.2.1.1 Normal Case

Normally TOF is set when the timer counter changes from 0xFFFF to 0x0000. When the TPM is not configured for center-aligned PWM (CPWMS=0), TOF gets set when the timer counter changes from the terminal count (the value in the modulo register) to 0x0000. This case corresponds to the normal meaning of counter overflow.

16.6.2.1.2 Center-Aligned PWM Case

When CPWMS=1, TOF gets set when the timer counter changes direction from up-counting to down-counting at the end of the terminal count (the value in the modulo register). In this case the TOF corresponds to the end of a PWM period.

16.6.2.2 Channel Event Interrupt Description

The meaning of channel interrupts depends on the channel's current mode (input-capture, output-compare, edge-aligned PWM, or center-aligned PWM).

16.6.2.2.1 Input Capture Events

When a channel is configured as an input capture channel, the ELSnB:ELSnA control bits select no edge (off), rising edges, falling edges or any edge as the edge which triggers an input capture event. When the selected edge is detected, the interrupt flag is set. The flag is cleared by the two-step sequence described in Section 16.6.2, "Description of Interrupt Operation."

16.6.2.2.2 Output Compare Events

When a channel is configured as an output compare channel, the interrupt flag is set each time the main timer counter matches the 16-bit value in the channel value register. The flag is cleared by the two-step sequence described Section 16.6.2, "Description of Interrupt Operation."

16.6.2.2.3 PWM End-of-Duty-Cycle Events

For channels configured for PWM operation there are two possibilities. When the channel is configured for edge-aligned PWM, the channel flag gets set when the timer counter matches the channel value register which marks the end of the active duty cycle period. When the channel is configured for center-aligned PWM, the timer count matches the channel value register twice during each PWM cycle. In this CPWM case, the channel flag is set at the start and at the end of the active duty cycle period which are the times when the timer counter matches the channel value register. The flag is cleared by the two-step sequence described Section 16.6.2, "Description of Interrupt Operation."

Chapter 17
Independent Real Time Clock (IRTC)

17.1 Introduction

The IRTC, independent real time clock, provides the functionality of a basic RTC like time keeping and calendaring and additionally provides protection against tampering, spurious memory/register updates and battery operation. It can additionally compensate the 1 Hz clock against variations in 32 kHz clock in oscillator due to crystal or temperature. A standby RAM is provided if the CPU wants to store any data that has to be retained when in battery operation mode.

NOTE

The IRTC Configuration Data Register (IRTC_CFG_DATA) bits [7:1] are reserved in MCF51EM256 family devices, and must be written as 0's. Bit 0 (CFG0) is used as the configuration bit to control the address mapping of the two flash blocks. Refer to section Section 3.4.2, “Dual Flash Controllers.”

MCF51EM256 series have only one tamper pin. Further reference to a different number of tamper pins must be disregarded.

The IRTC features a protection mechanism to protect against spurious writes into the IRTC registers by any run-away code. On power-on reset a 15 s window is allowed for the CPU to configure the IRTC after which the registers are locked. Refer to Section 17.4, “Overview," for details.

The IRTC tamper interrupt is enabled after reset (IRTC_IER register, bit TMPR = 1). A POR generates a tamper interrupt request by setting the bit TMPR in the IRTC_ISR register.

Year & Month Alarm (IRTC_ALM_YRMON), Days Alarm (IRTC_ALM_DAYS), Hours and Minutes Alarm (IRTC_ALM_HM), Seconds Alarm (IRTC_ALM_SEC), Status (IRTC_STATUS), Interrupt Status (IRTC_ISR) bits [15:1], Interrupt Enable (IRTC_IER) bits [15:1] and Configuration Data Registers (IRTC_CFG_DATA) are reset by IRTC soft reset or IRTC power-on reset. All other registers are reset by IRTC power-on reset. IRTC power-on reset occurs when V_{DD} or V_{bat} applies a voltage above the POR rearm voltage.

17.1.1 IRTC Power Supply Source

The IRTC power supply source depends on the MCU operation mode and the LVD configuration.

IRTC power supply with the MCU in Run, Wait Modes[1] or stop4[2]

- If (V_{DD} > V_{LVDH}), the IRTC is powered by V_{DD} pin
- If (V_{DD} < V_{LVDH}) and (V_{DD} > V_{LVDL}), the IRTC is powered by V_{DD} and V_{BAT} pins
- If (V_{DD} < V_{LVDL}), the IRTC is power by the V_{BAT} pin

IRTC power supply with the MCU in LPRun, LPWait, stop3 or stop2[3]

- The IRTC always operates from V_{BAT}

1. The IRTC power supply in Run and Wait Modes is independent of the LVD configuration.
2. Stop4 is entered upon execution of the Stop instruction with the LVD enabled or the BDM enabled.
3. In LPRun, LPWait, stop3 and stop2, the LVD is disabled.

17.2 Features

The IRTC supports the following features:

- Full clock – hour, minutes and seconds with option for storing values in BCD or binary format
- Calendaring – day, month, year and day of the week with option for storing values in BCD or binary format
- Auto adjustment for day light saving with user defined parameters
- Automatic month and leap year adjustment
- RTC utilizes ‘local time’ which implicitly contains the time zone offset
- Programmable alarm with interrupt. Alarm is output from IRTC in case MCU wants to use it as a wakeup event
- Seven periodic interrupts
- Minute countdown timer with minute resolution
- 32.768 kHz input clock with option to output the clock for use in the MCU’s ICS
- Hardware compensation to compensate 1 Hz clock (to the counters) against frequency variations in oscillator clock due to temperature or crystal characteristics. Correction factor calculated by firmware. (Programmable correction factor)
- IPS bus interface with protection against run-away code
- Reset to the IRTC block is generated only when both battery supply and CPU power are removed and either is powered up
- Battery operation (standby mode) ensures seamless IRTC operation when CPU power is removed
- Tamper detection to detect illegal access into the system
- Time stamp stored on tamper event
- Various configurability options available as explained in section below
- 32 bytes of standby RAM

17.3 Block Diagram

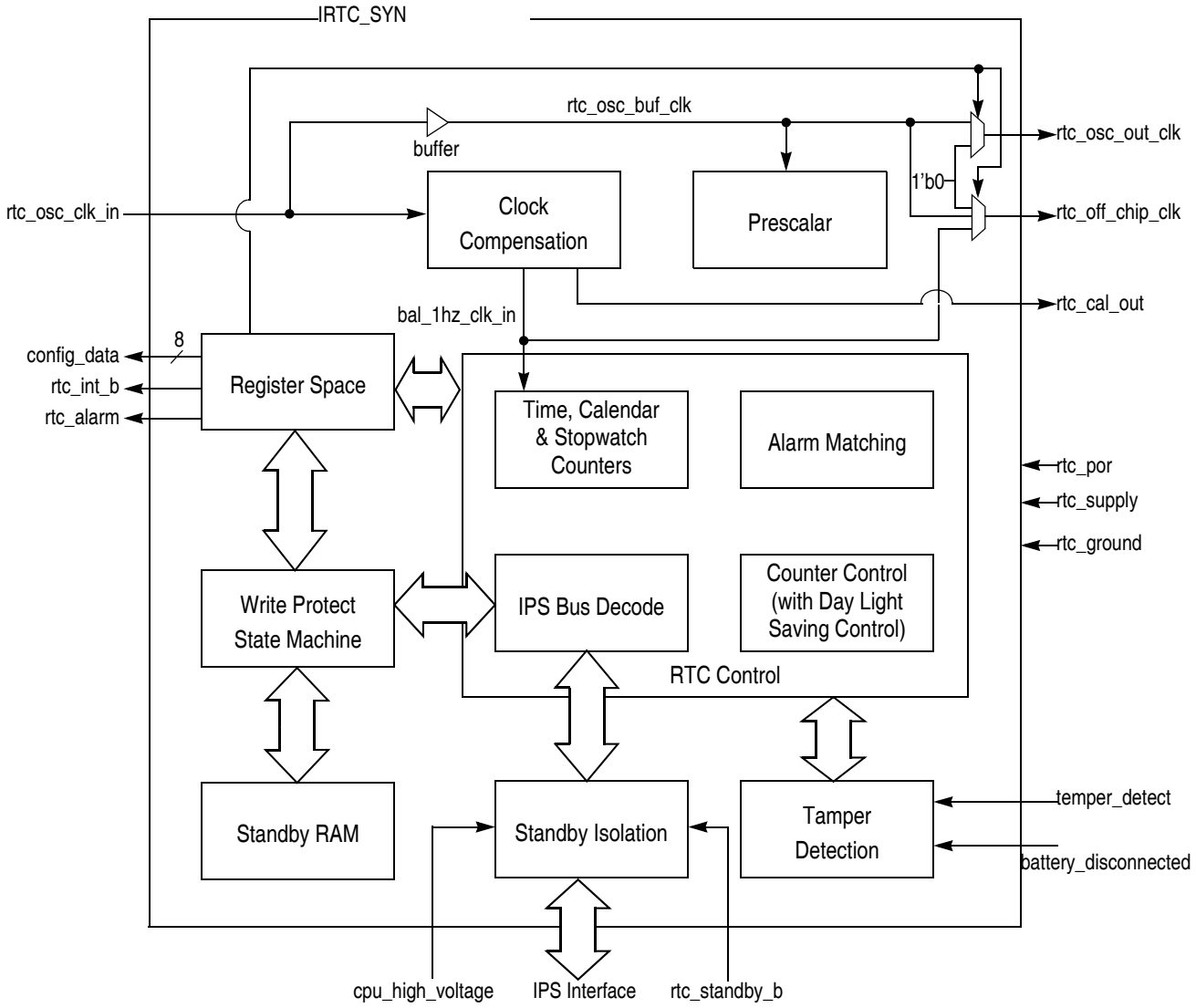


Figure 17-1. IRTC Block Diagram

17.4 Overview

The IRTC block provides basic time keeping functions by second, minute and hour counters and calendaring functions via date, day-of-week, month and year counters; with automatic adjustment for leap year and day light saving. Reading these counters indicates the current date and time and writing to these registers sets the date and time given by the user. The user has the option of reading and writing into time and date related registers in either BCD or Binary format.

The alarm is set for specific hour, minute and second. When the time counters match the alarm hour, minute and second setting, the alarm flag is set and an interrupt to CPU is generated if the alarm interrupt is enabled. The alarm can additionally be configured to match days, months and year to generate the alarm interrupt. The alarm signal has been brought out from IRTC for use by MCU to allow certain wakeup events.

A countdown timer with minute resolution is also provided for time keeping applications. This counter can be enabled or disabled separately. An interrupt is generated at the expiry of the counter. IRTC module also provides seven sampling timer interrupts apart from normal interrupts for alarm and countdown timeout.

A frequency compensation module is integrated into the IRTC to correct any error in 1 Hz clock due to variations in the 32 kHz clock caused by crystal inaccuracy, board variations or change in temperature. The compensation value for both crystal and temperature variation is set by software and correction is done in hardware.

The registers in the IRTC are programmable via the IPS bus. A protection mechanism is built-in the IRTC to protect against spurious writes into the IRTC by any run-away code. The protection mechanism requires the CPU to write a specific sequence of codes to the IRTC_CTRL[1:0] bits in the control register that allows write access to the registers. On completing the update of registers the CPU writes any value to IRTC_CTRL[1:0] bits to enable the write protection. After unlocking the registers, the CPU has a window of two seconds for updating the register space. On power on reset a window of 15 seconds is allowed for the CPU to configure the IRTC after which the registers are locked. Any further updates would require the CPU to unlock the registers.

The IRTC battery supply maintains normal RTC functionality when the CPU power is removed. The battery supply allows IRTC to keep functioning in case CPU is completely turned off. Reset to the IRTC block is generated only when both battery supply and CPU power are removed and either is powered up. The reset generation and switching of power is done external to the IRTC by an analog switch/regulator.

The IRTC is also equipped with a RAM that will be powered by the battery supply in the event of main supply being switched off. The size of this memory is 32 bytes. MCU can use this RAM to store any data it wants to retain in case the MCU power is switched off. This RAM will loose all its contents when both MCU and battery power have been removed.

IRTC can detect any intrusion via its tamper detection mechanism that will get enabled automatically after the calibration of IRTC is complete. For this purpose, a pin has been provided and high level input on this pin indicates a tamper which will get stored in the IRTC registers. Removal of battery after calibration will also be indicated as a tamper. Any tamper detected will cause an interrupt to the CPU. IRTC can also store the time and date stamp of the latest tamper event recorded. Tamper detection and its interrupt are enabled on reset.

IRTC has a 32-bit up-counter register that keeps incrementing on writes and read to this register returns the latest count value. This register will be of use in metering kind of applications where this register can be used to store the energy consumed over a period of time.

Figure 17-1 shows the block diagram for IRTC.

17.5 IRTC Programming Model

The IRTC contains twenty four 16-bit aligned registers and standby RAM. The table below summarized the registers and their address with access type. Both the registers and the standby RAM can be accessed via 8-bit and 16-bit read/write cycles.

Table 17-1. Register Map

Address	Register Name	Access	Write Protect
0x00	Month and Year Counter Register	read/write	all bits
0x02	Days and Day-of-Week Counter Register	read/write	all bits
0x04	Hour and Minutes Counter Register	read/write	all bits
0x06	Seconds Counter Register	read/write	all bits
0x08	Month and Year Alarm Register	read/write	all bits
0x0A	Days Alarm Register	read/write	all bits
0x0C	Hour and Minutes Alarm Register	read/write	all bits
0x0E	Seconds Alarm Register	read/write	all bits
0x10	Control Register	read/write	all bits except WE [1:0]
0x12	Status Register	read-only	n/a
0x14	Interrupt Status Register	read/write	all bits
0x16	Interrupt Enable Register	read/write	all bits
0x18	Countdown (Minutes) Timer Register	read/write	all bits
0x1A	RESERVED	n/a	n/a
0x1C	RESERVED	n/a	n/a
0x1E	RESERVED	n/a	n/a
0x20	Configuration Data Register	read/write	all bits
0x22	Daylight Saving Hour Register	read/write	all bits
0x24	Daylight Saving Month Register	read/write	all bits
0x26	Daylight Saving Day Register	read/write	all bits
0x28	Compensation Register	read/write	all bits
0x2A	Tamper Time Stamp Month & Year Register	read-only	n/a
0x2C	Tamper Time Stamp Days Register	read-only	n/a
0x2E	Tamper Time Stamp Hours & Minutes Register	read-only	n/a
0x30	Tamper Time Stamp Seconds Register	read-only	n/a
0x32	Count Up Register (upper word)	read-only	all bits
0x34	Count Up Register (lower word)	read/write	all bits
0x36	RESERVED	n/a	n/a
0x38	RESERVED	n/a	n/a
0x3A	RESERVED	n/a	n/a
0x3C	RESERVED	n/a	n/a

Table 17-1. Register Map (continued)

Address	Register Name	Access	Write Protect
0x3E	RESERVED	n/a	n/a
0x40 to 0x5F	Standby RAM.	read/write	all bits

All registers except the write enable bits in the Control Register are protected from spurious updated by any run-away code. The above registers will be accessible in the module based on the feature selection set by system integrator. The register map does not change but registers pertaining to unselected features will be considered as reserved. Writes will have no effect and reads will return zeros.

Write access to reserved locations & to registers during write protect enable mode will generate transfer error and read access to reserved locations will generate transfer error and the read data bus will show all 1s.

NOTE

IRTC does not check for the correctness of values programmed into its registers; hence programming illogical time & date entries will result in an undefined operation. Normal functionality is not guaranteed in that case.

Below is a detailed description of all the above registers. Each register has been explained using a register diagram showing the bits and their reset value with access type. This is followed by a detailed description of each bit/bit-field along with valid values for each field. Where ever applicable, fields have been shown in both binary and BCD.

17.5.1 IRTC Year & Month Counters Register (IRTC_YEARMON)

IRTC_YEARMON — Year & Month Counters Register — Offset: 0x00

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field (bin)	YEAR								0x00
Field (bcd)	Value remains in binary as year indicates offset from base year								
Bit Type	rw	rw	rw	rw	rw	rw	rw	rw	
Reset Value	0	0	0	0	0	0	0	0	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field (bin)					MONTH				0x01
Field (bcd)	MONTH (tens)				MONTH (units)				
Bit Type	rsvd	rsvd	rsvd	rsvd	rw	rw	rw	rw	
Reset Value	—	—	—	—	0	0	0	1	

Figure 17-2. IRTC Year & Month Counters Register (IRTC_YEARMON)

Table 17-2. IRTC_YEARMON Field Descriptions

Field	Description
15–8 YEAR	Year count value. Indicates the offset in years from the base year (hard coded as 2112) and does not show the actual year value. This is a signed value. Valid Values: –128 to 127 (Base year 2112 and if the value of YEAR bits is 0x10, the actual year will be 2112 + 16 = 2128).
7:4	Reserved bits in binary mode. Read returns zeros. In BCD mode, these are used.
3–0 MONTH	Month Count Value. Valid Values: 1 – 12 ‘1’ – January ‘7’ – July ‘0’ & ‘13’ to ‘15’ – Reserved. Should not be used. ‘2’ – February ‘8’ – August ‘3’ – March ‘9’ – September ‘4’ – April ‘10’ – October ‘5’ – May ‘11’ – November ‘6’ – June ‘12’ – December

This register stores the value of the month and year counters. The year bits do not store the year value but calculate the increment in years. This is a signed value and the range of values is from –128 to +127. The software programs the offset from that base year into this register. The BASE YEAR has been hard coded in the design as the year 2112. For example, if the current year is 2007, then this will be represented in this register as –105 or 0x97. The actual year value can be found by adding the BASE YEAR and the offset in the IRTC_YEARMON[15:8] register. Hence the range of year supported will be 1984 (2112 – 128) to 2239 (2112 + 127). Hence for year calculation we have:

Actual year = Base year (i.e. 2112) + Offset year

The month register stores the count value of the months register. Writing to this register loads the months counter with this new value. The valid values are mentioned in table above. Writing any other value will not guarantee correct operation of IRTC. Both month and year are unaffected on software reset.

In BCD mode, the value of months counter is represented in BCD format and the bits used are shown in the table above. Since the year value is a 2’s complement value this is not converted to BCD but the value remains in binary even in BCD mode.

MCU should first read the INVAL bit of the IRTC_STATUS (bit 0) to determine that the counters are stable and they can be changed. The INVAL bit ensures that no operation is done at the boundary of a second when counters change value. MCU should check the status of this bit both during reads and writes.

17.5.2 IRTC Day & Day-of-Week Counters Register (IRTC_DAYS)

IRTC_DAYS — Days & Day-of-Week Counters Register — Offset: 0x02

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field (bin)						DAY_OF_WEEK			0x02
Field (bcd)						Value remains unchanged			
Bit Type	rsvd	rsvd	rsvd	rsvd	rsvd	rw	rw	rw	
Reset Value	—	—	—	—	—	0	0	0	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field (bin)				DAYS					0x03
Field (bcd)	DAYS (tens)				DAYS (units)				
Bit Type	rsvd	rsvd	rsvd	rw	rw	rw	rw	rw	
Reset Value	—	—	—	0	0	0	0	1	

Figure 17-3. IRTC Day & Day-of-Week Counters Register (IRTC_DAYS)

Table 17-3. IRTC_DAYS Field Descriptions

Field	Description
15–11	Reserved bits. Read returns zeros.
10–8 DAY_OF_WEEK	Day of the Week Count. ‘0’ – Sunday ‘4’ – Thursday ‘1’ – Monday ‘5’ – Friday ‘2’ – Tuesday ‘6’ – Saturday ‘3’ – Wednesday ‘7’ – Reserved. Should not be used
7–5	Reserved bits in Binary mode. Read returns zeros. In BCD mode, these are used.
4–0 DAYS	Days counter value. Valid Count: 1 – 31

This read/write register contains the current value of the day-of-week counter and day counter. This register can be read at any time without affecting the counter count values. Writing to this register loads the value to the day-of-week and day counter and the counters continue to count from this new value. This register unaffected on software reset.

In the BCD mode, the days counter value gets converted in BCD format as shown in table, while the day-of-week counter remains unchanged as the value is same in binary and in BCD format. This is controlled by the bit in IRTC_CTRL register.

MCU should first read the INVAL bit of the IRTC_STATUS (bit 0) to determine that the counters are stable and they can be changed. The INVAL bit ensures that no operation is done at the boundary of a second when counters change value. MCU should check the status of this bit both during reads and writes.

17.5.3 IRTC Hours and Minutes Counter Register (IRTC_HOURMIN)

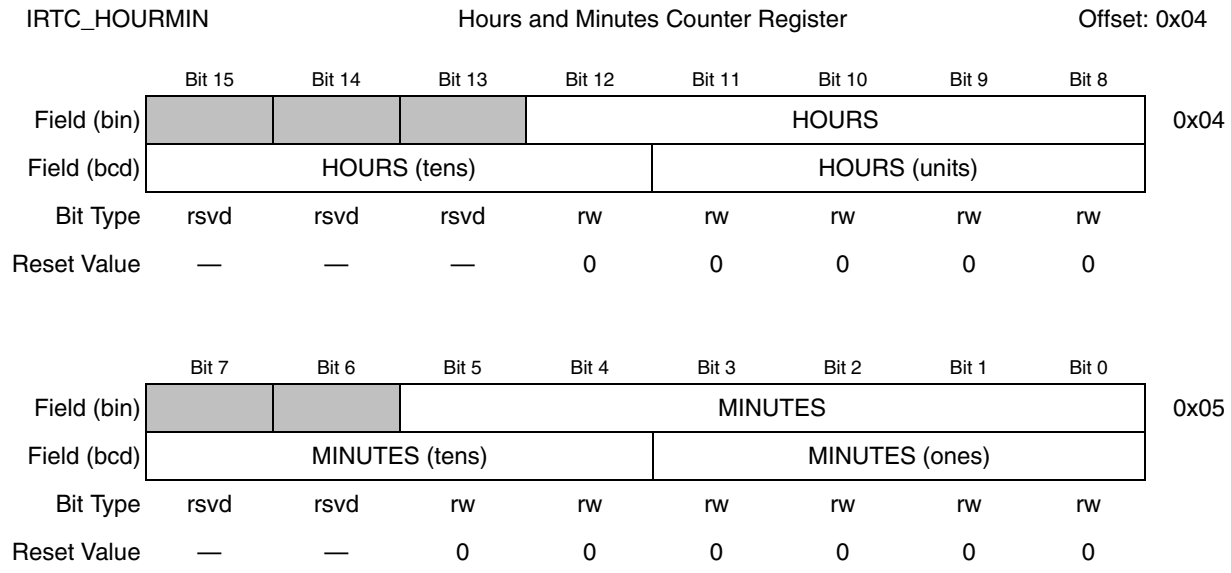


Figure 17-4. IRTC Hours and Minutes Counter Register (IRTC_HOURMIN)

Table 17-4. IRTC_HOURMIN Field Descriptions

Field	Description
15–13	Reserved bits in Binary mode. Read returns zeros. In BCD mode, these are used.
12–8 HOURS	Hour Counter Value. Count: 0 – 23. 0 – 11 is AM and 12 – 23 is PM.
7–6	Reserved bits in Binary mode. Read returns zeros. In BCD mode, these are used.
5–0 MINUTES	Minutes Counter Value. Count: 0 – 59

This register is used to program the hours and minutes counter. It can be read anytime to get the current value of the counters. Only power-on reset can reset this register. Hours counter can be set anything between 0 and 23 both included. Minutes counter can be set anything between 0 and 59 both included. This register is unaffected by software reset.

MCU should first read the INVAL bit of the IRTC_STATUS (bit 0) to determine that the counters are stable and they can be changed. The INVAL bit ensures that no operation is done at the boundary of a second when counters change value. MCU should check the status of this bit both during reads and writes.

17.5.4 IRTC Seconds Counter Register (IRTC_SECONDS)

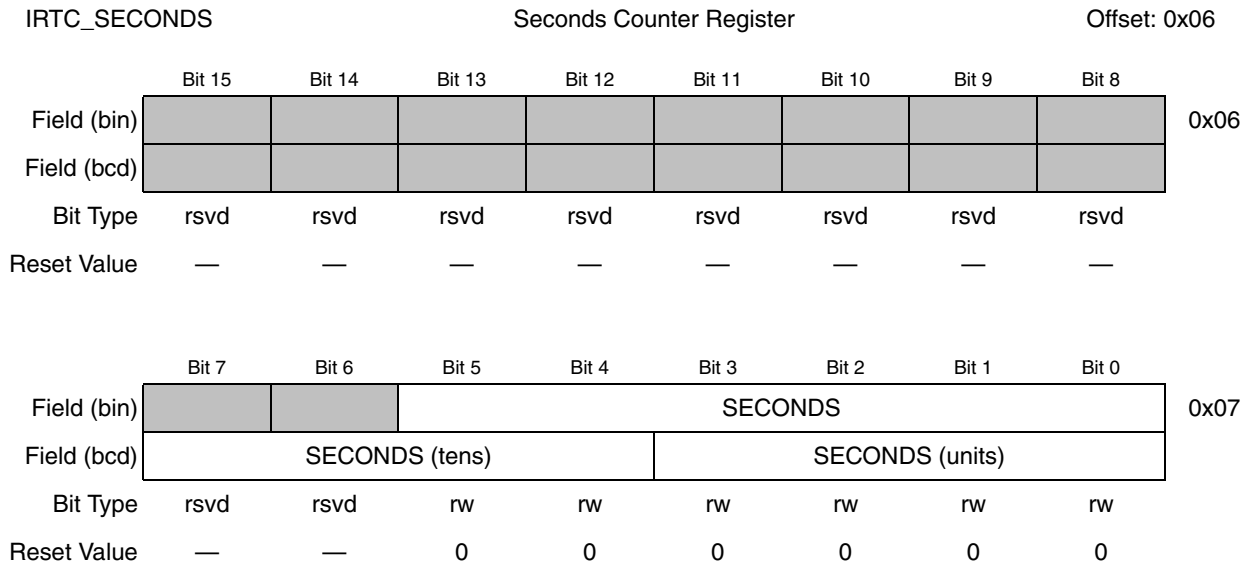


Figure 17-5. IRTC Seconds Counter Register (IRTC_SECONDS)

Table 17-5. IRTC_SECONDS Field Description

Field	Description
15–8	Reserved bits. Not writeable. Read returns zeros.
7–6	Reserved bits in Binary mode. Read returns zeros. In BCD mode, these are used.
5–0 SECONDS	Seconds Counter Value. Count: 0 – 59

This register is used to program the seconds counter. It can be read anytime to get the current value of the counter. Only power-on reset can reset this register. Seconds counter can be set anything between 0 and 59 both included. This register is unaffected by software reset.

MCU should first read the INVAL bit of the IRTC_STATUS (bit 0) to determine that the counters are stable and they can be changed. The INVAL bit ensures that no operation is done at the boundary of a second when counters change value. MCU should check the status of this bit both during reads and writes.

17.5.5 IRTC Year & Month Alarm Register (IRTC_ALM_YRMON)

IRTC_ALM_YRMON — Year and Month Alarm Register — Offset: 0x08

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field (bin)	ALM_YEAR								0x08
Field (bcd)	Value remains in binary as year indicates offset from base year								
Bit Type	rw	rw	rw	rw	rw	rw	rw	rw	
Reset Value	0	0	0	0	0	0	0	0	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field (bin)					ALM_MONTH				0x09
Field (bcd)	ALM_MONTHS (tens)				ALM_MONTHS (units)				
Bit Type	rsvd	rsvd	rsvd	rsvd	rw	rw	rw	rw	
Reset Value	—	—	—	—	0	0	0	0	

Figure 17-6. IRTC Year & Month Alarm Register (IRTC_ALM_YRMON)

Table 17-6. IRTC_ALM_YRMON Field Descriptions

Field	Description
15–8 ALM_YEAR	Year Count Value. Indicates the offset in years from the base year (hard coded as 2112) and does not show the actual year value. This is a signed value. Valid Values: –128 to +127 (Base Year 2112 and if the value of YEAR bits is 0x10, the actual year will be 2112 + 16 = 2128).
7–4	Reserved bits in Binary mode. Read returns zeros. In BCD mode, these are used.
3–0 ALM_MONTH	Month Alarm Value. Valid Values: 1 – 12 ‘1’ – January ‘2’ – February ‘3’ – March ‘4’ – April ‘5’ – May ‘6’ – June ‘7’ – July ‘8’ – August ‘9’ – September ‘10’ – October ‘11’ – November ‘12’ – December ‘13’ to ‘15’ – Reserved. Should not be used. ‘0’ – No match done.

The month and year alarm register is used to configure the month and year setting of the alarm. The alarm setting can be read or written anytime. This register is reset to its default state on software reset. Alarm interrupt bit is set when all values of alarm seconds, minutes, hours, days, month and year match their respective counter values.

17.5.6 IRTC Days Alarm Register (IRTC_ALM_DAYS)

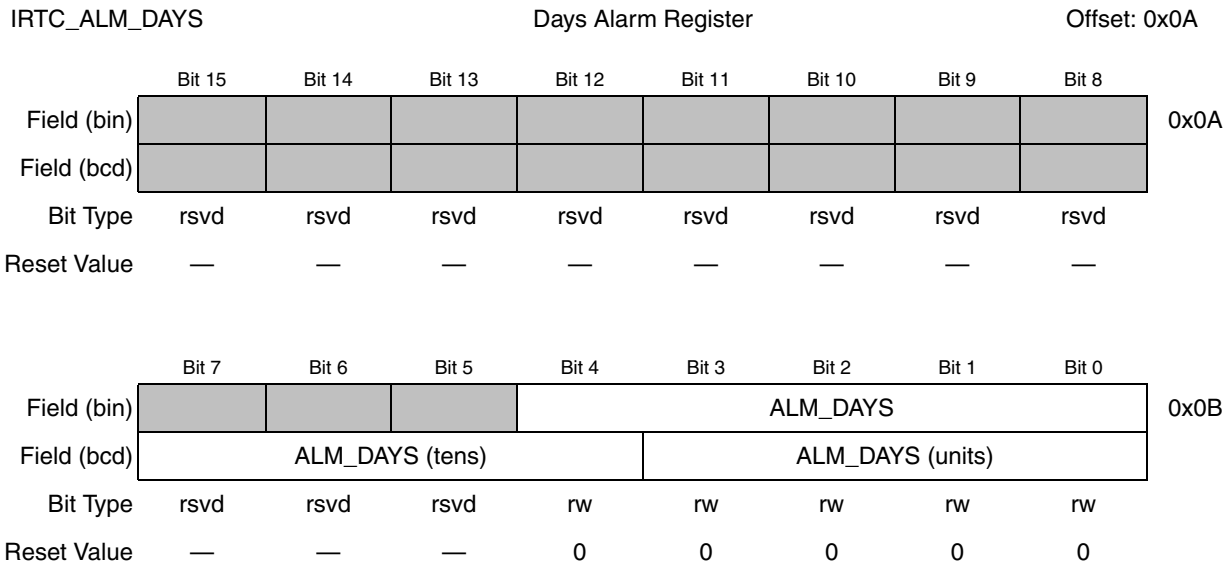


Figure 17-7. IRTC Days Alarm Register (IRTC_ALM_DAYS)

Table 17-7. IRTC_ALM_DAYS Field Descriptions

Field	Description
15–8	Reserved bits. Read returns zeros.
7–5	Reserved bits in Binary mode. Read returns zeros. In BCD mode, these are used.
4–0 ALM_DAYS	Days Alarm value. Valid Values: 1 – 31

The days alarm register is used to configure the day setting of the alarm. The alarm setting can be read or written anytime. This register is reset to its default state on software reset. Alarm interrupt bit is set when all values of alarm seconds, minutes, hours, days, month and year match their respective counter values.

17.5.7 IRTC Hours and Minutes Alarm Register (IRTC_ALM_HM)

IRTC_ALM_HM — Hours and Minutes Alarm Register — Offset: 0x0C

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field (bin)				ALM_HOURS					0x0C
Field (bcd)	ALM_HOURS (tens)				ALM_HOURS (units)				
Bit Type	rsvd	rsvd	rsvd	rw	rw	rw	rw	rw	
Reset Value	—	—	—	0	0	0	0	0	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field (bin)			ALM_MINUTES						0x0D
Field (bcd)	ALM_MINUTES (tens)				ALM_MINUTES (units)				
Bit Type	rsvd	rsvd	rw	rw	rw	rw	rw	rw	
Reset Value	—	—	0	0	0	0	0	0	

Figure 17-8. IRTC Hours and Minutes Alarm Register (IRTC_ALM_HM)

Table 17-8. IRTC_ALM_HM Field Descriptions

Field	Description
15–13	Reserved bits in Binary mode. Read returns zeros. In BCD mode, these are used.
12–8 ALM_HOURS	Alarm Hour Value. Count: 0 – 23
7–6	Reserved bits in Binary mode. Read returns zeros. In BCD mode, these are used.
5–0 ALM_MINUTES	Alarm Minutes Value. Count: 0 – 59

The hours and minutes alarm register is used to configure the hour and minute setting of the alarm. The alarm setting can be read or written anytime. This register is reset to default state on software reset. In BCD mode, all 8-bits are used to indicate the time.

17.5.8 IRTC Seconds Alarm Register (IRTC_ALM_SEC)

IRTC_ALM_SEC — Seconds Alarm Register — Offset: 0x0E

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field (bin)							INC_S	DEC_S	0x0E
Field (bcd)							INC_S	DEC_S	
Bit Type	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	sclr	sclr	
Reset Value	—	—	—	—	—	—	0	0	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field (bin)			ALM_SECONDS						0x0F
Field (bcd)	ALM_SECONDS (tens)				ALM_SECONDS (units)				
Bit Type	rsvd	rsvd	rw	rw	rw	rw	rw	rw	
Reset Value	—	—	0	0	0	0	0	0	

Figure 17-9. IRTC Seconds Alarm Register (IRTC_ALM_SEC)

Table 17-9. IRTC_ALM_SEC Field Descriptions

Field	Description
15–10	Reserved bits. Not writeable. Read returns zeros.
9 INC_S	This bit controls the increment of seconds counter incase the MCU wants to make corrections to the seconds counter to compensate for the leap seconds or to perform fine trimming of time when needed. Write to this bit has increments the seconds counter and then the bit gets cleared on next posedge.
8 DEC_S	This bit controls the decrement of seconds counter incase the MCU wants to make corrections to the seconds counter to compensate for the leap seconds or to perform fine trimming of time when needed. Write to this bit has decrements the seconds counter and then the bit gets cleared on next posedge.
7–6	Reserved bits in Binary mode. Read returns zeros. In BCD mode, these are used.
5–0 ALM_SECONDS	Alarm Seconds Value. Count: 0 – 59

The seconds alarm register is used to configure the seconds setting of the alarm. The alarm setting can be read or written anytime. This register is reset to default value on software reset. In BCD mode, all 8-bits are used to indicate the time.

Bits 9 & 8 provide option to the MCU to perform correction on seconds counter to compensate for the leap seconds. Write to these bits adds or subtracts 1 from the seconds counter and read returns zeros. MCU should first read the INVAL bit of the IRTC_STATUS (bit 0) to determine that the counters are stable and they can be incremented. The INVAL bit ensures that no operation is done at the boundary of a second when counters change value. MCU should check the status of this bit both during reads and writes.

17.5.9 IRTC Control Register (IRTC_CTRL)

IRTC_CTRL — IRTC Control Register — Offset: 0x10

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field	OCLK	OFF_CHIP_CLK		TAMPER DETECT DURATION				SWR	0x10
Bit Type	rw	rw	rw	rw	rw	rw	rw	rw/sclr	
Reset Value	1	0	0	0	0	0	0	0	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field	BCDEN	DSTEN			ALARM MATCH		WE1	WE0	0x11
Bit Type	rw	rw	rsvd	rsvd	rw	rw	rw/sclr	rw/sclr	
Reset Value	0	0	—	—	0	0	0	0	

Figure 17-10. IRTC Control Register (IRTC_CTRL)

Table 17-10. IRTC_CTRL Field Descriptions

Field	Description
15	Oscillator Clock Output Enable. This bit controls the buffered IRTC oscillator clock to be output from the IRTC block. This is mainly needed for 8-bit MCUs and 32-bit Coldfire V1s where it is used by the ICS block of the MCU. ‘1’ – Clock Output Enabled ‘0’ – Clock Output Disabled
14–13 OFF_CHIP_CLK	Off-chip Clock Output Enable. Needed mostly for 8-bit MPUs. This bit controls the clock to be output from the IRTC block for external use. “00” – No clock is output “01” – Compensated 1 Hz clock output “10” – Buffered Oscillator clock output “11” – Reserved. Do not use.
12–9 TAMPER DETECT DURATION	Tamper Detect Duration. This bit indicates the number of oscillator clocks for which the tamper_detect signal should remains stable before being detected as a tamper. These bits are used by the tamper filtering operation. “0” – Filtering operation disabled “1” to “15” – number of 32kHz oscillator clocks to be counted when tamper is asserted With the tamper duration set to Zero any tamper detected on the tamper pins will directly set the tamper status & interrupt bits. Caution is required when making the tamper filter duration equal to 0 as any glitches on the tamper pins will cause a tamper interrupt.
8 SWR	Software Reset. This is a self clearing bit. It automatically gets cleared on the next clock after write. ‘1’ – Software Reset ‘0’ – No Software Reset Software reset clears the contents of alarm, interrupt (status & enable except tamper) registers and has no effect on DST, Standby RAM, Up counter, time, calendaring and tamper detect registers.
7 BCDEN	BCD format enable. This bit controls whether the read/write value of time and date registers will be in binary or BCD. The registers are maintained in binary and the conversion to BCD to binary or vice-versa is done for the read/write IPS bus. It is done for some registers only. ‘1’ – Registers read for Hour, Minutes, Seconds and Day displayed in BCD format ‘0’ – Registers read for Hour, Minutes, Seconds and Day displayed in Binary format

Table 17-10. IRTC_CTRL Field Descriptions

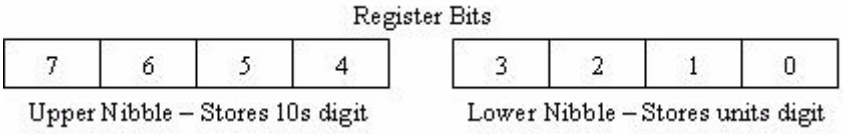
Field	Description
6 DSTEN	Daylight Saving Enable. Automatic adjustment of time is done when enabled. '1' – Enabled. Daylight Saving changes are applied. '0' – Disabled. Daylight Saving changes are NOT applied.
5–4	Reserved bits. Not writeable. Read returns zeros.
3–2 ALARM MATCH	Alarm Match bits. These bits define which time and calendar counters will be used for matching and generate an alarm "00" – Only Seconds, Minutes and Hours matched "01" – Only Seconds, Minutes, Hours and Days matched "10" – Only Seconds, Minutes, Hours, Days and Months matched "11" – Only Seconds, Minutes, Hours, Days, Months and Year (offset) matched
1–0 WE	Write Enable bits. Controls the entry and exit into/from the Register/Memory write protection mode. Self clearing bits. Reads will return zeros. '10' → Enable Write Protection – Registers are locked '00 – 01 – 11 – 10' ◊ Disable Write Protection – Registers are unlocked **NOTE:** When the registers are unlocked, they remain in this unlocked state for a time of 1 to 2 seconds after which the registers are automatically locked. After power-on-reset too, the registers come out as unlocked but they get locked automatically 14 to 15 seconds after power on.

This is the control register and governs all operations being done inside the IRTC. This register is used to specify the software reset, compensation controls, tamper controls and write protection control. Details of compensation and tamper logic are given in the Section 17.8, "Block Description (Brief Overview)."

There is no enable bit for the IRTC as this block is functional after reset and cannot be disabled. In Basic & Standard Feature sets, only Bits 15 & 8 are valid and other bits are reserved.

The write protect bits are the only bits that are freely writeable by CPU. Rest of all bits, registers and RAM are protected via a write protect mechanism. The write protect bits WE[1:0] are self clearing bits that always return zeros on read. The above mentioned sequence should be written into these bits to enable or disable write protection.

Depending on the BCD format enable bit, the width of time (seconds, minutes and hours) and day related registers changes and the values in the register becomes as follows:



The table below shows how data gets changed on reads. The reverse happens for writes. Data input is in BCD and converted to binary before storing in appropriate registers.

Table 17-11. Register Changes for BCD Format

Register Name	BCD Enable = '0'	BCD Enable = '1'
	Values stored in Binary	Values stored in BCD.
IRTC_HOURMIN	Bits 12:8 – Hour Bits 5:0 – Minutes	Bits 15:8 – Hour Bits 7:0 – Minutes

Table 17-11. Register Changes for BCD Format

IRTC_SECONDS	Bits 5:0 – Seconds	Bits 7:0 – Seconds
IRTC_DAYS	Bits 4:0 – Days	Bits 7:0 – Days
IRTC_MONYEAR	Bits 3:0 – Months	Bits 7:0 – Months
IRTC_ALM_HOURMIN	Bits 12:8 – Hour Bits 5:0 – Minutes	Bits 15:8 – Hour Bits 7:0 – Minutes
IRTC_ALM_SECONDS	Bits 5:0 – Seconds	Bits 7:0 – Seconds
IRTC_ALM_DAYS	Bits 4:0 – Days	Bits 7:0 – Days
IRTC_ALM_MONYR	Bits 3:0 – Months	Bits 7:0 – Months
IRTC_DST_HOUR	Bits 12:8 – Hour Start Bits 4:0 – Hour End	Bits 15:8 – Hour Start Bits 7:0 – Hour End
IRTC_DST_DAY	Bits 12:8 – Day Start Bits 4:0 – Day End	Bits 15:8 – Day Start Bits 7:0 – Day End
IRTC_DST_MONTH	Bits 11:8 – Months Start Bits 3:0 – Months End	Bits 11:8 – Months Start Bits 3:0 – Months End
IRTC_TTSR_HM	Bits 12:8 – Hour Bits 5:0 – Minutes	Bits 15:8 – Hour Bits 7:0 – Minutes
IRTC_TTSR_SEC	Bits 5:0 – Seconds	Bits 7:0 – Seconds
IRTC_COUNT_DN	Bits 5:0 – Minutes	Bits 7:0 – Minutes

17.5.10 IRTC Status Register (IRTC_STATUS)

IRTC_STATUS — Status Register — Offset: 0x12

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field									0x12
Bit Type	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	
Reset Value	—	—	—	—	—	—	—	—	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field			CLV	WPE	OCAL	BERR	C_DON	INVAL	0x13
Bit Type	rsvd	rsvd	ro	ro	ro	rw1c	rw1c	ro	
Reset Value	—	—	0	0	0	0	0	0	

Figure 17-11. IRTC Status Register (IRTC_STATUS)

Table 17-12. IRTC_STATUS Feild Descriptions

Field	Description
15–6	Reserved bits. Not writeable. Read returns zeros.

Table 17-12. IRTC_STATUS Feild Descriptions

5 CLV	CPU Low Voltage. This read-only bit is asserted when the MCU/CPU power falls below the read only threshold when all write cycles are terminated normally and no change is done to the registers.This bit is read-only and self-cleared. It is set when the V_{DD} is lower than $V_{lvwl\text{-}falling}$ (– 2.33 V), and automatically cleared when the V_{DD} is higher than $V_{lvwl\text{-}rising}$ (–2.41 V). When this bit is set, the IRTC registers are not writable.
4 WPE	Write Protect Enable. Indicates that registers are in locked mode and write to registers is disabled. Any write access made to the register space is ignored when write protection is enabled (i.e. registers in locked mode).
3 OCAL	Calibration Output Bit or Compensation Interval Bit. This status bit is asserted for a time equal to compensation interval seconds. This bit is used by MCU to calculate the interrupts serviced during this interval and perform corrections in case of deviations. This bit toggles on every compensation interval start and is either 0 or 1 during the entire duration. This bit will not toggle if compensation logic has been disabled by MCU. This bit is same as the irtc_cal_out signal output from IRTC.
2 BERR	Bus Error. Indicates that a read or write cycle was started by MCU when the INVAL bit is set. Write access to time/date registers gets nullified (terminate normally) and no register value gets changed. Read during INVAL bit asserted returns 16'hFFFF. No Transfer Error is asserted.
1 C_DON	Compensation Done. Read only bit gets cleared by writing 1 to it. '1' – Compensation Completed '0' – Compensation busy or not enabled Done bit is asserted a few oscillator cycles before actual compensation interval compensation so that back to back compensation can be enabled
0 INVAL	Invalid Time bit. Indicates the time is invalid or changing and should not be read. This bit is asserted 1 oscillator clock cycle before and after the 1 Hz (seconds') boundary. Not cleared on read. Write access to time/date registers gets nullified (terminate normally) and no register value gets changed. Read during INVAL bit asserted returns 16'hFFFF. No Transfer Error is asserted. '1' – Counter values changing. Time/Date is invalid '0' – Time can be read. Time is valid

This register indicates the status of various processes going inside the IRTC. Status of crystal or temperature compensation can be obtained. This register also helps the MCU to read time or date register when their values are stable and not changing. Software reset resets the register to its default state. Done bits get cleared on read.

17.5.11 IRTC Interrupt Status Register (IRTC_ISR)

IRTC_ISR — Interrupt Status Register — Offset: 0x14

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0	0x14
Bit Type	rw1c	rw1c	rw1c	rw1c	rw1c	rw1c	rw1c	rw1c	
Reset Value	0	0	0	0	0	0	0	0	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field	2Hz	1Hz	MIN	HR	DAY	ALM	STW	TMPR	0x15
Bit Type	rw1c	rw1c	rw1c	rw1c	rw1c	rw1c	rw1c	rw1c	
Reset Value	0	0	0	0	0	0	0	1	

Figure 17-12. IRTC Interrupt Status Register (IRTC_ISR)

Table 17-13. IRTC_ISR Field Descriptions

Field	Description
15 SAM7	Sampling Timer Interrupt Flag 7; indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 512 Hz
14 SAM6	Sampling Timer Interrupt Flag 6; indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 256 Hz
13 SAM5	Sampling Timer Interrupt Flag 5; indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 128 Hz
12 SAM4	Sampling Timer Interrupt Flag 4; indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 64 Hz
11 SAM3	Sampling Timer Interrupt Flag 3; indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 32 Hz
10 SAM2	Sampling Timer Interrupt Flag 2; indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 16 Hz
9 SAM1	Sampling Timer Interrupt Flag 1; indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 8 Hz
8 SAM0	Sampling Timer Interrupt Flag 0; indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 4 Hz
7 2Hz	Sampling Timer Interrupt at 2 Hz frequency, indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 2 Hz
6 1Hz	1 Hz Flag. Indicates that the seconds counter has incremented
5 MIN	Minutes Flag. Indicates that the minutes counter has incremented
4 HR	Hour Flag. Indicates that the hour counter has incremented

Table 17-13. IRTC_ISR Field Descriptions

Field	Description
3 DAY	Day Flag. Indicates that the day counter has incremented
2 ALM	Alarm Flag. Indicates that the alarm value programmed matches the counter values. This interrupt is generated only when the seconds, minutes, hours and days counter match the alarm register values
1 STW	Countdown Timer Expiry Flag. Indicates that the programmed count in the countdown timer has expired
0 TMPR	Tamper Detect Interrupt, indicates that a tamper has been done to the system and security is compromised. Since POR is considered as a tamper, the reset value is 1.

The real-time clock interrupt status register (IRTC_ISR) indicates the status of the various real-time clock interrupts. When an event of the types included in this register occurs then the bit will be set in this register regardless of its corresponding interrupt enable bit being set. The status bits are cleared by writing a value of 1, which also clears the interrupt. Interrupts may occur while the system clock is idle or in standby mode. When the system enters the active power mode, interrupt will be indicated to the CPU.

Tamper interrupt status is set on reset as POR is generated when battery and CPU power are unavailable and either is powered up. Removal of battery is considered as a tamper and hence this bit is set on reset.

The status register is also cleared on software reset except for the tamper status which remains unaffected.

17.5.12 IRTC Interrupt Enable Register (IRTC_IER)

IRTC_IER — Interrupt Enable Register — Offset: 0x16

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0	0x16
Bit Type	rw	rw	rw	rw	rw	rw	rw	rw	
Reset Value	0	0	0	0	0	0	0	0	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field	2Hz	1Hz	MIN	HR	DAY	ALM	STW	TMPR	0x17
Bit Type	rw	rw	rw	rw	rw	rw	rw	rw	
Reset Value	0	0	0	0	0	0	0	1	

Figure 17-13. IRTC Interrupt Enable Register (IRTC_IER)

Table 17-14. IRTC_IER Field Descriptions

Field	Description
15 SAM7	Sampling Timer Interrupt Flag 7, enable/disable interrupt ‘1’ – Enable ‘0’ – Disable
14 SAM6	Sampling Timer Interrupt Flag 6, enable/disable interrupt ‘1’ – Enable ‘0’ – Disable
13 SAM5	Sampling Timer Interrupt Flag 5, enable/disable interrupt ‘1’ – Enable ‘0’ – Disable
12 SAM4	Sampling Timer Interrupt Flag 4, enable/disable interrupt ‘1’ – Enable ‘0’ – Disable
11 SAM3	Sampling Timer Interrupt Flag 3, enable/disable interrupt ‘1’ – Enable ‘0’ – Disable
10 SAM2	Sampling Timer Interrupt Flag 2, enable/disable interrupt ‘1’ – Enable ‘0’ – Disable
9 SAM1	Sampling Timer Interrupt Flag 1, enable/disable interrupt ‘1’ – Enable ‘0’ – Disable
8 SAM0	Sampling Timer Interrupt Flag 0, enable/disable interrupt ‘1’ – Enable ‘0’ – Disable
7 2Hz	Sampling Timer Interrupt at 2 Hz frequency, enable/disable interrupt ‘1’ – Enable ‘0’ – Disable
6 1Hz	1 Hz Flag, enable disable interrupt ‘1’ – Enable ‘0’ – Disable
5 MIN	Minutes Flag, enable disable interrupt ‘1’ – Enable ‘0’ – Disable
4 HR	Hour Flag, enable disable interrupt ‘1’ – Enable ‘0’ – Disable
3 DAY	Day Flag, enable disable interrupt ‘1’ – Enable ‘0’ – Disable
2 ALM	Alarm Flag, enable disable interrupt ‘1’ – Enable ‘0’ – Disable

Table 17-14. IRTC_IER Field Descriptions

Field	Description
1 STW	Countdown Timer Expiry Flag, enable disable interrupt ‘1’ – Enable ‘0’ – Disable
0 TMPR	Tamper Detect Interrupt, enable disable interrupt ‘1’ – Enable ‘0’ – Disable **NOTE:** This interrupt gets enabled on POR and is unaffected by software reset.

The real-time clock interrupt enable register (IRTC_IER) is used to enable/disable the various real-time clock interrupts. Disabling an interrupt bit has no effect on its corresponding status bit.

Alarm interrupt is set as per the following table.

Table 17-15. Alarm Match Table

Register Bits (IRTC_CTRL[3:2])	Counters Matched	Alarm Type
00	Seconds, Minutes and Hours matched	Daily
01	Seconds, Minutes, Hours and Days matched	Monthly
10	Seconds, Minutes, Hours, Days and Months matched	Yearly
11	Seconds, Minutes, Hours, Days, Months & Year (offset) matched	One-Time

A single interrupt (ipi_irtc_int_b) is output from the IP which is an ORed version of all the above interrupt. The CPU should read the status register in the interrupt status routine to determine which interrupt has occurred.

The tamper detect interrupt enable is an exception as it is enabled after POR. All interrupts enables except tamper interrupt enable are reset to default state on software reset.

17.5.13 IRTC Countdown (Minutes) Timer Register (IRTC_COUNT_DN)

IRTC_COUNT_DN — Countdown (Minutes) Timer Register — Offset: 0x18

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field (bin)									0x18
Field (bcd)									
Bit Type	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	
Reset Value	—	—	—	—	—	—	—	—	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field (bin)	COUNTDOWN_COUNT								0x19
Field (bcd)	COUNTDOWN_COUNT (tens)				COUNTDOWN _COUNT (units)				
Bit Type	rw	rw	rw	rw	rw	rw	rw	rw	
Reset Value	0	0	0	0	0	0	0	0	

Figure 17-14. IRTC Countdown (Minutes) Timer Register (IRTC_COUNT_DN)

Table 17-16. IRTC_COUNT_DN Field Descriptions

Field	Description
15–8	Reserved Bits. Read returns zeros.
7–0 COUNTDOWN_COUNT	Countdown counter value. Valid count values: 1 – 99.

This counter is used to generate an interrupt on a minute boundary, for example, to turn off the LCD controller after five minutes of inactivity. The countdown timer is decremented by the minute (MIN) tick output from the real-time clock, so the average tolerance of the count is 0.5 minutes. For better accuracy, enable the countdown timer by waiting for the MIN bit of the IRTC_ISR register to be set (i.e. waiting for the minute interrupt). Loading the countdown timer counter with 0 will have no effect. Software reset brings the countdown timer count to its default state. Interrupt is generated when the timer reaches 0. The value in the register shows the current value of the counter at all times. Note that the actual delay includes the seconds from setting the countdown to the next minute tick.

The max value is 99 as in BCD mode, a value bigger than 99 cannot be displayed in 8-bits. Programming a value bigger than 99 does not ensure correct reading in BCD mode. However binary value can still be read.

17.5.14 IRTC Configuration Data Register (IRTC_CFG_DATA)

IRTC_CFG_DATA — Configuration Data Register — Offset: 0x20

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field									0x20
Bit Type	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	
Reset Value	—	—	—	—	—	—	—	—	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field								CFG0	0x21
Bit Type	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rw	
Reset Value	0	0	0	0	0	0	0	0	

Figure 17-15. IRTC Configuration Data Register (IRTC_CFG_DATA)

Table 17-17. IRTC_CFG_DATA Field Descriptions

Field	Description
15–1	Reserved bits. Read returns zeros.Bit7-Bit1 are reserved for Nucleus family devices, should be written as 0's.
0	Flash array select bit — Used as the configuration bit to control the address mapping of the two flash blocks. Refer to Section 3.4.2, “Dual Flash Controllers,” for details.

17.5.15 IRTC Daylight Saving Hour Register (IRTC_DST_HOUR)

IRTC_DST_HOUR — Daylight Saving Hour Register — Offset: 0x22

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field (bin)				DST_START_HOUR					0x22
Field (bcd)	DST_START_HOUR (tens)			DST_START_HOUR (units)					
Bit Type	rsvd	rsvd	rsvd	rw	rw	rw	rw	rw	
Reset Value	—	—	—	0	0	0	0	0	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field (bin)				DST_END_HOUR					0x23
Field (bcd)	DST_END_HOUR (tens)			DST_END_HOUR (units)					
Bit Type	rsvd	rsvd	rsvd	rw	rw	rw	rw	rw	
Reset Value	—	—	—	0	0	0	0	0	

Figure 17-16. IRTC Daylight Saving Hour Register (IRTC_DST_HOUR)

Table 17-18. IRTC_DST_HOUR Field Descriptions

Field	Description
15–13	Reserved bits in Binary mode. Read returns zeros. In BCD mode, these are used.
12–8 DST_START_HOUR	Hour value when Daylight saving has to start. Valid values: 0 – 23
7–5	Reserved bits in Binary mode. Read returns zeros. In BCD mode, these are used.
4–0 DST_END_HOUR	Hour value when Daylight saving has to end. Valid values: 0 – 23

This register stores the time in hours when the Daylight Saving has to be applied or reversed. The CPU should program the correct hour value (0 – 23) as per the regional settings. For example, if the Daylight Saving starts at 1:00 AM on March 25 and ends at 1:00 AM on October 28 in 2007 then the time at which time advances or falls back is acutally 1:59 AM. Hence the CPU should program 1 for the hour count value (and not 2!) i.e. write 0x0101 in this register. 59 minute count is automatically checked inside IRTC and hence not required to be programmed. This register has no effect of software reset.

In BCD mode, all eight bits are used to indicate the value of hours in both fields.

17.5.16 IRTC Daylight Saving Month Register (IRTC_DST_MNTH)

IRTC_DST_MNTH — Daylight Saving Month Register — Offset: 0x24

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field (bin)					DST_START_MONTH				0x24
Field (bcd)	DST_START_MONTH (tens)				DST_START_MONTH (units)				
Bit Type	rsvd	rsvd	rsvd	rsvd	rw	rw	rw	rw	
Reset Value	—	—	—	—	0	0	0	0	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field (bin)					DST_END_MONTH				0x25
Field (bcd)	DST_END_MONTH (tens)				DST_END_MONTH (units)				
Bit Type	rsvd	rsvd	rsvd	rsvd	rw	rw	rw	rw	
Reset Value	—	—	—	—	0	0	0	0	

Figure 17-17. IRTC Daylight Saving Month Register (IRTC_DST_MNTH)

Table 17-19. IRTC_DST_MNTH Field Descriptions

Field	Description
15–12	Reserved bits. Read returns zeros.
11–8 DST_START_MONTH	Month value when Daylight saving has to start. Valid values: 0 – 12

Table 17-19. IRTC_DST_MNTH Field Descriptions

7–4	Reserved bits. Read returns zeros.
3–0 DST_END_MONTH	Month value when Daylight saving has to end. Valid values: 0–12

This register stores the month when the Daylight Saving has to be applied or reversed. The CPU should program the correct month value (1–12) as per the regional settings. For example, if the daylight saving starts at March 25 and ends at October 28 in 2007. Hence the CPU should write 0x030A in this register. This register has no effect of software reset.

17.5.17 IRTC Daylight Saving Day Register (IRTC_DST_DAY)

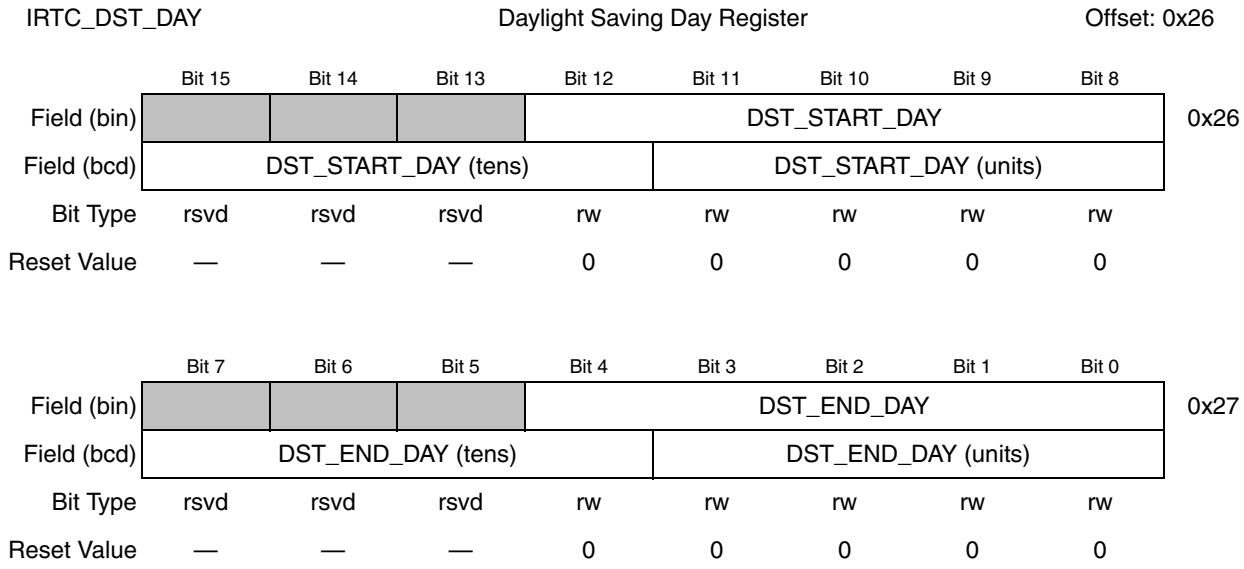


Figure 17-18. IRTC Daylight Saving Day Register (IRTC_DST_DAY)

Table 17-20. IRTC_DST_DAY Field Descriptions

Field	Description
15–12	Reserved bits in Binary mode. Read returns zeros. In BCD mode, these are used.
11–8 DST_START_DAY	Day value when Daylight saving has to start. Valid values: 1 – 31
7–4	Reserved bits in Binary mode. Read returns zeros. In BCD mode, these are used.
3–0 DST_END_DAY	Day value when Daylight saving has to end. Valid values: 1 – 31

This register stores the day when the Daylight Saving has to be applied or reversed. The CPU should program the correct day value (1 – 31) as per the regional settings. For example, if the Daylight Saving starts at March 25 and ends at October 28 in 2007. Hence the CPU should write 0x191C in this register. This register is unaffected by software reset. In BCD mode, all 8-bits are used to indicate the day value.

17.5.18 IRTC Compensation Register (IRTC_COMPEN)

IRTC_COMPEN — Compensation Register — Offset: 0x28

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field	COMPENSATION_INTERVAL								0x28
Bit Type	rw	rw	rw	rw	rw	rw	rw	rw	
Reset Value	0	0	0	0	0	0	0	0	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field	COMPENSATION_VALUE								0x29
Bit Type	rw	rw	rw	rw	rw	rw	rw	rw	
Reset Value	0	0	0	0	0	0	0	0	

Figure 17-19. IRTC Compensation Register (IRTC_COMPEN)

Table 17-21. IRTC_COMPEN Field Descriptions

Field	Description
15–8	Compensation Interval. Indicates the window over which the compensation has to be carried out. Minimum interval is 1 second and maximum is 255 seconds. A compensation interval of 0 disables the compensation logic. MCU can simply write zeros to these bits to disable the compensation logic. Non-zero value starts the compensation logic. Assertion of “Done” bit clears this register.
7–0	Compensation value. Two’s complement number which indicates the number of Oscillator clock cycles the IRTC requires to compensate for the specified compensation interval. Range: –128 to +127. A value of zero indicates no compensation is needed.

The compensation register stores the compensation value that will be used by the compensation block to correct the 1 Hz clock. This value gives the number of oscillator clock cycles to be added or removed. The value that is stored is in the 2’s complement format. The range of value that can be programmed is –128 to 127. When the compensation cycle is complete the ‘Done’ bit in the status register is set. IRTC will continue to compensate with this value until unless disabled. If in between a new value is programmed then that cycle will start only when compensation cycle is over.

17.5.19 IRTC Tamper Time Stamp Month & Year Register (IRTC_TTSR_MY)

IRTC_TTSR_MY — Tamper Time Stamp Month & Year Register — Offset: 0x2A

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field (bin)	TIME STAMP YEAR								0x2A
Field (bcd)	Value remains in binary as year indicates offset from base year								
Bit Type	ro	ro	ro	ro	ro	ro	ro	ro	
Reset Value	0	0	0	0	0	0	0	0	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field (bin)					TIME STAMP MONTHS				0x2B
Field (bcd)	TIME STAMP MONTHS (tens)				TIME STAMP MONTHS (units)				
Bit Type	rsvd	rsvd	rsvd	rsvd	ro	ro	ro	ro	
Reset Value	—	—	—	—	0	0	0	1	

Figure 17-20. IRTC Tamper Time Stamp Month & Year Register (IRTC_TTSR_MY)

Table 17-22. IRTC_TTSR_MY Field Descriptions

Field	Description
15–8	Tamper Detect Year Value. Count: 0 – 255
7–6	Reserved bits. Read returns zeros. Not Writeable
5–0	Tamper Detect Month Value. Valid Values: 1 – 12 ‘1’ – January ‘2’ – February ‘3’ – March ‘4’ – April ‘5’ – May ‘6’ – June ‘7’ – July ‘8’ – August ‘9’ – September ‘10’ – October ‘11’ – November ‘12’ – December ‘0’ & ‘13’ to ‘15’ – Reserved. Should not be used.

This register is a used to store the value of months and year counters when a tamper is detected. This register serves as a purpose for CPU to know when a tamper had occurred. This register is has no effect on software reset. The value shown in this register is for recent tamper detection only. Previous intrusions cannot be detected. Writing to this register has no effect and is a read only register. In BCD mode, the values of the months and year counters are shown in that format.

17.5.20 IRTC Tamper Time Stamp Day Register (IRTC_TTSR_DAY)

IRTC_TTSR_DAY — Tamper Time Stamp Day Register — Offset: 0x2C

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field (bin)									0x2C
Field (bcd)									
Bit Type	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	
Reset Value	—	—	—	—	—	—	—	—	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field (bin)				TIME STAMP DAY					0x2D
Field (bcd)	TIME STAMP DAY (tens)				TIME STAMP DAY (units)				
Bit Type	rsvd	rsvd	ro	ro	ro	ro	ro	ro	
Reset Value	—	—	0	0	0	0	0	1	

Figure 17-21. IRTC Tamper Time Stamp Day Register (IRTC_TTSR_DAY)

Table 17-23. IRTC_TTSR_DAY Field Descriptions

Field	Description
15–8	Reserved bits. Read returns zeros. Not Writeable
7–5	Reserved bits in Binary mode. Read returns zeros. In BCD mode, these are used.
4–0	Tamper Detect Days Value. Count: 1 – 31

This register is a used to store the value of days counter when a tamper is detected. This register serves as a purpose for CPU to know when a tamper had occurred. This register is has no effect on software reset. The value shown in this register is for recent tamper detection only. Previous intrusions cannot be detected. Writing to this register has no effect and is a read only register. In BCD mode, the value of days counter is shown in that format.

17.5.21 IRTC Tamper Time Stamp Hours & Minutes Register

IRTC_TTSR_HM — Tamper Time Stamp Hours & Minutes Register — Offset: 0x2E

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field (bin)				TIME STAMP HOURS					0x2E
Field (bcd)	TIME STAMP HOURS (tens)				TIME STAMP HOURS (units)				
Bit Type	rsvd	rsvd	rsvd	ro	ro	ro	ro	ro	
Reset Value	—	—	—	0	0	0	0	0	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field (bin)			TIME STAMP MINUTES						0x2F
Field (bcd)	TIME STAMP MINUTES (tens)				TIME STAMP MINUTES (units)				
Bit Type	rsvd	rsvd	ro	ro	ro	ro	ro	ro	
Reset Value	—	—	0	0	0	0	0	0	

Figure 17-22. IRTC Tamper Time Stamp Hours & Minutes Register (IRTC_TTSR_HM)

Table 17-24. IRTC_TTSR_HM Field Descriptions

Field	Description
15–13	Reserved bits in Binary mode. Read returns zeros. In BCD mode, these are used.
12–8	Tamper Detect Hours Value. Count: 0 – 23
7–6	Reserved bits in Binary mode. Read returns zeros. In BCD mode, these are used.
5–0	Tamper Detect Minutes Value. Count: 0 – 59

This register is a used to store the value of hours and minutes counters when a tamper is detected. This register serves as a purpose for CPU to know when a tamper had occurred. This register is has no effect on software reset. The value shown in this register is for recent tamper detection only. Previous intrusions cannot be detected. Writing to this register has no effect and is a read only register. In BCD mode, the values of hours and minutes are shown in that format.

17.5.22 IRTC Tamper Time Stamp Seconds Register (IRTC_TTSR_SEC)

IRTC_TTSR_SEC — Tamper Time Stamp Seconds Register — Offset: 0x30

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field (bin)							STATUS		0x30
Field (bcd)							Value in Binary		
Bit Type	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	ro	ro	
Reset Value	—	—	—	—	—	—	1	1	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field (bin)			TIME STAMP SECONDS						0x31
Field (bcd)	TIME STAMP SECONDS (tens)				TIME STAMP SECONDS (units)				
Bit Type	rsvd	rsvd	ro	ro	ro	ro	ro	ro	
Reset Value	—	—	0	0	0	0	0	0	

Figure 17-23. IRTC Tamper Time Stamp Seconds Register (IRTC_TTSR_SEC)

Table 17-25. IRTC_TTSR_SEC Field Descriptions

Field	Description
15–10	Reserved bits. Not writeable. Read returns zeros.
9–8	Tamper Detect Status. Indicates the type of tamper detected. Value is valid when tamper interrupt status bit is set. Read only Bits that reset on tamper interrupt status bit being acknowledged. 00 – No tamper detected 01 – Tamper detected via external signal 10 – Battery disconnected when MCU power is ON 11 – Battery disconnected when MCU power is OFF
7–6	Reserved bits in Binary mode. Read returns zeros. In BCD mode, these are used.
5–0	Tamper Detect Seconds Value. Count: 0 – 59

This register is a used to store the value of seconds counter when a tamper is detected. This register serves as a purpose for CPU to know when a tamper had occurred. This register is has no effect on software reset. The value shown in this register is for recent tamper detection only. Previous intrusions cannot be detected. In BCD mode, the value of seconds counter is shown in that format.

The tamper status bits indicate the reason for tamper to the CPU. These bits are valid when the interrupt status bit is set in the IRTC_ISR and cleared when the interrupt status bit is cleared. The reset value is “11” as POR is generated when both mains and battery supply are down, which can be due to tampering. The CPU can take appropriate action based on this. After calibration, CPU should clear these bits and the interrupt status bit. Writing to this register has no effect and is a read only register.

17.5.23 IRTC Up-Counter Higher Register (IRTC_UP_CNTR_H)

IRTC_UP_CNTR_H — Up Counter Higher Register — Offset: 0x32

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field	UP COUNTER Upper Word Byte3								0x32
Bit Type	ro	ro	ro	ro	ro	ro	ro	ro	
Reset Value	0	0	0	0	0	0	0	0	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field	UP COUNTER Upper Word Byte2								0x33
Bit Type	ro	ro	ro	ro	ro	ro	ro	ro	
Reset Value	0	0	0	0	0	0	0	0	

Figure 17-24. IRTC Up-Counter Higher Register (IRTC_UP_CNTR_H)

Table 17-26. IRTC_UP_CNTR_H Field Descriptions

Field	Description
15–0	Up Counter Register Value. Upper 16-bits of the 32-bit counter. Value increments on rollover from IRTC_UP_CNTR_L.

This register serves the purpose of keeping a track of count of an event that the CPU wants to track. For example the energy units consumed over a period. This register has been indicated as read-only as CPU can read the contents any time and any write to this register has no effect. The value in this register increment on rollover from IRTC_UP_CNTR_L. Software reset has no effect on the state of this counter.

17.5.24 IRTC Up-Counter Lower Register (IRTC_UP_CNTR_L)

IRTC_UP_CNTR_L — Up Counter Lower Register — Offset: 0x34

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
Field	UP COUNTER Lower Word Byte1								0x34
Bit Type	ro	ro	ro	ro	ro	ro	ro	ro	
Reset Value	0	0	0	0	0	0	0	0	

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Field	UP COUNTER Lower Word Byte0								0x35
Bit Type	rw	rw	rw	rw	rw	rw	rw	rw	
Reset Value	0	0	0	0	0	0	0	0	

Figure 17-25. IRTC Up-Counter Lower Register (IRTC_UP_CNTR_L)

Table 17-27. IRTC_UP_CNTR_L Field Descriptions

Field	Description
15–8	Up Counter Register Value. Byte 1 of the Up counter. Read only bits. Not writeable.
7–0	Up Counter Register Value. Byte 0 (or LSB) of the Up counter. This byte is write-able only and carry generated from this byte overflows and increments the other three bytes.

This register serves the purpose of keeping a track of count of an event that the CPU wants to track. For example the energy units consumed over a period. This register has been indicated as read-only as CPU can read the contents any time and any write to the lower 8-bits of the register only, which causes it to increment. Software reset has no effect on this counter.

The arrangement of the count value in these two registers is {Byte3, Byte2, Byte1, Byte0}. Big Endian is followed for all registers.

17.6 Top Level Data Flow

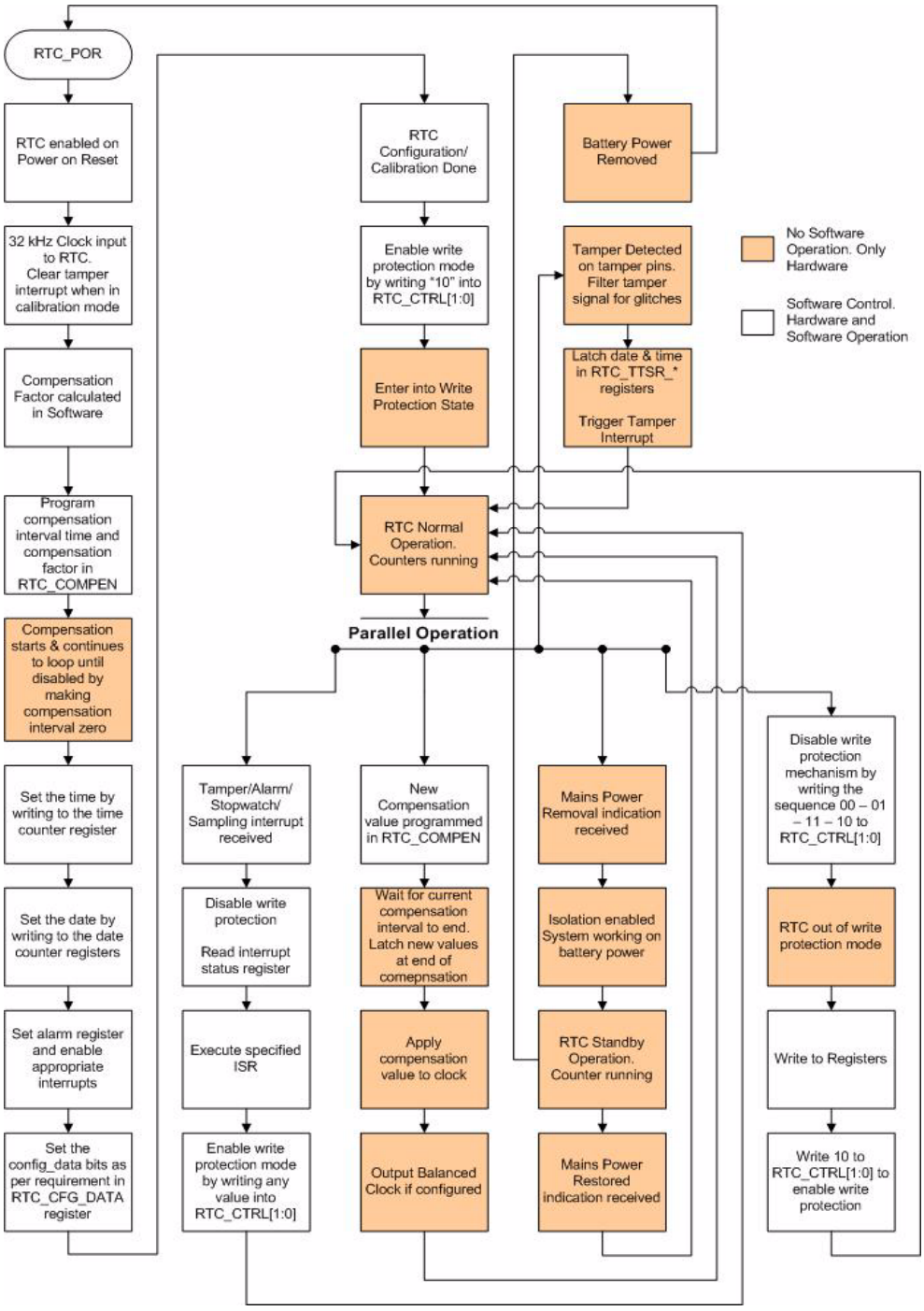


Figure 17-26. Top Level Data Flow

17.7 Modes of Operation (Based on Data Flow)

17.7.1 IRTC Configuration

- This is the initial flow that needs to be done after every power-on reset
- CPU sets the date and time into the IRTC registers
- CPU sets other control information needed for IRTC to function
- CPU enables the various interrupts and alarm time as per its requirements
- CPU stores in the Standby RAM inside IRTC, any critical data needed to be preserved in the event of main voltage loss. This can be done during normal operation too.
- IRTC has configuration bits that can be used for application dependent protocol and these bits need to retain state during mains voltage removal. These are programmed by the CPU
- CPU enables the tamper control bits. If not, this is done after 15 seconds of power on.
- Once the complete configuration of IRTC is done, the CPU should put the IRTC in write protect state. If not, this is automatically done after 15 seconds of power on.

17.7.2 IRTC Normal Operation

- During the Normal Operation of IRTC, all write accesses to registers are blocked. Only by following the unlock sequence mentioned in Register Description can CPU get the write access and modify the contents on the registers.
- During the normal operation, any interrupt might occur or the Standby RAM needs to be updated or some control information needs to change.
- Any such operation should be preceded by the Unlock Sequence mentioned above.
- Interrupts can occur when the time in counters match the value in alarm registers or when a sampling timer expires or a tamper is detected
- CPU should service these interrupts and then put the IRTC back in write-protect mode. If not, write protect is enabled 2 seconds after unlock.

17.7.3 IRTC Standby Operation

- Standby operation is defined as operation of IRTC on battery voltage and no MCU voltage
- Whenever the main voltage falls below a certain threshold, the system switches to battery power and IRTC functions normally.
- The switching of power and detection of low voltage threshold of MCU voltage is done in analog blocks outside the IRTC design. Refer to Section 17.1.1, “IRTC Power Supply Source,” for details.
- Whenever the MCU voltage is restored, the IRTC switches over to this power
- If the battery drains out and when the power is supplied again to IRTC (by new battery or CPU power), a power on reset is sent to IRTC and all registers are reset. Application needs to signal for re-calibration of the IRTC.

17.7.4 IRTC Tamper Detect

- CPU should enable the tamper detect pins when the configuration of IRTC is complete. If not, it is automatically enabled 15 seconds after power on.
- IRTC has one tamper detect pin. High level input on this pin is considered as a tamper and an interrupt is generated if the tamper interrupt is enabled.
- Whenever the battery power is removed and MCU power is ON, it is considered as a tamper and a time stamp of that event is stored. In case the battery is removed as a part of maintenance then the application should take care of this and not indicate a tamper.
- An interrupt is sent to CPU. The IRTC interrupt service routine should read the Interrupt Status Register to determine if a tamper had occurred.

17.7.5 IRTC Calibration

- CPU performs calculations (external to IRTC) to determines the correction factor in the 1 Hz clock to remove any variations that might creep into the 32 kHz Oscillator Clock due to Temperature or Crystal. Calibration procedure done in software provides the correction factor which is used by compensation hardware.
- Application programs this correction factor into the IRTC registers, hardware within the IRTC compensates the clock to output balanced compensation interval.
- IRTC is said to be calibrated but this performed is done every user-defined interval

17.8 Block Description (Brief Overview)

17.8.1 Compensation Logic

The compensation circuit provides an accurate and wide compensation range, which is suitable for many crystals, and can correct errors as high as 3906 PPM and as low as 0.119 PPM. The same hardware logic supports both temperature compensation and frequency compensation.

To perform temperature compensation the CPU maintains a look-up-table which lists the change in frequency of crystal for each degree change in temperature. CPU wakes up periodically to measure the external temperature via a temperature sensor connected to an A/D converter. CPU uses the look-up-table to determine the compensation factor and writes the value to be compensated (in terms of number of IRTC oscillator clock cycles in 2’s complement format) in the IRTC compensation register (IRTC_COMPEN). Based on the value written the hardware add/remove pulses accordingly to adjust the 1 Hz frequency due to variation on temperature.

To perform crystal compensation, the calculation of correction is done by firmware using crystal characteristics, where the CPU sets the correction factor in the two’s complement format in the same IRTC compensation register (IRTC_COMPEN). Based on the values written in the IRTC_COMPEN register, the circuit compensates by adding/skipping pulses in the oscillator clock signal.

There are two important components in the compensation algorithm viz. Compensation value and compensation interval. These are defined as:

- Compensation/Correction value: Compensation/Correction Value is a 2's complement value with which the IRTC oscillator clock is modified by either adding or removing pulses from it.
- Compensation interval: compensation Interval is the duration in which the correction value is applied. This is the time in which this block adds or removes pulses thereby ensuring that the compensation interval is close to the interval obtained with an ideal 1 Hz clock.

Vital Statistics:

- Range of compensation Interval: 1 second to 255 seconds (0 disables compensation)
- Range of compensation: –128 clocks to 127clocks (IRTC oscillator clocks)
- Selection criteria: Compensation is done only when enabled by CPU. CPU can disable compensation by programming a 0 compensation interval

Compensation Flow:

The operation starts in an IDLE state waiting for the firmware to enable compensation. Since the same hardware logic is used for temperature and crystal compensation, the firmware should be able to provide a value that takes into account correction for both temperature and crystal. When enabled, the compensation cycles are added or removed till the compensation interval expires. On completion of the compensation interval, the done bit is asserted to the CPU and if compensation is still enabled, the next compensation cycle starts. Compensation state machine returns to IDLE state when compensation is disabled by writing 0 to compensation interval. A newly programmed value is picked only when the current compensation cycle has completed.

Figure 17-27 shows the flow chart for the logical compensation flow of the state machine which has been explained in subsequent pages.

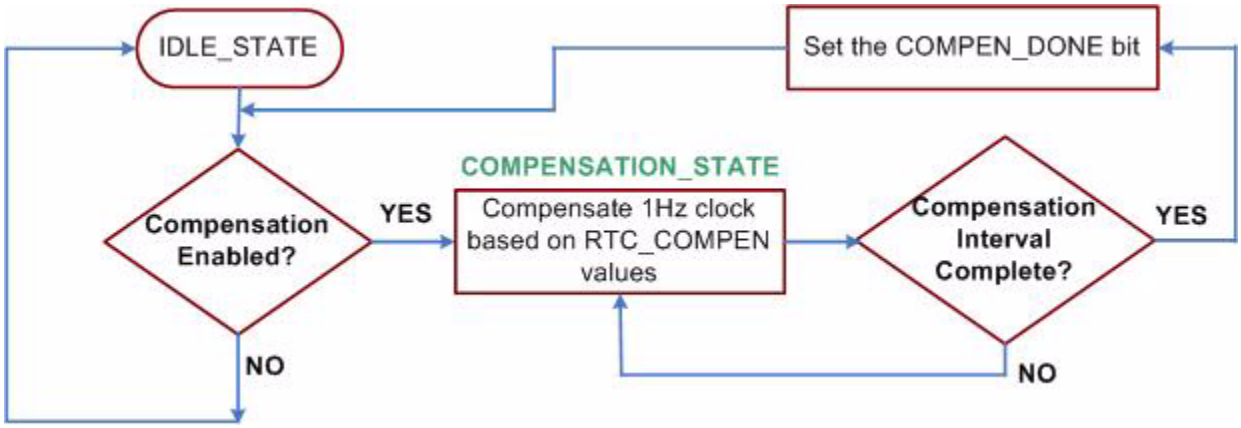


Figure 17-27. Compensation Control Flow

Compensation Logic Hardware:

The compensation logic hardware comprises of a simple counter (irtc_osc_32k_div_counter) which divides the 32 kHz clock down to 1 Hz by counting up till 32767. In order to add or remove pulses the start point of the counter is shifted and the counter still counts up to 32767 to generate a balanced 1 Hz clock. The state machine for this block controls the loading of correction value into the counter and ensures that each compensation window is always aligned to the seconds boundary. Switching to newly programmed compensation values is done when the compensation interval of current run is complete and CPU has not

disabled the compensation logic. If no value is programmed the state machine continues to perform compensation with the previously programmed values till compensation is disabled by writing zero as the compensation interval. Figure below shows the block diagram for the compensation block.

Recommendation for Optimal Compensation:

Since the addition and removal of pulses is done in the first second of the compensation interval, the CPU has the option of finding the compensation factor over a period of time and then calculating the correction factor per second and enable compensation hardware every second for better accuracy. The 1 Hz clock generated will have uniform period.

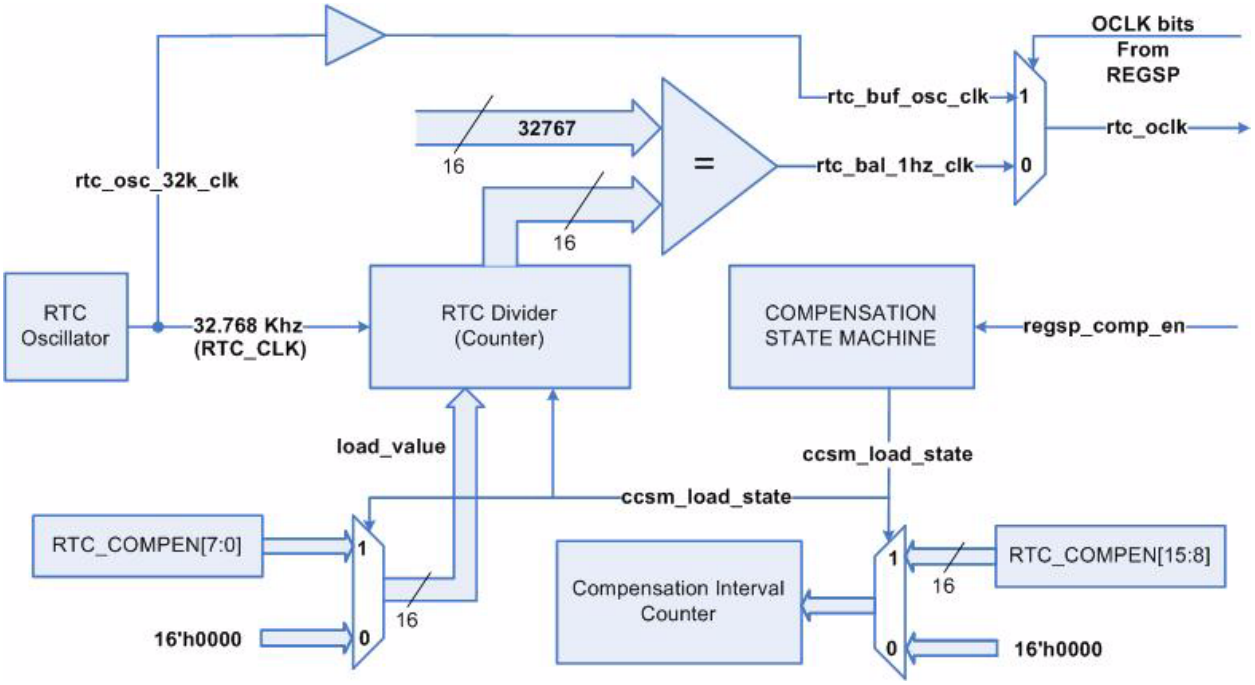


Figure 17-28. Compensation Logic Block Diagram

17.8.2 Write Protection Logic

This logic protects the IRTC registers and Standby RAM from any spurious updates that can happen due run-away code. The logic is based on a state machine that monitors the values written to WE[1:0] bits of the IRTC_CTRL register. By default unconditional write access is allowed to these bits only.

To enable write protection, “10” is written on these bits. To disable write protection, the sequence “00 – 01 – 11 – 10” is written onto these bits.

After a power on reset, the write-protect mechanism is disabled, allowing the user code to calibrate the IRTC clock, set the time in the clock registers, and set the date in the calendar registers. Once calibration and time & date settings are done, the user code should enable write protection mode. If not, the registers are put into write protect mode 15 seconds after power on. In case the write protect mode is unlocked to update registers, then the write protect mode is enabled 2 seconds after unlock if not done by CPU.

Any access made to the register space when write protection is enabled (i.e. registers in locked mode) will cause the transfer error signal to be asserted.

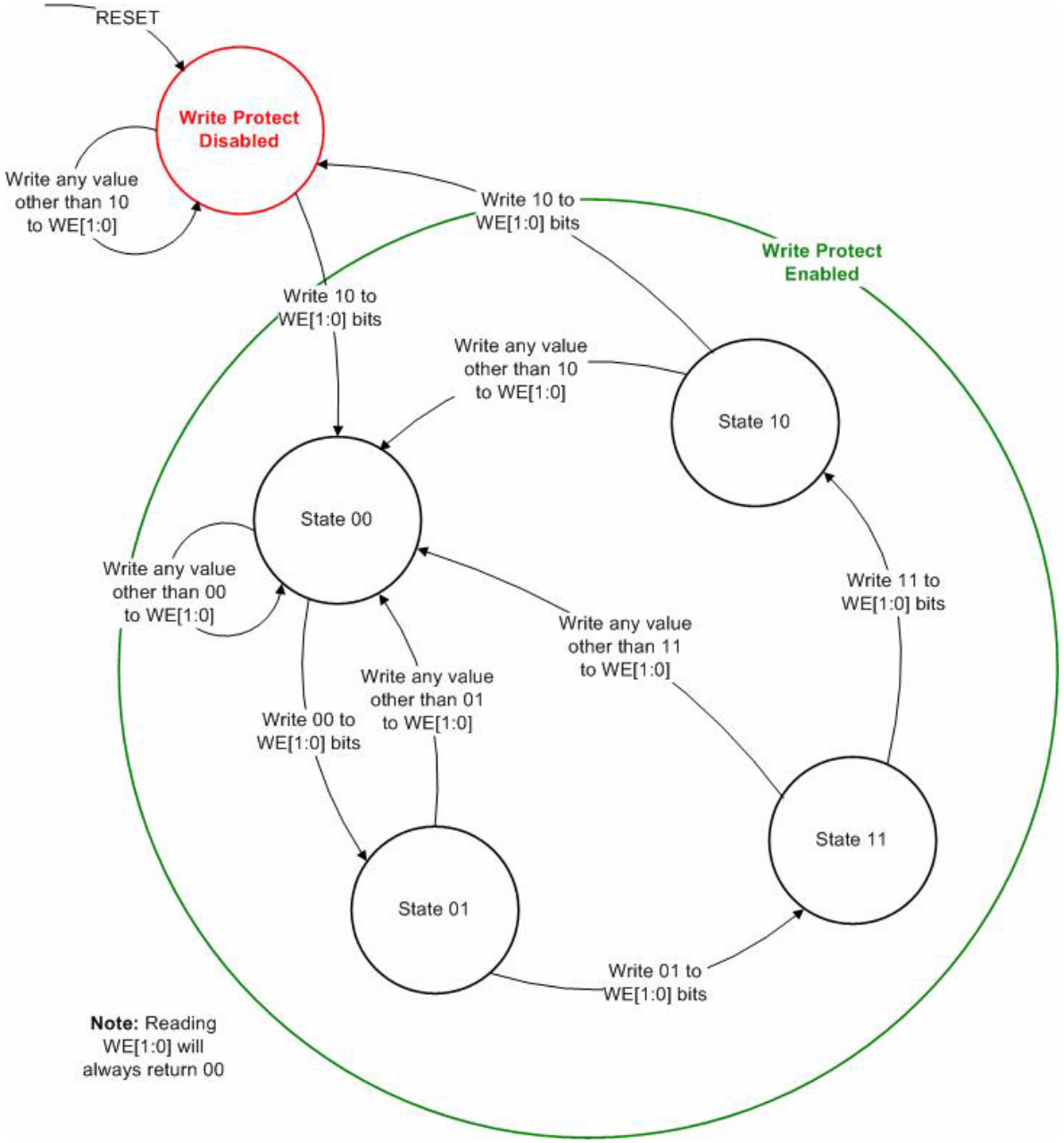


Figure 17-29. Write Protect State Machine

17.8.3 Tamper Detection Logic

The tamper detect mechanism has been provided to detect any intrusion made to system. The tamper logic monitors for 3 conditions of tampering as explained below.

Tamper Condition 1: Battery removed when MCU is powered OFF: The detection of battery removal during system power off is done using a flop that is asserted on power-on reset. Since there is no difference between a proper shutdown and this tamper condition, this is considered as a tamper always. Now it is up to the firmware to differentiate between the tamper condition and normal power up. One way is to ignore

this tamper interrupt when the MCU or application is in Service mode and simply reset the tamper interrupt. For other conditions it will be taken as a tamper.

Tamper Condition 2: Battery removed when MCU is powered ON: The analog circuitry monitoring the battery voltage indicates when the battery voltage is removed. This signal is used by the tamper circuit to indicate a tamper provided MCU power in ON. A flop will be present that will be cleared by CPU during calibration. Any change of state on this signal will be detected as a tamper.

Tamper Condition 3: Off Chip Tamper Indication:

An external pin on the MCU is also used to monitor tampering external to the MCU. For example, this pin can be used to indicate that the energy meter unit is opened. Again a flop will be present inside IRTC that will be cleared by CPU during calibration. High level of this pin will be detected as a tamper.

A tamper filtering circuit is connected to both tamper pins to prevent any glitches triggering the tamper interrupt. The duration for which a tamper signal should be asserted to be indicated as tamper is programmable by user in the IRTC control register. The filtered signals are then used to generate the tamper status and interrupt signals.

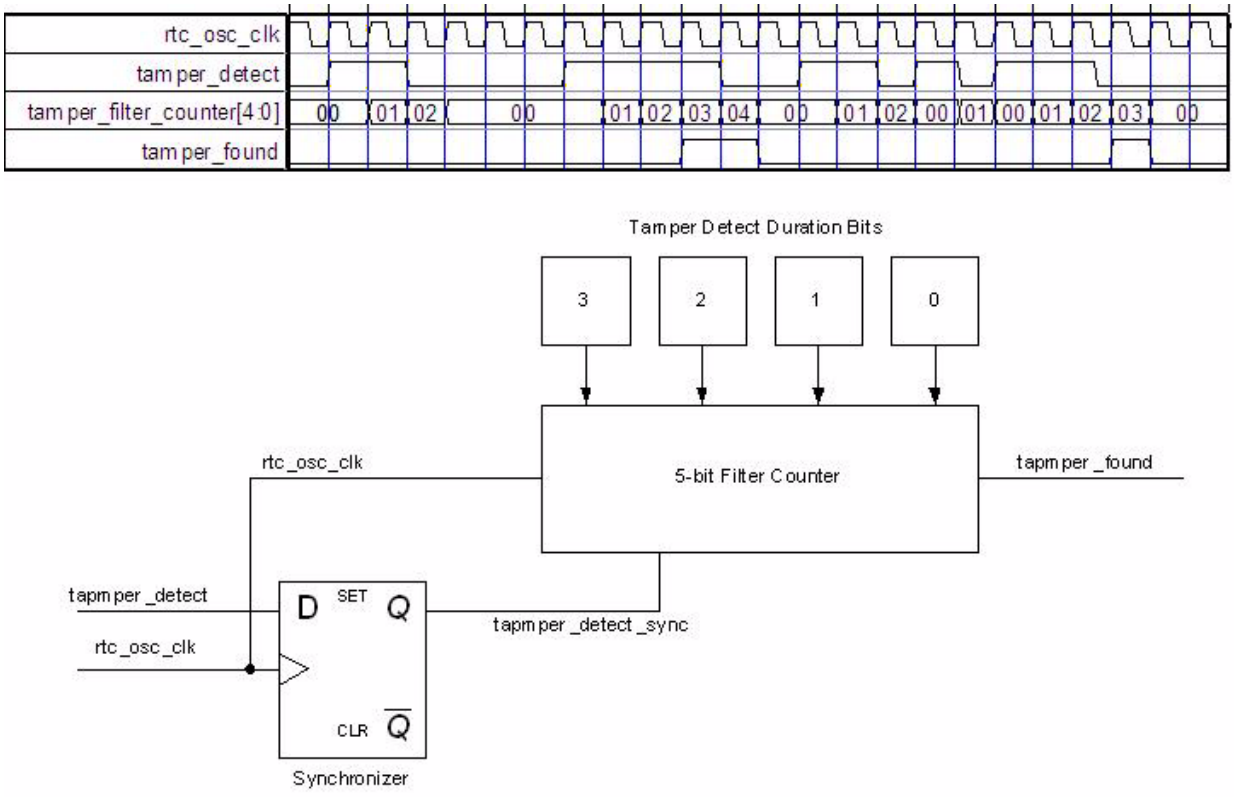


Figure 17-30. Tamper Capture Circuit and Timing Diagram

Tamper Detection Flow:

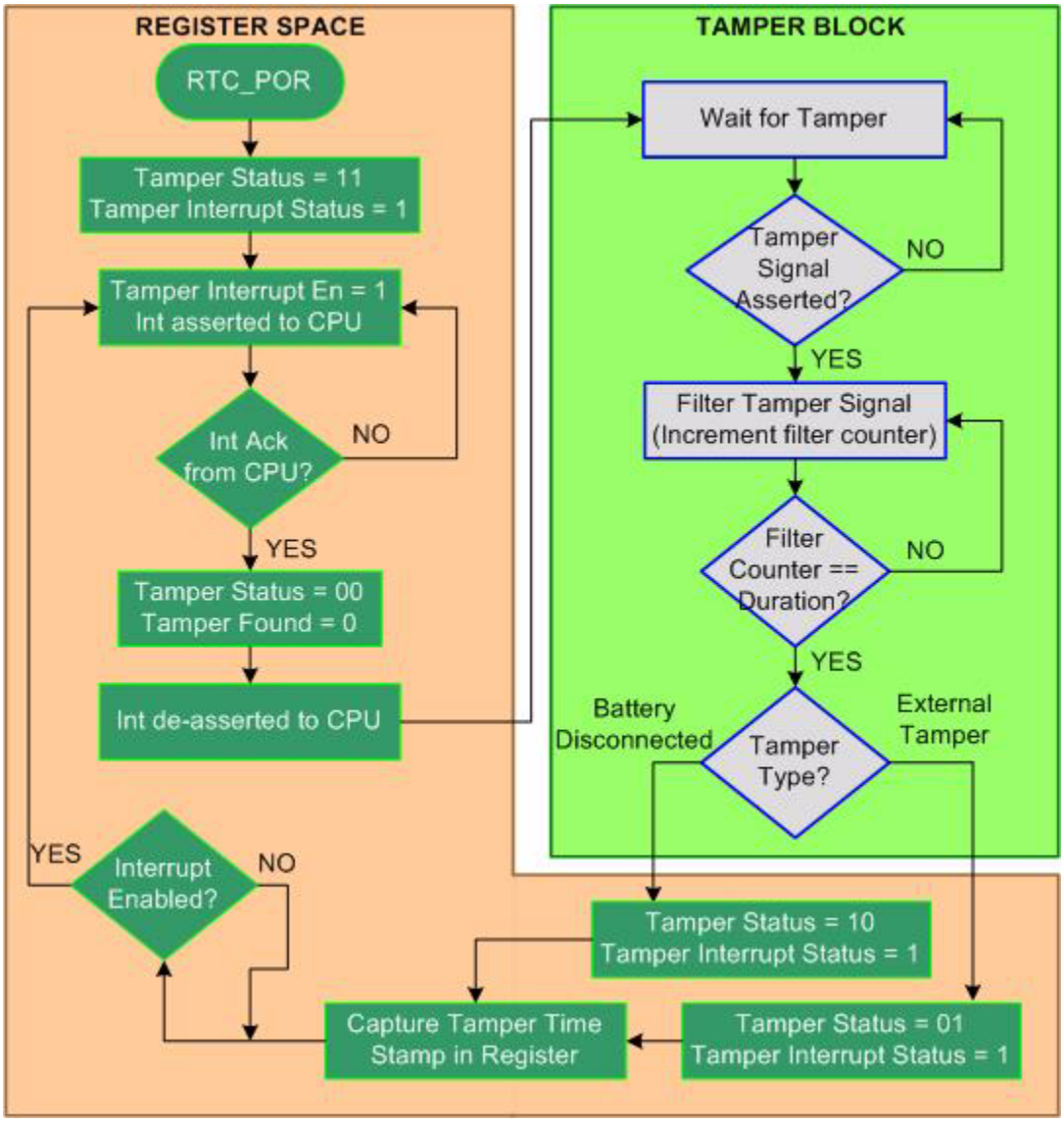


Figure 17-31. Tamper Detection Flow Diagram

Description:

- POR asserts tamper status bits in register space (in IRTC_TTSR_SEC) and tamper interrupt status bit (in IRTC_ISR)
- Tamper Interrupt Enable bit is also asserted on POR and an interrupt to CPU is indicated
- When CPU acknowledges the tamper interrupt by writing 1’b1 to the tamper interrupt status bit, the interrupt is de-asserted and tamper status bits are cleared.
- Any tamper signal asserted externally (battery disconnected or tamper detected) are filtered in the tamper block.
- Tamper signal is asserted when the filter counter matches the programmed filter duration.
- Based on the type of tamper detected the appropriate status (10 for battery disconnected or 01 for external tamper) is captured in the tamper status register (IRTC_TTSR_SEC).

- Register Space also captures the time stamp (both in IRTC_TTSR_*) and asserts the tamper interrupt status bit (in IRTC_ISR)
- If interrupt is enabled CPU is interrupted by asserting the interrupt
- Tamper status bits (in Register Space) are cleared when 1 is written to tamper interrupt status bit (in IRTC_ISR). This clears all the flops in the tamper block.
- Tamper interrupt can be armed or disabled by writing to the interrupt enable register bit (in IRTC_IER) through the IPS Bus Interface

Chapter 18
8-Bit Modulo Timer (MTIM)

18.1 Introduction

The MTIM is a simple 8-bit timer with several software selectable clock sources and a programmable interrupt.

For MCUs that have more than one MTIM, the MTIMs are collectively called MTIMx. For example, MTIMx for an MCU with two MTIMs would refer to MTIM1 and MTIM2. For MCUs that have exactly one MTIM, it is always referred to as MTIM1.

18.1.1 Features

Timer system features include:

- 8-bit up-counter
  - — Free-running or 8-bit modulo limit
  - — Software controllable interrupt on overflow
  - — Counter reset bit (TRST)
  - — Counter stop bit (TSTP)
- Four software selectable clock sources for input to prescaler:
  - — System bus clock — rising edge
  - — Fixed frequency clock (XCLK) — rising edge
  - — External clock source on the TCLK pin — rising edge
  - — External clock source on the TCLK pin — falling edge
- Nine selectable clock prescale values:
  - — Clock source divide by 1, 2, 4, 8, 16, 32, 64, 128, or 256

18.1.2 Modes of Operation

This section defines the MTIM’s operation in stop, wait and background debug modes.

18.1.2.1 MTIM in Wait Mode

The MTIM continues to run in wait mode if enabled before executing the WAIT instruction. Therefore, the MTIM can be used to bring the MCU out of wait mode if the timer overflow interrupt is enabled. For lowest possible current consumption, the MTIM should be stopped by software if not needed as an interrupt source during wait mode.

18.1.2.2 MTIM in Stop Modes

The MTIM is disabled in all stop modes, regardless of the settings before executing the STOP instruction. Therefore, the MTIM cannot be used as a wake up source from stop modes.

Waking from stop1 and stop2 modes, the MTIM will be put into its reset state. If stop3 is exited with a reset, the MTIM will be put into its reset state. If stop3 is exited with an interrupt, the MTIM continues from the state it was in when stop3 was entered. If the counter was active upon entering stop3, the count will resume from the current value.

18.1.2.3 MTIM in Active Background Mode

The MTIM suspends all counting until the microcontroller returns to normal user operating mode. Counting resumes from the suspended value as long as an MTIM reset did not occur (TRST written to a 1 or MTIMxMOD written).

18.1.3 Block Diagram

The block diagram for the modulo timer module is shown Figure 18-1.

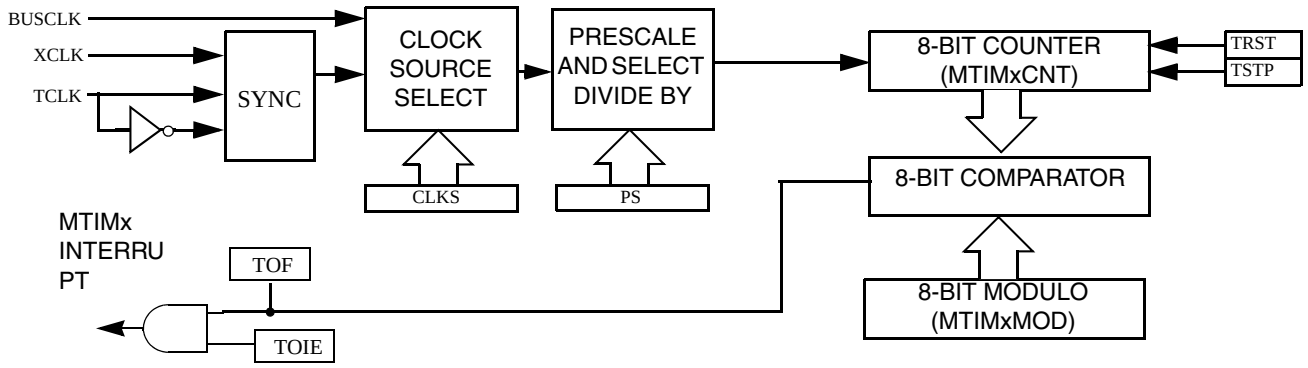


Figure 18-1. Modulo Timer (MTIM) Block Diagram

18.2 External Signal Description

The MTIM includes one external signal, TCLK, used to input an external clock when selected as the MTIM clock source. The signal properties of TCLK are shown in Table 18-1.

Table 18-1.

Signal	Function	I/O
TCLK	External clock source input into MTIM	I

The TCLK input must be synchronized by the bus clock. Also, variations in duty cycle and clock jitter must be accommodated. Therefore, the TCLK signal must be limited to one-fourth of the bus frequency.

The TCLK pin can be muxed with a general-purpose port pin. See the Pins and Connections chapter for the pin location and priority of this function.

18.3 Register Definition

Figure 18-2 is a summary of <<BLOCK NAME>> registers.

Figure 18-2. MTIMx Register Summary

Name		7	6	5	4	3	2	1	0
MTIMxSC	R	TOF	TOIE	0	TSTP	0	0	0	0
	W			TRST					
MTIMxCLK	R	0	0	CLKS		PS			
	W								
MTIMxCNT	R	COUNT							
	W								
MTIMxMOD	R	MOD							
	W								

Each MTIM includes four registers:

- An 8-bit status and control register
- An 8-bit clock configuration register
- An 8-bit counter register
- An 8-bit modulo register

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all MTIM registers.This section refers to registers and control bits only by their names and relative address offsets.

Some MCUs may have more than one MTIM, so register names include placeholder characters to identify which MTIM is being referenced.

18.3.1 MTIMx Status and Control Register (MTIMxSC)

MTIMxSC contains the overflow status flag and control bits which are used to configure the interrupt enable, reset the counter, and stop the counter.

	7	6	5	4	3	2	1	0
R	TOF	TOIE	0	TSTP	0	0	0	0
W			TRST					
Reset:	0	0	0	1	0	0	0	0

Figure 18-3. MTIMx Status and Control Register

Table 18-2. MTIMxSC Field Descriptions

Field	Description
7 TOF	**MTIM Overflow Flag** — This read-only bit is set when the MTIM counter register overflows to $00 after reaching the value in the MTIM modulo register. Clear TOF by reading the MTIMxSC register while TOF is set, then writing a 0 to TOF. TOF is also cleared when TRST is written to a 1 or when any value is written to the MTIMxMOD register. 0 MTIM counter has not reached the overflow value in the MTIM modulo register. 1 MTIM counter has reached the overflow value in the MTIM modulo register.
6 TOIE	**MTIM Overflow Interrupt Enable** — This read/write bit enables MTIM overflow interrupts. If TOIE is set, then an interrupt is generated when TOF = 1. Reset clears TOIE. Do not set TOIE if TOF = 1. Clear TOF first, then set TOIE. 0 TOF interrupts are disabled. Use software polling. 1 TOF interrupts are enabled.
5 TRST	**MTIM Counter Reset** — When a 1 is written to this write-only bit, the MTIM counter register resets to $00 and TOF is cleared. Reading this bit always returns 0. 0 No effect. MTIM counter remains at current state. 1 MTIM counter is reset to $00.
4 TSTP	**MTIM Counter Stop** — When set, this read/write bit stops the MTIM counter at its current value. Counting resumes from the current value when TSTP is cleared. Reset sets TSTP to prevent the MTIM from counting. 0 MTIM counter is active. 1 MTIM counter is stopped.
3:0	Unused register bits, always read 0.

18.3.2 MTIMx Clock Configuration Register (MTIMxCLK)

MTIMxCLK contains the clock select bits (CLKS) and the prescaler select bits (PS).

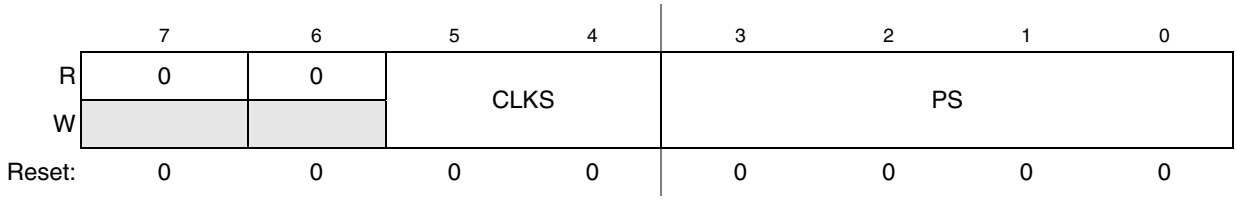


Figure 18-4. MTIMx Clock Configuration Register

Table 18-3. MTIMxCLK Field Descriptions

Field	Description
7:6	Unused register bits, always read 0.
5:4 CLKS	**Clock Source Select** — These two read/write bits select one of four different clock sources as the input to the MTIM prescaler. Changing the clock source while the counter is active does not clear the counter. The count continues with the new clock source. Reset clears CLKS to 000. 00 Encoding 0. Bus clock (BUSCLK) 01 Encoding 1. Fixed-frequency clock (XCLK) 10 Encoding 3. External source (TCLK pin), falling edge 11 Encoding 4. External source (TCLK pin), rising edge All other encodings default to the bus clock (BUSCLK).
3:0 PS	**Clock Source Prescaler** — These four read/write bits select one of nine outputs from the 8-bit prescaler. Changing the prescaler value while the counter is active does not clear the counter. The count continues with the new prescaler value. Reset clears PS to 0000. 0000 Encoding 0. MTIM clock source ÷ 1 0001 Encoding 1. MTIM clock source ÷ 2 0010 Encoding 2. MTIM clock source ÷ 4 0011 Encoding 3. MTIM clock source ÷ 8 0100 Encoding 4. MTIM clock source ÷ 16 0101 Encoding 5. MTIM clock source ÷ 32 0110 Encoding 6. MTIM clock source ÷ 64 0111 Encoding 7. MTIM clock source ÷ 128 1000 Encoding 8. MTIM clock source ÷ 256 All other encodings default to MTIM clock source ÷ 256.

18.3.3 MTIMx Counter Register (MTIMxCNT)

MTIMxCNT is the read-only value of the current MTIM count of the 8-bit counter.

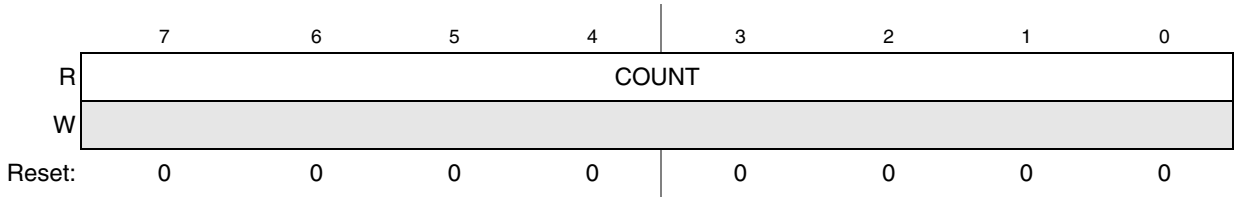


Figure 18-5. MTIMx Counter Register

Table 18-4. MTIMxCNT Field Descriptions

Field	Description
7:0 COUNT	**MTIM Count** — These eight read-only bits contain the current value of the 8-bit counter. Writes have no effect to this register. Reset clears the count to $00.

18.3.4 MTIMx Modulo Register (MTIMxMOD)

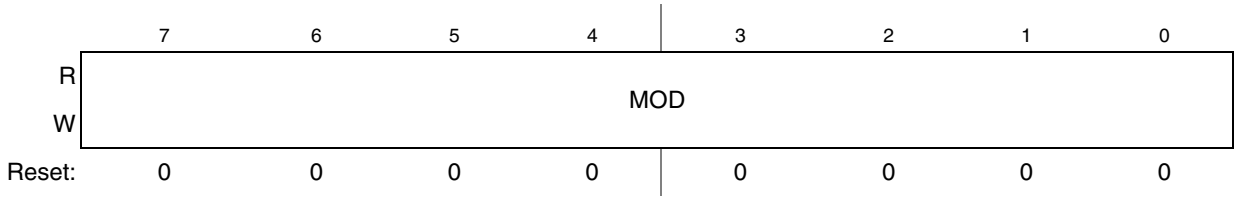


Figure 18-6. MTIMx Modulo Register

Table 18-5. MTIMxMOD Field Descriptions

Field	Description
7:0 MOD	**MTIM Modulo** — These eight read/write bits contain the modulo value used to reset the count and set TOF. A value of $00 puts the MTIM in free-running mode. Writing to MTIMxMOD resets the COUNT to $00 and clears TOF. Reset sets the modulo to $00.

18.4 Functional Description

The MTIM is composed of a main 8-bit up-counter with an 8-bit modulo register, a clock source selector, and a prescaler block with nine selectable values. The module also contains software selectable interrupt logic.

The MTIM counter (MTIMxCNT) has three modes of operation: stopped, free-running, and modulo. Out of reset, the counter is stopped. If the counter is started without writing a new value to the modulo register, then the counter will be in free-running mode. The counter is in modulo mode when a value other than $00 is in the modulo register while the counter is running.

After any MCU reset, the counter is stopped and reset to $00, and the modulus is set to $00. The bus clock is selected as the default clock source and the prescale value is divide by 1. To start the MTIM in free-running mode, simply write to the MTIM status and control register (MTIMxSC) and clear the MTIM stop bit (TSTP).

Four clock sources are software selectable: the internal bus clock, the fixed frequency clock (XCLK), and an external clock on the TCLK pin, selectable as incrementing on either rising or falling edges. The MTIM clock select bits (CLKS1:CLKS0) in MTIMxSC are used to select the desired clock source. If the counter is active (TSTP = 0) when a new clock source is selected, the counter will continue counting from the previous value using the new clock source.

Nine prescale values are software selectable: clock source divided by 1, 2, 4, 8, 16, 32, 64, 128, or 256. The prescaler select bits (PS[3:0]) in MTIMxSC select the desired prescale value. If the counter is active (TSTP = 0) when a new prescaler value is selected, the counter will continue counting from the previous value using the new prescaler value.

The MTIM modulo register (MTIMxMOD) allows the overflow compare value to be set to any value from $01 to $FF. Reset clears the modulo value to $00, which results in a free running counter.

When the counter is active (TSTP = 0), the counter increments at the selected rate until the count matches the modulo value. When these values match, the counter overflows to $00 and continues counting. The MTIM overflow flag (TOF) is set whenever the counter overflows. The flag sets on the transition from the modulo value to $00. Writing to MTIMxMOD while the counter is active resets the counter to $00 and clears TOF.

Clearing TOF is a two-step process. The first step is to read the MTIMxSC register while TOF is set. The second step is to write a 0 to TOF. If another overflow occurs between the first and second steps, the clearing process is reset and TOF will remain set after the second step is performed. This will prevent the second occurrence from being missed. TOF is also cleared when a 1 is written to TRST or when any value is written to the MTIMxMOD register.

The MTIM allows for an optional interrupt to be generated whenever TOF is set. To enable the MTIM overflow interrupt, set the MTIM overflow interrupt enable bit (TOIE) in MTIMxSC. TOIE should never be written to a 1 while TOF = 1. Instead, TOF should be cleared first, then the TOIE can be set to 1.

18.4.1 MTIM Operation Example

This section shows an example of the MTIM operation as the counter reaches a matching value from the modulo register.

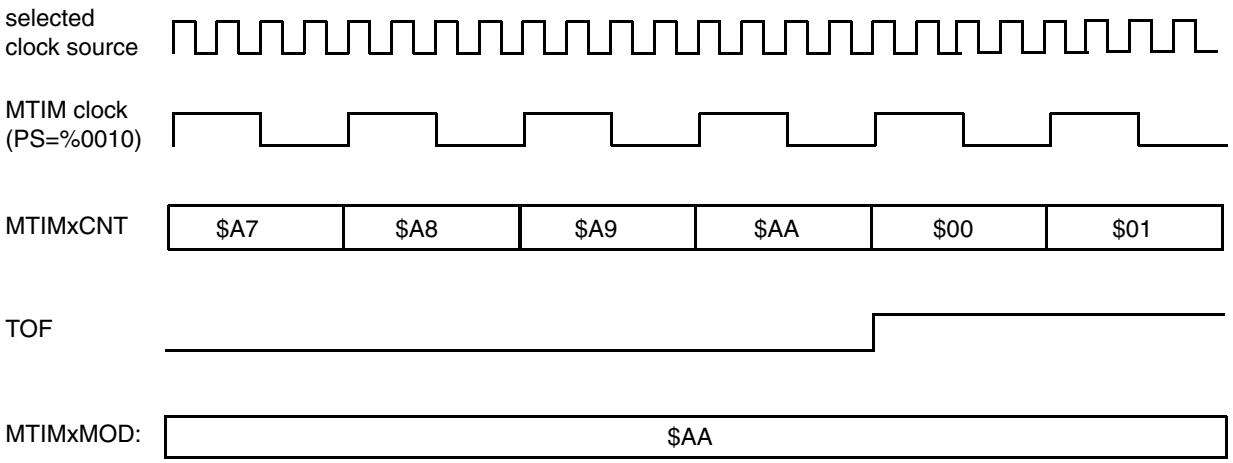


Figure 18-7. MTIM counter overflow example

In the example of Figure 18-7, the selected clock source could be any of the five possible choices. The prescaler is set to PS = %0010 or divide-by-4. The modulo value in the MTIMxMOD register is set to $AA. When the counter, MTIMxCNT, reaches the modulo value of $AA, the counter overflows to $00 and continues counting. The timer overflow flag, TOF, sets when the counter value changes from $AA to $00. An MTIM overflow interrupt is generated when TOF is set, if TOIE = 1.

Chapter 19
16-Bit Modulo Timer (MTIM16)

19.1 Introduction

The MTIM3 is a simple 16-bit timer with several software selectable clock sources and a programmable interrupt.

19.2 Features

Timer system features include:

- 16-bit up-counter
  - — Free-running or 16-bit modulo limit
  - — Software controllable interrupt on overflow
  - — Counter reset bit (TRST)
  - — Counter stop bit (TSTP)
- Four software selectable clock sources for input to prescaler:
  - — System bus clock — rising edge
  - — Fixed frequency clock (XCLK) — rising edge
  - — External clock source on the TCLK pin — rising edge
  - — External clock source on the TCLK pin — falling edge
- Nine selectable clock prescale values:
  - — Clock source divide by 1, 2, 4, 8, 16, 32, 64, 128, or 256

19.2.1 Block Diagram

The block diagram for the modulo timer module is shown Figure 19-1.

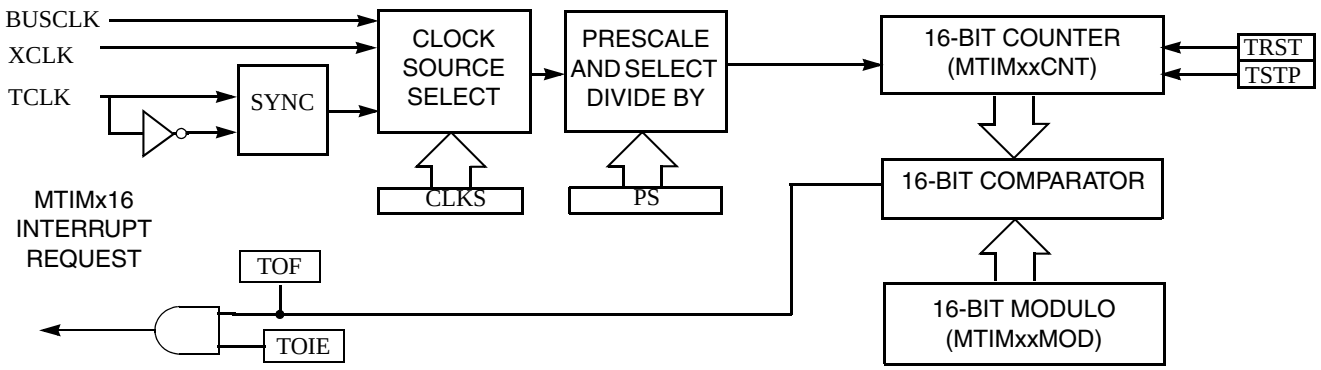


Figure 19-1. Modulo Timer (S08MTIM16) Block Diagram

19.2.2 Modes of Operation

This section defines MTIM16 operation in stop, wait, and background debug modes.

19.2.2.1 MTIM16 in Wait Mode

The MTIM16 continues to run in wait mode if enabled prior to the execution of the WAIT instruction. The timer overflow interrupt brings the MCU out of wait mode if it is enabled. For lowest possible current

consumption, the MTIM16 should be stopped by software if it is not needed as an interrupt source during wait mode.

19.2.2.2 MTIM16 in Stop Modes

The MTIM16 is disabled in all stop modes, regardless of the settings before executing the STOP instruction. Therefore, the MTIM16 cannot be used as a wake up source from stop mode.

Upon waking from stop2 mode, the MTIM16 will enter its reset state. If stop3 is exited with a reset, the MTIM16 will enter its reset state. If stop3 is exited with an interrupt, the MTIM16 continues from the state it was in stop3 . If the counter was active upon entering stop3, the count will resume from the current value.

19.2.2.3 MTIM16 in Active Background Mode

The MTIM16 stops all counting until the microcontroller returns to normal user operating mode. Counting resumes from the suspended value as long as an MTIM16 reset did not occur (TRST written to a 1).

19.3 External Signal Description

19.3.1 TCLK — External Clock Source Input into MTIM16

The MTIM16 includes one external signal, TCLK, used to input an external clock when selected as the MTIM16 clock source.The signal properties of TCLK are shown in Table 19-1.

Table 19-1. TCLK Properties

Signal	Function	I/O
TCLK	External clock source input into MTIM16	I

The TCLK input must be synchronized by the bus clock. Also, variations in duty cycle and clock jitter must be accommodated. As a result, the TCLK signal must be limited to one-fourth of the bus frequency.

The TCLK pin can be muxed with a general-purpose port pin. See Chapter 2, “Pins and Connections” for the pin location and priority of this function.

19.4 Register Definition

Each MTIM16 includes four registers:

- An 8-bit status and control register
- An 8-bit clock configuration register
- A 16-bit counter register

A 16-bit modulo registerRefer to the direct-page register summary in the Memory chapter for the absolute address assignments for all MTIM16 registers.This section refers to registers and control bits only by their names and relative address offsets.

Some MCUs may have more than one MTIM16, so register names include placeholder characters to identify the correct MTIM16.

19.4.1 MTIMx16 Status and Control Register (MTIMxSC)

MTIMxSC contains the overflow status flag and control bits. These are used to configure the interrupt enable, reset the counter, and stop the counter.

	7	6	5	4	3	2	1	0
R	TOF	TOIE	0	TSTP	0	0	0	0
W			TRST					
Reset:	0	0	0	1	0	0	0	0

Figure 19-2. MTIM16 Status and Control Register (MTIMxSC)

Table 19-2. MTIMxSC Field Descriptions

Field	Description
7 TOF	**MTIM16 Overflow Flag** — This bit is set when the MTIM16 counter register overflows to 0x0000 after reaching the value in the MTIM16 modulo register. Clear TOF by reading the MTIMxSC register while TOF is set, then writing a 0 to TOF. Writing a 1 has not effect. TOF is also cleared when TRST is written to a 1. 0 MTIM16 counter has not reached the overflow value in the MTIM16 modulo register. 1 MTIM16 counter has reached the overflow value in the MTIM16 modulo register.
6 TOIE	**MTIM16 Overflow Interrupt Enable** — This read/write bit enables MTIM16 overflow interrupts. If TOIE is set, then an interrupt is generated when TOF = 1. Reset clears TOIE. Do not set TOIE if TOF = 1. Clear TOF first, then set TOIE. 0 TOF interrupts are disabled. Use software polling. 1 TOF interrupts are enabled.
5 TRST	**MTIM16 Counter Reset** — When an 1 is written to this write-only bit, the MTIM16 counter register resets to 0x0000 and TOF is cleared. Writing an 1 to this bit also makes the modulo value to take effect at once. Reading this bit always returns 0. 0 No effect. MTIM16 counter remains in its current state. 1 MTIM16 counter is reset to 0x0000.
4 TSTP	**MTIM16 Counter Stop** — When set, this read/write bit stops the MTIM16 counter at its current value.Counting resumes from the current value when TSTP is cleared.Reset sets TSTP to prevent the MTIM16 from counting. 0 MTIM16 counter is active. 1 MTIM16 counter is stopped.
3:0	Unused register bits, always read 0.

19.4.2 MTIM16 Clock Configuration Register (MTIMxCLK)

MTIMxCLK contains the clock select bits (CLKS) and the prescaler select bits (PS).

	7	6	5	4	3	2	1	0
R	0	0	CLKS		PS			
W								
Reset:	0	0	0	0	0	0	0	0

Figure 19-3. MTIM16 Clock Configuration Register (MTIMxCLK)

Table 19-3. MTIMxCLK Field Descriptions

Field	Description
7:6	Unused register bits, always read 0.
5:4 CLKS	**Clock Source Select** — These two read/write bits select one of four different clock sources as the input to the MTIM16 prescaler. Changing the clock source while the counter is active does not clear the counter. The count continues with the new clock source. Reset clears CLKS to 00. 00 Encoding 0. Bus clock (BUSCLK) 01 Encoding 1. Fixed-frequency clock (XCLK) 10 Encoding 3. External source (TCLK pin), falling edge 11 Encoding 4. External source (TCLK pin), rising edge
3:0 PS	**Clock Source Prescaler** — These four read/write bits select one of nine outputs from the 8-bit prescaler. Changing the prescaler value while the counter is active does not clear the counter. The count continues with the new prescaler value. Reset clears PS to 0000. 0000 Encoding 0. MTIM16 clock source ÷ 1 0001 Encoding 1. MTIM 16clock source ÷ 2 0010 Encoding 2. MTIM16 clock source ÷ 4 0011 Encoding 3. MTIM16 clock source ÷ 8 0100 Encoding 4. MTIM16 clock source ÷ 16 0101 Encoding 5. MTIM16 clock source ÷ 32 0110 Encoding 6. MTIM16 clock source ÷ 64 0111 Encoding 7. MTIM16 clock source ÷ 128 1000 Encoding 8. MTIM16 clock source ÷ 256 All other encodings default to MTIM16 clock source ÷ 256.

19.4.3 MTIM16 Counter Register High/Low (MTIMxCNTH:L)

MTIMxCNTH is the read-only value of the high byte of current MTIM16 16-bit counter.

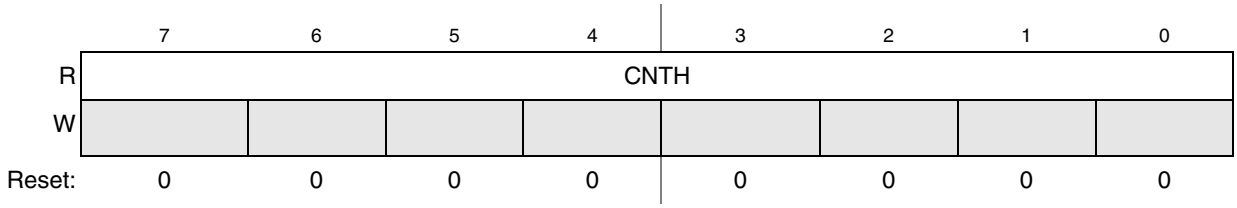


Figure 19-4. MTIMx16 Counter Register High (MTIMxxCNTH)

Table 19-4. MTIMxCNTH Field Descriptions

Field	Description
7:0 CNTH	**MTIM16 Count (High Byte)**— These eight read-only bits contain the current high byte value of the 16-bit counter. Writing has no effect to this register. Reset clears the register to 0x00.

MTIMxCNTL is the read-only value of the low byte of current MTIM16 16-bit counter.

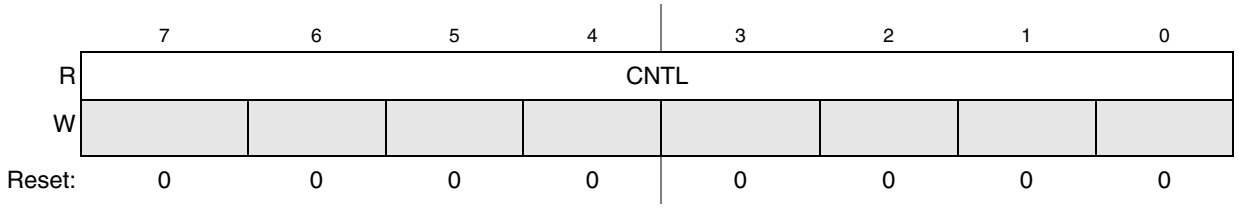


Figure 19-5. MTIM16 Counter Register Low (MTIMxCNTL)

Table 19-5. MTIMxCNTL Field Descriptions

Field	Description
7:0 CNTL	**MTIM16 Count (Low Byte)** — These eight read-only bits contain the current low byte value of the 16-bit counter. writing has no effect to this register. Reset clears the register to 0x00.

When either MTIMxCNTH or MTIMxCNTL is read, the content of the two registers is latched into a buffer where they remain latched until the other register is read.This allows the coherent 16-bit to be read in both big-endian and little-endian compile environments and ensures the 16-bit counter is unaffected by the read operation. The coherency mechanism is automatically restarted by an MCU reset or setting of TRST bit of MTIMxSC register (whether BDM mode is active or not).

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the counter register are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, the appropriate value from the other half of the 16-bit value will be read after returning to normal execution.The value read from the MTIMxCNTH and MTIMxCNTL registers in BDM mode is the value of these registers and not the value of their read buffer.

19.4.4 MTIM16 Modulo Register High/Low (MTIMxMODH/MTIMxMODL)

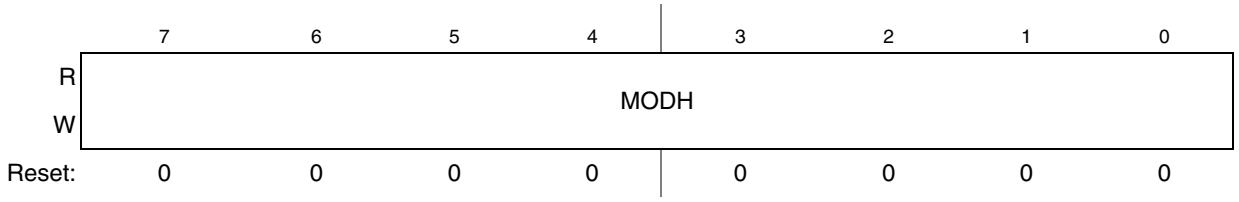


Figure 19-6. MTIM16 Modulo Register High (MTIMxMODH)

Table 19-6. MTIMxMODH Field Descriptions

Field	Description
7:0 MODH	**MTIM16 Modulo (High Byte)** — These eight read/write bits contain the modulo high byte value used to reset the counter and set TOF.Reset sets the register to 0x00.

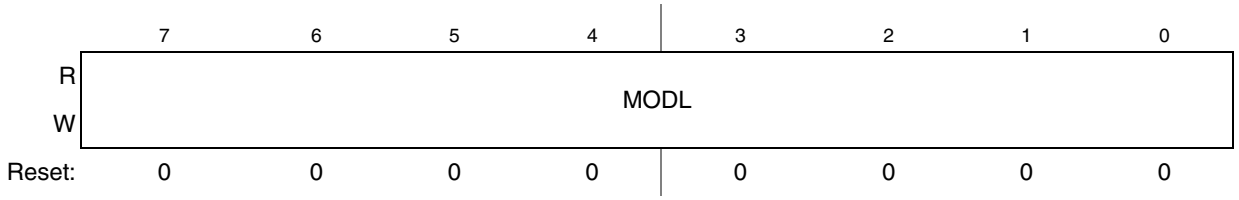


Figure 19-7. MTIM16 Modulo Register Low (MTIMxMODL)

Table 19-7. MTIMxMODL Field Descriptions

Field	Description
7:0 MODL	**MTIM16 Modulo (Low Byte)** — These eight read/write bits contain the modulo low byte value used to reset the counter and set TOF. Reset sets the register to 0x00.

A value of 0x0000 in MTIMxMODH:L puts the MTIM16 in free-running mode. Writing to either MTIMxMODH or MTIMxMODL latches the value into a buffer and the registers are updated with the value of their write buffer after the second byte writing, the updated MTIMxMODH:L will take effect in the next MITIM16 counter cycle except for the first writing of modulo after a chip reset or in BDM mode. But after a software reset, the MTIMxMODH:L takes effect at once even if it didn't take effect before the reset. On the first writing of MTIMxMODH:L after chip reset, the counter is reset and the modulo takes effect immediately. The latching mechanism may be manually reset by setting the TRST bit of MTIMxSC register (whether BDM is active or not).

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the modulo register are written while BDM is active. Any writing to the modulo registers bypasses the buffer latches and writes directly to the modulo register while BDM is active, and also the counter is cleared at the same time.The reading of MTIMxMODH:L returns the modulo value which is taking effect whenever in normal run mode or in BDM mode.

19.5 Functional Description

The MTIM16 is composed of a main 16-bit up-counter with 16-bit modulo register, a clock source selector, and a prescaler block with nine selectable values. The module also contains software selectable interrupt logic.

The MTIM16 counter (MTIMxCNTH:L) has three modes of operation: stopped, free-running, and modulo. The counter is stopped out of reset. If the counter starts without writing a new value to the modulo registers, it will be in free-running mode. The counter is in modulo mode when a value other than 0x0000 is in the modulo registers.

After an MCU reset, the counter stops and resets to 0x0000, and the modulo is also reseted to 0x0000. The bus clock functions as the default clock source and the prescale value is divided by 1. To start the MTIM16 in free-running mode, write to the MTIM16 status and control register (MTIMxSC) and clear the MTIM16 stop bit (TSTP).

Four clock sources are software selectable: the internal bus clock, the fixed frequency clock (XCLK), and an external clock on the TCLK pin, selectable as incrementing on either rising or falling edges. The

MTIM16 clock select bits (CLKS1:CLKS0) in MTIMxSC are used to select the desired clock source. If the counter is active (TSTP = 0) when a new clock source is selected, the counter continues counting from the previous value using the new clock source.

Nine prescale values are software selectable: clock source divided by 1, 2, 4, 8, 16, 32, 64, 128, or 256. The prescaler select bits (PS[3:0]) in MTIMxSC select the desired prescale value. If the counter is active (TSTP = 0) when a new prescaler value is selected, the counter continues counting from the previous value using the new prescaler value.

The MTIM16 modulo register (MTIMxMODH:L) allows the overflow compare value to be set to any value from 0x0001 to 0xFFFF. Reset clears the modulo value to 0x0000, which results in a free running counter.

When the counter is active (TSTP = 0), it increases at the selected rate until the count matches the modulo value. When these values match, the counter overflows to 0x0000 and continues counting. The MTIM16 overflow flag (TOF) is set whenever the counter overflows. The flag sets on the transition from the modulo value to 0x0000.

Clearing TOF is a two-step process. The first step is to read the MTIMxSC register while TOF is set. The second step is to write a 0 to TOF. If another overflow occurs between the first and second steps, the clearing process is reset and TOF stays set after the second step is performed. This will prevent the second occurrence from being missed. TOF is also cleared when a 1 is written to TRST.

The MTIM16 allows for an optional interrupt to be generated whenever TOF is set. To enable the MTIM16 overflow interrupt, set the MTIM16 overflow interrupt enable bit (TOIE) in MTIMxSC. TOIE should never be written to a 1 while TOF = 1. Instead, TOF should be cleared first, then the TOIE can be set to 1.

19.5.1 MTIM16 Operation Example

This section shows an example of the MTIM16 operation as the counter reaches a matching value from the modulo register.

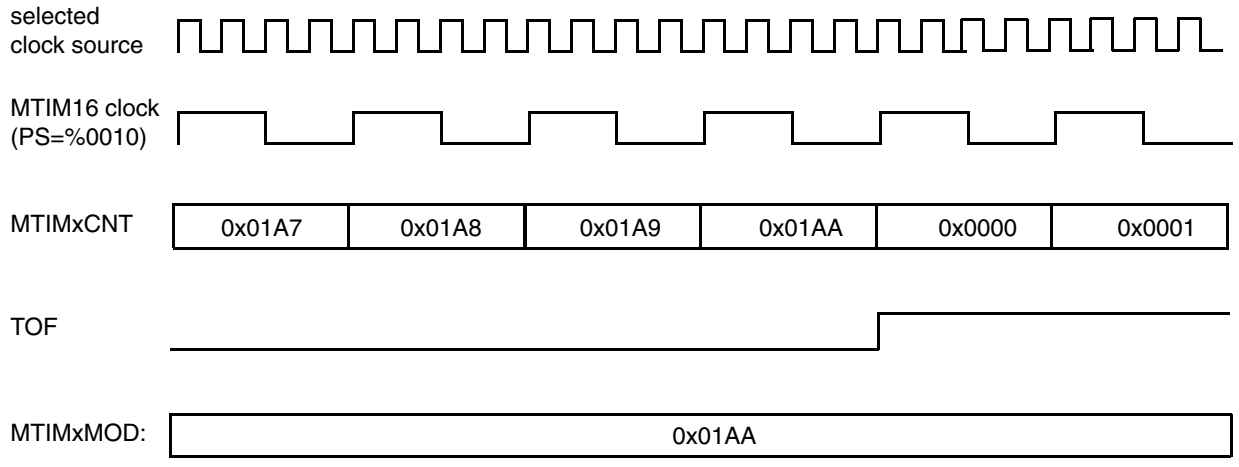


Figure 19-8. MTIM16 Counter Overflow Example

In the example of Figure 19-8, the selected clock source could be any of the four possible choices. The prescaler is set to PS = %0010 or divide-by-4. The modulo value in the MTIMxMODH:L register is set to 0x01AA. When the counter, MTIMxCNTH:L, reaches the modulo value of 0x01AA, the counter overflows to 0x0000 and continues counting. The timer overflow flag, TOF, sets when the counter value changes from 0x01AA to 0x0000. An MTIM16 overflow interrupt is generated when TOF is set, if TOIE = 1.

Chapter 20
Cyclic Redundancy Check (CRC)

20.1 Introduction

The CRC block provides hardware accelleration for CRC16-CCITT compliancy with $x^{16} + x^{12} + x^5 + 1$ polynomial. It is implemented on the 8-bit peripheral bus, and provides a very simple programming interface of only two 8-bit registers. The V1 ColdFire 8-bit peripheral bus bridge serializes 16-bit accesses into two 8-bit accesses, so both registers can be read/written using a single 16-bit read/write instruction.

This peripheral is available in both RUN and LPRUN modes.

NOTE

For details on low-power mode operation, refer to Table 6-4 in Chapter 6, "Modes of Operation".

20.1.1 Features

Features of the CRC module include:

- Hardware CRC generator circuit using 16-bit shift register
- CRC16-CCITT compliancy with $x^{16} + x^{12} + x^5 + 1$ polynomial
- Error detection for all single, double, odd, and most multi-bit errors
- Programmable initial seed value
- High-speed CRC calculation
- Optional feature to transpose input data and CRC result via transpose register, required on applications where bytes are in LSb (Least Significant bit) format.

20.1.2 Modes of Operation

This section defines the CRC operation in run, wait, and stop modes.

- Run Mode — This is the basic mode of operation.
- Wait Mode — The CRC module is operational.
- Stop 1 and 2 Modes- The CRC module is not functional in these modes and will be put into its reset state upon recovery from stop.
- Stop 3 Mode — In this mode, the CRC module will go into a low power standby state. Any CRC calculations in progress will stop and resume after the CPU goes into run mode.

20.1.3 Block Diagram

Figure 20-1 provides a block diagram of the CRC module.

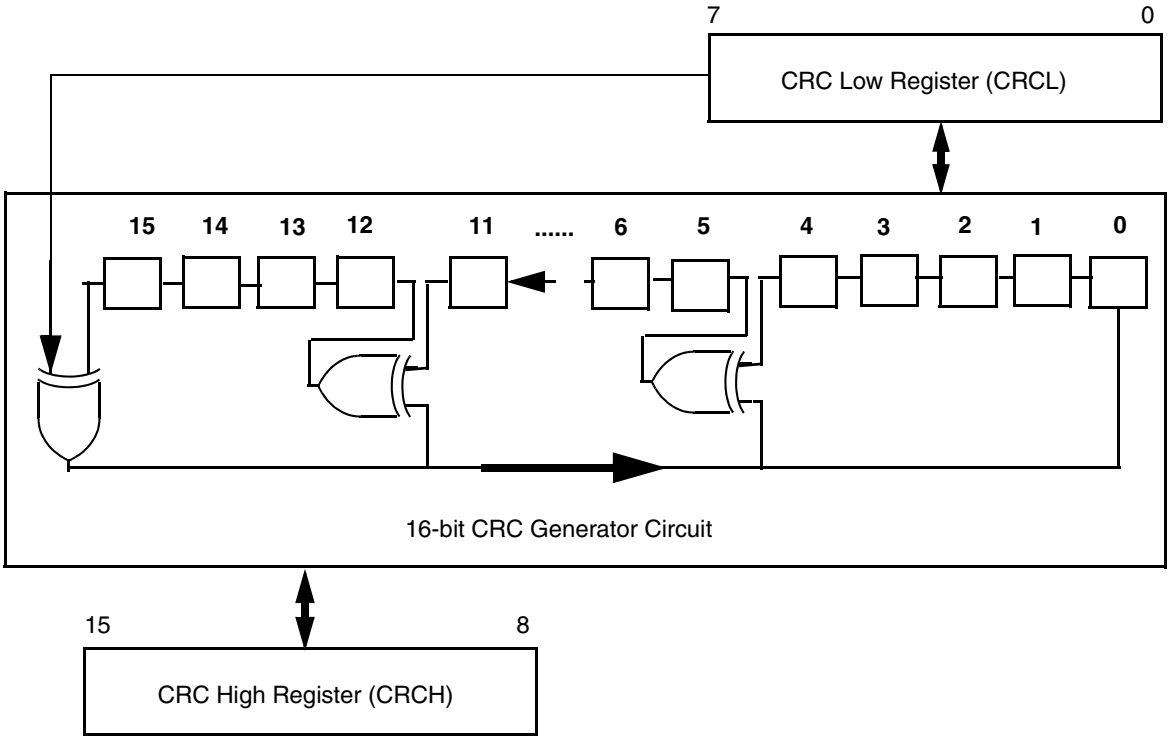


Figure 20-1. Cyclic Redundancy Check (CRC) Module Block Diagram

20.2 External Signal Description

There are no CRC signals that connect off chip.

20.3 Register Definition

20.3.1 Memory Map

Table 20-1. CRC Register Summary

Name		7	6	5	4	3	2	1	0
CRCH (offset=0)	R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	W								
CRCL (offset=1)	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	W								

Table 20-1. CRC Register Summary

Name		7	6	5	4	3	2	1	0
TRANSPOSE (offset=2)	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	W								

NOTE

Offsets 4,5,6 and 7 are also mapped onto the CRCL register. This is an *alias* only used on CF1Core (version 1 of ColdFire core) and should be ignored for HCS08 cores. See Section 20.4.2, “Programming model extension for CF1Core for more details.

20.3.2 Register Descriptions

The CRC module includes:

- A 16-bit CRC result and seed register (CRCH:CRCL)
- An 8-bit transpose register to convert from LSb to MSb format (or vice-versa) when required by the application

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all CRC registers. This section refers to registers only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

20.3.2.1 CRC High Register (CRCH)

	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 20-2. CRC High Register (CRCH)

Table 20-2. Register Field Descriptions

Field	Description
7:0 CRCH	**CRCH** -- This is the high byte of the 16-bit CRC register. A write to CRCH will load the high byte of the initial 16-bit seed value directly into bits 15-8 of the shift register in the CRC generator. The CRC generator will then expect the low byte of the seed value to be written to CRCL and loaded directly into bits 7-0 of the shift register. Once both seed bytes written to CRCH:CRCL have been loaded into the CRC generator, and a byte of data has been written to CRCL, the shift register will begin shifting. A read of CRCH will read bits 15-8 of the current CRC calculation result directly out of the shift register in the CRC generator.

20.3.2.2 CRC Low Register (CRCL)

	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 20-3. CRC High Register (CRCH)

Table 20-3. Register Field Descriptions

Field	Description
7:0 CRCL	**CRCL** — This is the low byte of the 16-bit CRC register. Normally, a write to CRCL will cause the CRC generator to begin clocking through the 16-bit CRC generator. As a special case, if a write to CRCH has occurred previously, a subsequent write to CRCL will load the value in the register as the low byte of a 16-bit seed value directly into bits 7-0 of the shift register in the CRC generator. A read of CRCL will read bits 7-0 of the current CRC calculation result directly out of the shift register in the CRC generator.

20.3.2.3 Transpose Register (TRANSPOSE)

	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 20-4. Transpose Register (CRCH)

Table 20-4. Register Field Descriptions

Field	Description
7:0 TRANSPOSE	**TRANSPOSE** — This register is used to transpose data, converting from LSb to MSb (or vice-versa). The byte to be transposed should be first written to TRANSPOSE and then subsequent reads from TRANSPOSE will return the transposed value of the last written byte (bit 7 becomes bit 0, bit 6 becomes bit 1, and so forth).

20.4 Functional Description

To enable the CRC function, a write to the CRCH register will trigger the first half of the seed mechanism which will place the CRCH value directly into bits 15-8 of the CRC generator shift register. The CRC generator will then expect a write to CRCL to complete the seed mechanism.

As soon as the CRCL register is written to, its value will be loaded directly into bits 7-0 of the shift register, and the second half of the seed mechanism will be complete. This value in CRCH:CRCL will be the initial seed value in the CRC generator.

Now the first byte of the data on which the CRC calculation will be applied should be written to CRCL. This write after the completion of the seed mechanism will trigger the CRC module to begin the CRC checking process. The CRC generator will shift the bits in the CRCL register (MSB first) into the shift register of the generator. After all 8 bits have been shifted into the CRC generator (in the next bus cycle after the data write to CRCL), the result of the shifting, or the value currently in the shift register, can be read directly from CRCH:CRCL, and the next data byte to include in the CRC calculation can be written to the CRCL register.

This next byte will then also be shifted through the CRC generator's 16-bit shift register, and after the shifting has been completed, the result of this second calculation can be read directly from CRCH:CRCL.

After each byte has finished shifting, a new CRC result will appear in CRCH:CRCL, and an additional byte may be written to the CRCL register to be included within the CRC16-CCITT calculation. A new CRC result will appear in CRCH:CRCL each time 8-bits have been shifted into the shift register.

To start a new CRC calculation, write to CRCH, and the seed mechanism for a new CRC calculation will begin again.

20.4.1 ITU-T (CCITT) Recommendations and Expected CRC Results

The CRC polynomial 0x1021 ($x^{16} + x^{12} + x^5 + 1$) is popularly known as *CRC-CCITT* since it was initially proposed by the ITU-T (formerly CCITT) committee.

Although the ITU-T recommendations are very clear about the polynomial to be used, 0x1021, they accept variations in the way the polynomial is implemented:

- ITU-T V.41 implements the same circuit shown in Figure 20-1, but it recommends a SEED = 0x0000.
- ITU-T T.30 and ITU-T X.25 implement the same circuit shown in Figure 20-1, but they recommend the final CRC result to be negated (one-complement operation). Also, they recommend a SEED = 0xFFFF.

Moreover, it is common to find circuits in literature slightly different from the one suggested by the recommendations above, but also known as CRC-CCITT circuits (many variations require the message to be augmented with zeros).

The circuit implemented in the CRC module is exactly the one suggested by the ITU-T V.41 recommendation, with an added flexibility of a programmable SEED. As in ITU-T V.41, no augmentation is needed and the CRC result is not negated. Table 20-5 shows some expected results that can be used as a reference. Notice that these are ASCII string messages. For example, message "123456789" is encoded with bytes 0x31 to 0x39 (see ASCII table).

Table 20-5. Expected CRC results

ASCII String Message	SEED (initial CRC value)	CRC result
"A"	0x0000	0x58e5
"A"	0xffff	0xb915

“A”	0x1d0f[1]	0x9479
“123456789”	0x0000	0x31c3
“123456789”	0xffff	0x29b1
“123456789”	0x1d0f[1]	0xe5cc
A string of 256 upper case “A” characters with no line breaks	0x0000	0xabe3
A string of 256 upper case “A” characters with no line breaks	0xffff	0xea0b
A string of 256 upper case “A” characters with no line breaks	0x1d0f[1]	0xe938

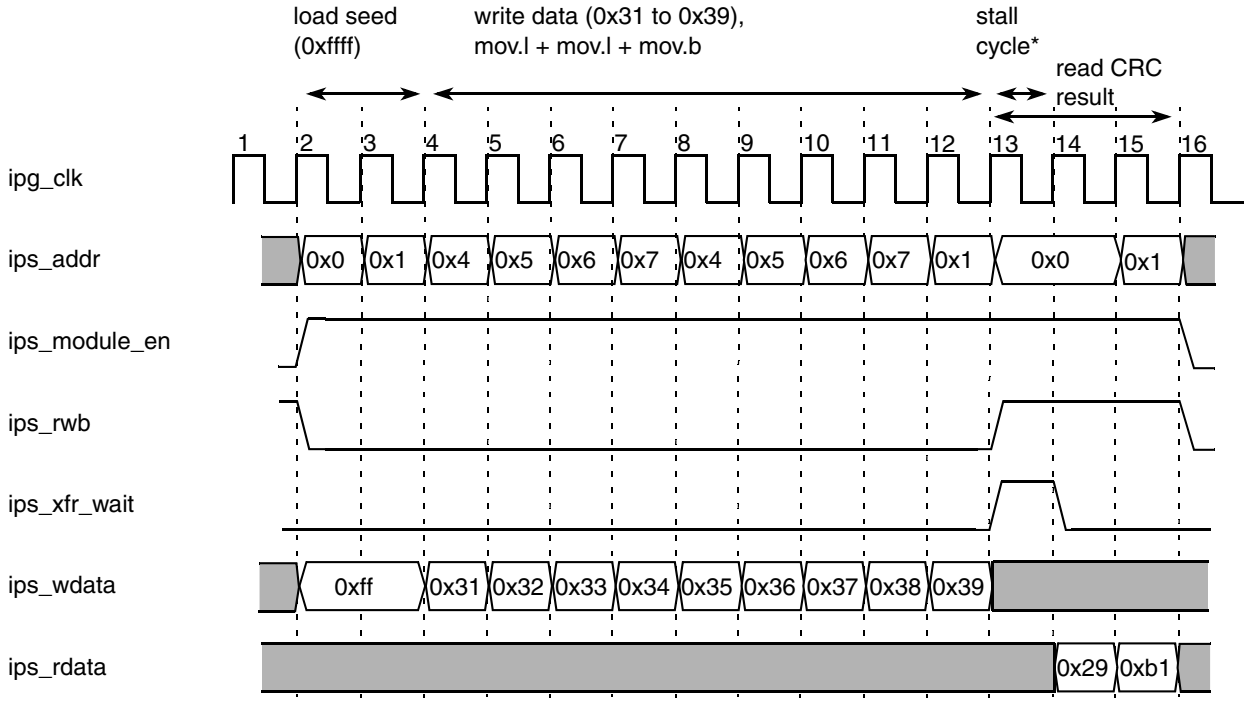
[1] One common variation of CRC-CCITT require the message to be augmented with zeros and a SEED=0xFFFF. The CRC module will give the same results of this alternative implementation when SEED=0x1D0F and no message augmentation.

20.4.2 Programming model extension for CF1Core

The CRC module extends its original programming model to allow faster CRC calculations on CF1 cores. Memory offsets 0x4, 0x5, 0x6 and 0x7 are mapped (aliased) onto the CRCL register, in a way that the CF1Core can execute 32-bit store instructions to write data to the CRC module. The code should use a single *mov.l* store instruction to send four bytes beginning at address 0x4, which will be decomposed by the platform into four sequential/consecutive byte writes to offsets 0x4-0x7 (all aliased to the CRCL position).

In addition, reads from 0x4-0x7 return the contents of the CRCL register. Reads from 0x2-0x3 (unused/reserved) return 0x00. Writes to 0x2-0x3 have no effect.

Due to internal CF1Core characteristics, this approach provides a greater data transfer rate than the original programming model used on HCS08 (single byte writes to CRCL position). Figure 20-5 illustrates a message calculation on CF1Core.



* On cycle 13, there is a read-after-write hazard, since calculation of 0x39 data is underway. *ips_xfr_wait* is asserted to signalize a stall cycle (the IPS master should wait until cycle 14 to read the CRC result).

Figure 20-5. CRC calculation of ASCII message “123456789” (0x31 to 0x39) on CF1Core

20.4.3 Transpose feature

The CRC module provides an optional feature to transpose data (invert bit order). This feature is specially useful on applications where the LSb format is used, since the CRC CCITT expects data in the MSb format. In that case, before writing new data bytes to CRCL, these bytes should be transposed as follows:

1. Write data byte to TRANSPOSE register
2. Read data from TRANSPOSE register (subsequent reads will result in the transposed value of the last written data)
3. Write transposed byte to CRCL.

After all data is fed into CRC, the CRCH:CRCL result is available in the MSb format. Then, these two bytes should also be transposed: the values read from CRCH:CRCL should be written/read to/from the TRANSPOSE register.

Although the transpose feature was initially designed to address LSb applications interfacing with the CRC module, it is important to notice that this feature is not necessarily tied to CRC applications. Since it was designed as an independent register, any application should be able to transpose data by writing/reading to/from the TRANSPOSE register (e.g.: Big endian / Little endian conversion in USB).

20.5 Initialization Information

To initialize the CRC Module and initiate a CRC16-CCITT calculation, follow this procedure:

1. Write high byte of initial seed value to CRCH.
2. Write low byte of initial seed value to CRCL.
3. Write first byte of data on which CRC is to be calculated to CRCL (an alternative option is offered for CF1Cores. See Section 20.4.2, “Programming model extension for CF1Core).
4. In the next bus cycle after step 3, if desired, the CRC result from the first byte can be read from CRCH:CRCL.
5. Repeat steps 3-4 until the end of all data to be checked.

Chapter 21
Analog-to-Digital Converter (ADC16)

21.1 Introduction

The 16-bit analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

NOTE

The registers APCTL1, APCTL2, APCTL3 and APCTL4 have no functionality in the MCF51EM256 series. The pin control function is performed by the Mux control, please refer to Section 4.7, "Pin Mux Controls" for further information.

V_{BG} is from the MCU voltage regulator and V_{alt} is from V_{REF} module.

Ignore any references to stop1 low-power mode in this chapter, because this device does not support it.

For details on low-power mode operation, refer to Table 6-4 in Chapter 6, "Modes of Operation".

21.1.1 ADC Clock Gating

The bus clock to each ADC can be gated on and off using the SCGC1_ADC*x* bits (see Section 7.7.9, "System Clock Gating Control 1 Register (SCGC1)"). These bits are set after any reset, which enables the bus clock to this module. To conserve power, the SCGC1_ADC*x* can be cleared to disable the clock to this module when not in use. See Section 7.6, "Peripheral Clock Gating," for details.

21.1.2 Hardware Trigger

The hardware triggers of ADC1, ADC2, ADC3 and ADC4 are provided from PDB channel 1, 2, 3 and 4 correspondingly, when ADTRG is set in ADC*x*SC2.

When enabled, ADC*x* will be triggered every time PDB channel *x* TriggerA output is asserted. The PDB channel *x* PreTriggerA and PreTriggerB will set the ADHWTSA and ADHWTSB correspondingly.

For details on ADC hardware trigger function, refer to Section 21.5.4, "Hardware Trigger and Channel Selects". For details on PDB, refer Chapter 22, "Programmable Delay Block (PDB)."

21.1.3 ADC Alternate Clock

The ADC is capable of performing conversions by using the MCU bus clock, the bus clock divided by two, the local asynchronous clock (ADACK) in the module, or the alternate clock (ALTCLK). The ALTCLK on the MCF51EM256 series is connected to the ICSERCLK.

21.2 ADC Connections

The MCF51EM256 series each includes four separately controllable ADCs. Assignment of device ADC pins is spread over each of the available ADCs as shown in Table 21-1.

Table 21-1. ADC Channel Assignments

Pins	Function	ADC	ADCH
DADP0	Differential Analog Channel Input	ADC3	00000 (0)
DADM0			
DADP1	Differential Analog Channel Input	ADC1	00001 (1)
DADM1			
DADP2	Differential Analog Channel Input	ADC2	00010 (2)
DADM2			
DADP3	Differential Analog Channel Input	ADC4	00011 (3)
DADM3			
AD4	Analog Channel input	ADC3	00100 (4)
AD5	Analog Channel input	ADC1	00101 (5)
AD6	Analog Channel input	ADC2	00110 (6)
AD7	Analog Channel input	ADC4	00111 (7)
AD8	Analog Channel input	ADC3	01000 (8)
AD9	Analog Channel input	ADC3	01001 (9)
AD10	Analog Channel input	ADC1	01010 (10)
AD11	Analog Channel input	ADC1	01011 (11)
AD12	Analog Channel input	ADC1	01100 (12)
AD13	Analog Channel input	ADC1, ADC3	01101 (13)
AD14	Analog Channel input	ADC2	01110 (14)
AD15	Analog Channel input	ADC2	01111 (15)
AD16[1]	Analog Channel input	ADC2	10000 (16)
AD17[1]	Analog Channel input	ADC2, ADC4	10001 (17)
AD18[1]	Analog Channel input	ADC2, ADC4	10010 (18)
AD19[1]	Analog Channel input	ADC2, ADC4	10011 (19)

Table 21-1. ADC Channel Assignments (continued)

Pins	Function	ADC	ADCH
—	VREFO	ADC1, ADC2, ADC3, ADC4	10100 (20)
—	V_{LL1}	ADC2	11000 (24)
—	Temperature Sensor	ADC1, ADC2, ADC3, ADC4	11010 (26)
—	Bandgap	ADC1, ADC2, ADC3, ADC4	11011 (27)
V_{REFH}	Voltage reference High	ADC1, ADC2, ADC3, ADC4	11101 (29)
V_{REFL}	Voltage reference Low	ADC1, ADC2, ADC3, ADC4	11110 (30)

[1] Users must not select this pin as a channel if $V_{LL3} > V_{DDA}$

21.2.1 Features

Features of the ADC module include:

- Linear successive approximation algorithm with up to 16-bit resolution
- Up to four pairs of differential and 24 single-ended external analog inputs
- Output Modes:
  - Differential 16-bit, 13-bit, 11-bit, and 9-bit modes
  - Single-ended 16-bit, 12-bit, 10-bit, and 8-bit modes
- Output formatted in 2’s complement 16b sign extended for differential modes
- Output in right-justified unsigned format for single-ended
- Single or continuous conversion (automatic return to idle after single conversion)
- Configurable sample time and conversion speed/power
- Conversion complete / Hardware average complete flag and interrupt
- Input clock selectable from up to four sources
- Operation in wait or stop3 modes for lower noise operation
- Asynchronous clock source for lower noise operation with option to output the clock
- Selectable asynchronous hardware conversion trigger with hardware channel select
- Automatic compare with interrupt for less-than, greater-than or equal-to, within range, or out-of-range, programmable value
- Temperature sensor
- Hardware average function
- Selectable voltage reference
  - Internal
  - External
  - Alternate
- Self-Calibration mode

21.2.2 Block Diagram

Figure 21-1 provides a block diagram of the ADC module.

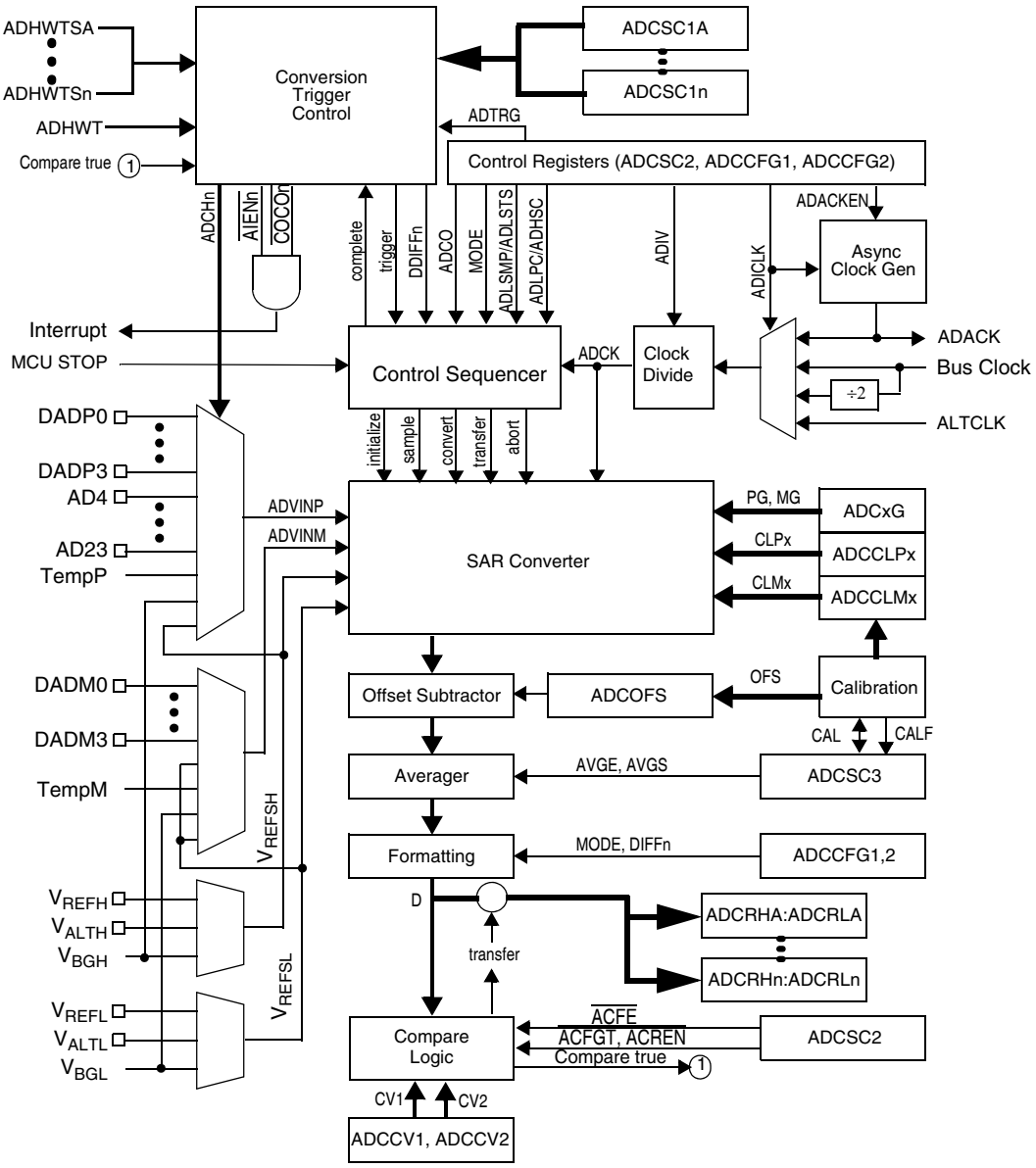


Figure 21-1. ADC Block Diagram

21.3 External Signal Description

The ADC module supports up to four pairs of differential inputs and 24 single-ended inputs. Each differential pair requires two inputs, DADPx and DADMx.The ADC also requires four supply/reference/ground connections.

Table 21-2. Signal Properties

Name	Function
DADP0-DADP3	Differential Analog Channel Inputs
DADM0-DADM3	Differential Analog Channel Inputs

Table 21-2. Signal Properties (continued)

Name	Function
AD4–AD23	Analog Channel inputs
V_{REFSH}	Voltage Reference Select High
V_{REFSL}	Voltage Reference Select Low
V_{DDAD}	Analog power supply
V_{SSAD}	Analog ground

21.3.1 Analog Power (V_{DDAD})

The ADC analog portion uses V_{DDAD} as its power connection. In some packages, V_{DDAD} is connected internally to V_{DD}. If externally available, connect the V_{DDAD} pin to the same voltage potential as V_{DD}. External filtering may be necessary to ensure clean V_{DDAD} for good results.

21.3.2 Analog Ground (V_{SSAD})

The ADC analog portion uses V_{SSAD} as its ground connection. In some packages, V_{SSAD} is connected internally to V_{SS}. If externally available, connect the V_{SSAD} pin to the same voltage potential as V_{SS}.

21.3.3 Voltage Reference Select High (V_{REFSH})

V_{REFSH} is the high-reference voltage for the converter.

The ADC can be configured to accept one of three voltage reference pairs for V_{REFSH}. Each pair contains a positive reference which must be between the minimum Ref Voltage High (defined in data sheet) and V_{DDAD}, and a ground reference which must be at the same potential as V_{SSAD}.

The three pairs are:

- external (V_{REFH} and V_{REFL})
- alternate (V_{ALTH} and V_{ALTL})
- internal bandgap (V_{BGH} and V_{BGL})

These voltage references are selected using the REFSEL bits in the ADCSC2 register.The alternate (V_{ALTH} and V_{ALTL}) voltage reference pair may select additional external pins or internal sources depending on MCU configuration.Consult the module introduction for information on the Voltage References specific to this MCU.

In some packages, V_{REFH} is connected in the package to V_{DDAD}. If externally available, the positive reference(s) may be connected to the same potential as V_{DDAD} or may be driven by an external source to a level between the minimum Ref Voltage High (defined in the Data Sheet) and the V_{DDAD} potential (V_{REFH} must never exceed V_{DDAD}).

21.3.4 Voltage Reference Select Low (V_{REFL})

V_{REFSL} is the low reference voltage for the converter.The ADC can be configured to accept one of three voltage reference pairs for V_{REFSL}. Each pair contains a positive reference which must be between the minimum Ref Voltage High (defined in Appendix A) and V_{DDAD}, and a ground reference which must be at the same potential as V_{SSAD}. The three pairs are external (V_{REFH} and V_{REFL}), alternate (V_{ALTH} and V_{ALTL}) and the internal bandgap (V_{BGH} and V_{BGL}). These voltage references are selected using the REFSEL bits.The alternate (V_{ALTH} and V_{ALTL}) voltage reference pair may select additional external pins or internal sources depending on MCU configuration.Consult the module introduction for information on the Voltage References specific to this MCU.

In some packages, V_{REFL} is connected in the package to V_{SSAD}. If externally available, connect the ground reference(s) to the same voltage potential as V_{SSAD}.

21.3.5 Analog Channel Inputs (ADx)

The ADC module supports up to 24 single-ended analog inputs. A single-ended input is selected for conversion through the ADCHn channel select bits when the DIFFn bit in the ADCSC1n register is low.

21.3.6 Differential Analog Channel Inputs (DADx)

The ADC module supports up to four differential analog channel inputs. Each differential analog input is a pair of external pins (DADPx and DADMx) referenced to each other to provide the most accurate analog to digital readings. A differential input is selected for conversion through the ADCHn channel select bits when the DIFFn bit in the ADCSC1n register bit is high.

21.4 Register Definition

These memory-mapped registers control and monitor operation of the ADC:

- Status and channel control registers, ADCSC1A:ADCSC1n
- Configuration registers, ADCCFG1 and ADCCFG2
- Data result registers, ADCRHA:ADCRLA to ADCRHn:ADCRLn
- Compare value registers, ADCCV1H, ADCCV1L, ADCCV2H, and ADCCV2L
- General status and control registers, ADCSC2 and ADCSC3
- Configuration registers, ADCCFG1 and ADCCFG2
- Offset Correction Registers, ADCOFSH and ADCOFSL
- Plus-input gain registers, ADCPGH and ADCPGL
- Minus-input gain registers, ADCMGH and ADCMGL
- Plus-side general calibration registers, ADCCLP0, ADCCLP1, ADCCLP2, ADCCLP3H, ADCCLP3L, ADCCLP4H, ADCCLP4L, ADCCLSP, ADCCLDP
- Minus-side general calibration registers, ADCCLM0, ADCCLM1, ADCCLM2, ADCCLM3H, ADCCLM3L, ADCCLM4H, ADCCLM4L, ADCCLSM, ADCCLDM
- Pin enable registers, APCTL1, APCTL2, APCTL3, and APCTL4

21.4.1 Status and Control Registers 1 (ADCSC1A:ADCSC1n)

This section describes the function of the ADC status and channel control registers, ADCSC1A through ADCSC1n. ADCSC1A is used for both software and hardware trigger modes of operation. ADCSC1B-ADCSC1n indicate potentially multiple ADCSC1 registers for use only in hardware trigger mode. Consult the module introduction for information on the number of ADCSC1n registers specific to this MCU. The ADCSC1A to ADCSC1n registers have identical fields, and are used in a “ping-pong” approach to control ADC operation. At any one point in time, only one of the ADCSC1A to ADCSC1n registers is actively controlling ADC conversions. Updating ADCSC1A while ADCSC1n is actively controlling a conversion is allowed (and vice-versa for any of the ADCSC1n registers specific to this MCU). Writing ADCSC1A while ADCSC1A is actively controlling a conversion aborts the current conversion. In software trigger mode (ADTRG=0), writes to ADCSC1A subsequently initiates a new conversion (if the ADCHn bits are equal to a value other than all 1s). Similarly, writing any of the ADCSC1n registers while that specific ADCSC1n register is actively controlling a conversion aborts the current conversion. Any of the ADCSC1B -ADCSC1n registers are not used for software trigger operation and therefore writes to the ADCSC1B -ADCSC1n registers do not initiate a new conversion.

	7	6	5	4	3	2	1	0
R	COCOn	AIENn	DIFFn	ADCHn				
W								
Reset:	0	0	0	1	1	1	1	1

Figure 21-2. Status and Channel Control Register 1n (ADCSC1n)

Table 21-3. ADCSC1:ADCSC1n Field Descriptions

Field	Description
7 COCOn	**Conversion Complete Flag** - The COCOn flag is a read-only bit that is set each time a conversion is completed when the compare function is disabled (ACFE=0) and the hardware average function is disabled (AVGE=0). When the compare function is enabled (ACFE=1), the COCOn flag is set upon completion of a conversion only if the compare result is true. When the hardware average function is enabled (AVGE=1), the COCOn flag is set upon completion of the selected number of conversions (determined by the AVGS bits). The COCO1 flag will also set at the completion of a Calibration sequence. The COCOn bit is cleared when the respective ADCSCn is written or when the respective ADCRLn is read. 0 Conversion not completed 1 Conversion completed
6 AIENn	**Interrupt Enable** - AIENn enables conversion complete interrupts. When COCOn becomes set while the respective AIENn is high, an interrupt is asserted. 0 Conversion complete interrupt disabled 1 Conversion complete interrupt enabled

Table 21-3. ADCSC1:ADCSC1n Field Descriptions (continued)

Field	Description
5 DIFFn	**Differential Mode Enable** - DIFFn configures the ADC to operate in differential mode. When enabled this mode automatically selects from the differential channels, changes the conversion algorithm and the number of cycles to complete a conversion. 0 Single-ended conversions and input channels are selected 1 Differential conversions and input channels are selected
4:0 ADCHn[4:0]	**Input Channel Select** - The ADCHn bits form a 5-bit field that selects one of the input channels. The input channel decode is dependent upon the value of the DIFFn bit as detailed in Table 21-4. The successive approximation converter subsystem is turned off when the channel select bits are all set(ADCHn = 11111). This feature allows for explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way prevents an additional, single conversion from being performed. It is not necessary to set the channel select bits to all ones to place the ADC in a low-power state when continuous conversions are not enabled because the module automatically enters a low-power state when a conversion completes.

Table 21-4. Input Channel Select

ADCHn	Input Selected when DIFFn=0	Input Selected when DIFFn=1
00000–00011	DADP0-DADP3	DAD0-DAD3[1]
00100-10111	AD4-AD23	Reserved
11000-11001	Reserved	Reserved
11010	Temp Sensor (single-ended)	Temp Sensor (differential)
11011	Bandgap (single-ended)	Bandgap (differential)
11100	Reserved	Reserved
11101	V_{REFSH}[2]	$-V_{REFSH}$[2] (differential)
11110	V_{REFSL}[2]	Reserved
11111	Module disabled	

1 DAD0-DAD3 are associated with the input pin pairs DADPx and DADMx.

2 Voltage Reference selected is determined by the REFSEL bits in the ADCSC2 register. Refer to Section 21.5.3 for more information on voltage reference selection.

21.4.2 Configuration Register 1 (ADCCFG1)

ADCCFG1 selects the mode of operation, clock source, clock divide, and configure for low power or long sample time.

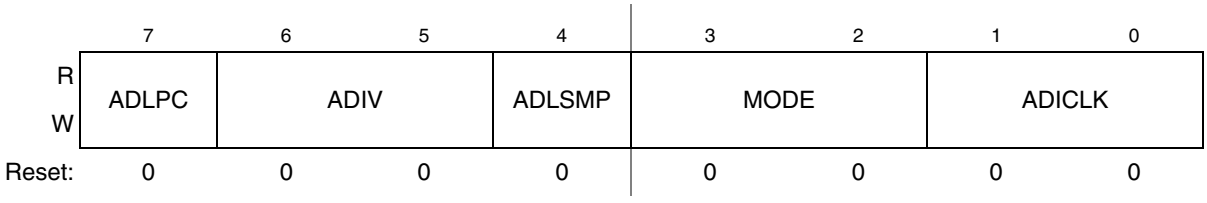


Figure 21-3. Configuration Register (ADCCFG1)

Table 21-5. ADCCFG1 Register Field Descriptions

Field	Description
7 ADLPC	**Low-Power Configuration -** ADLPC controls the power configuration of the successive approximation converter. This optimizes power consumption when higher sample rates are not required. 0 Normal power configuration 1 Low-power configuration: The power is reduced at the expense of maximum clock speed.
6:5 ADIV[6:5]	**Clock Divide Select -** ADIV selects the divide ratio used by the ADC to generate the internal clock ADCK. Table 21-6 shows the available clock configurations.
4 ADLSMP	**Sample Time Configuration -** ADLSMP selects between different sample times based on the conversion mode selected. This bit adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required. **When ADLSMP=1, the Long Sample Time Select bits (**ADLSTS[1:0]) can select the extent of the long sample time. 0 Short sample time 1 Long sample time (The ADLTS bits can select the extent of the long sample time)
3:2 MODE[3:2]	**Conversion Mode Selection -** MODE bits are used to select between the ADC resolution mode. See Table 21-7.
1:0 ADICLK[1:0]	**Input Clock Select -** ADICLK bits select the input clock source to generate the internal clock ADCK. See Table 21-8.

Table 21-6. Clock Divide Select

ADIV	Divide Ratio	Clock Rate
00	1	Input clock
01	2	Input clock ÷ 2
10	4	Input clock ÷ 4
11	8	Input clock ÷ 8

Table 21-7. Conversion Modes

MODE	DIFFn	Conversion Mode Description
00	0	single-ended 8-bit conversion
00	1	Differential 9-bit conversion with 2s complement output
01	0	single-ended 12-bit conversion
01	1	Differential 13-bit conversion with 2s complement output
10	0	single-ended 10-bit conversion
10	1	Differential 11-bit conversion with 2s complement output
11	0	single-ended 16-bit conversion
11	1	Differential 16-bit conversion with 2s complement output

Table 21-8. Input Clock Select

ADICLK	Selected Clock Source
00	Bus clock
01	Bus clock divided by 2
10	Alternate clock (ALTCLK)
11	Asynchronous clock (ADACK)

21.4.3 Configuration Register 2 (ADCCFG2)

ADCCFG2 selects differential mode, the special high speed configuration for very high speed conversions, and selects the long sample time duration during long sample mode.

	7	6	5	4	3	2	1	0
R	0	0	0	0	ADACKEN	ADHSC	ADLSTS	
W								
Reset:	0	0	0	0	0	0	0	0

Figure 21-4. Configuration Register 2(ADCCFG2)

Table 21-9. ADCCFG2 Register Field Descriptions

Field	Description
3 ADACKEN	**Asynchronous clock output enable** - ADACKEN enables the ADC's asynchronous clock source and the clock source output regardless of the conversion and input clock select (ADICLK bits) status of the ADC. Based on MCU configuration the asynchronous clock may be used by other modules (see module introduction section). Setting this bit allows the clock to be used even while the ADC is idle or operating from a different clock source. Also, latency of initiating a single or first-continuous conversion with the asynchronous clock selected is reduced since the ADACK clock is already operational. 0 Asynchronous clock output disabled; Asynchronous clock only enabled if selected by ADICLK and a conversion is active 1 Asynchronous clock and clock output enabled regardless of the state of the ADC
2 ADHSC	**High Speed Configuration- ADHSC configures the ADC for very high speed operation. The conversion sequence is altered (4 ADCK cycles added to the conversion time) to allow higher speed conversion clocks.** 0 Normal conversion sequence selected 1 High speed conversion sequence selected (4 additional ADCK cycles to total conversion time)
1:0 ADLSTS	**Long Sample Time Select -** ADLSTS selects between the extended sample times when long sample time is selected (ADLSMP=1). This allows higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required. 00 Default longest sample time (20 extra ADCK cycles; 24 ADCK cycles total) 01 12 extra ADCK cycles; 16 ADCK cycles total sample time 10 6 extra ADCK cycles; 10 ADCK cycles total sample time 11 2 extra ADCK cycles; 6 ADCK cycles total sample time

21.4.4 Data Result Registers (ADCRHA:ADCRLA to ADCRHn:ADCRLn)

The Data Result Registers (ADCRHA:ADCRLA to ADCRHn:ADCRLn) contain the result of an ADC conversion of the channel selected by the respective status and channel control register (ADCSC1A:ADCSC1n). For every ADCSC1A:ADCSC1n status and channel control register, there is a respective ADCRHA:ADCRLA to ADCRHn:ADCRLn data result register. Consult the module introduction for information on the number of ADCRHn:ADCRLn registers specific to this MCU.Reading ADCRHn prevents the ADC from transferring subsequent conversion results into the result registers until ADCRLn is read. If ADCRLn is not read until after the next conversion is completed, the intermediate conversion result is lost. In 8-bit single-ended mode, there is no interlocking with ADCRLn.

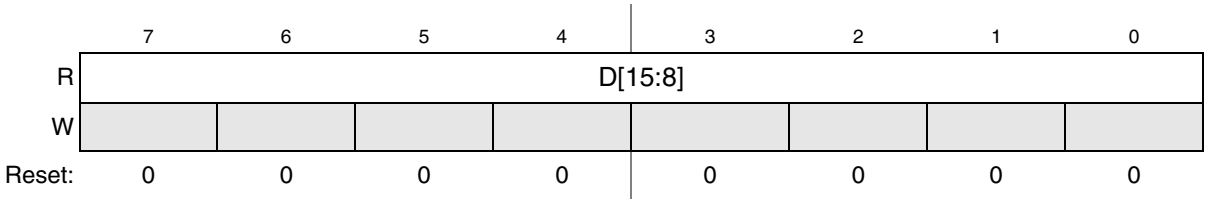


Figure 21-5. Data Result High Register (ADCRHn)

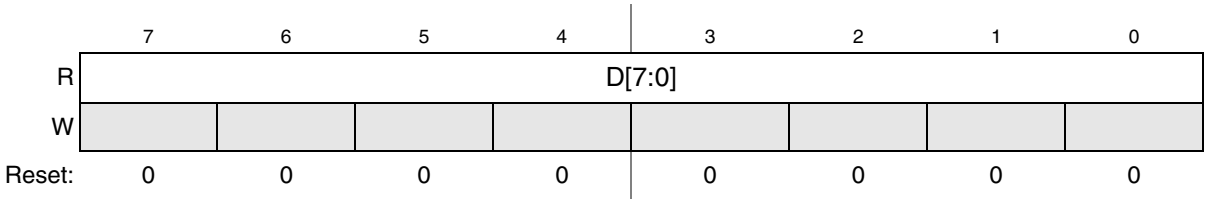


Figure 21-6. Data Result Low Register (ADCRLn)

ADCRHn contains the upper bits of the result of a conversion based on the conversion mode. ADCRLn contains the lower eight bits of the result of a conversion, or all eight bits of an 8-bit single-ended conversion. Unused bits in the ADCRHn register are cleared in unsigned right justified modes and carry the sign bit (MSB) in sign extended 2's complement modes. For example when configured for 10-bit single-ended mode, D[15:10] are cleared. When configured for 11-bit differential mode, D[15:10] carry the sign bit (bit 10 extended through bit 15).

Table 21-10 describes the behavior of the data result registers in the different modes of operation.

Table 21-10. Data Result Register Description

Conversion Mode	Data Result Register bits																Format
	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
16b differential	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	signed 2's complement
16b single-ended	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	unsigned right justified
13b differential	S	S	S	S	D	D	D	D	D	D	D	D	D	D	D	D	sign extended 2's complement
12b single-ended	0	0	0	0	D	D	D	D	D	D	D	D	D	D	D	D	unsigned right justified
11b differential	S	S	S	S	S	S	D	D	D	D	D	D	D	D	D	D	sign extended 2's complement

Table 21-10. Data Result Register Description

Conversion Mode	Data Result Register bits																Format
	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
10b single-ended	0	0	0	0	0	0	D	D	D	D	D	D	D	D	D	D	unsigned right justified
9b differential	S	S	S	S	S	S	S	S	D	D	D	D	D	D	D	D	sign extended 2's complement
8b single-ended	0	0	0	0	0	0	0	0	D	D	D	D	D	D	D	D	unsigned right justified

S: Sign bit or sign bit extension.
D: Data (2's complement data if indicated).

21.4.5 Compare Value Registers (ADCCV1H:ADCCV1L and ADCCV2H:ADCCV2L)

The Compare Value Registers (ADCCV1H:ADCCV1L & ADCCV2H:ADCCV2L) contain a compare value used to compare with the conversion result when the compare function is enabled (ACFE=1). This register is formatted the same for both bit position definition and value format (unsigned or sign-extended 2's complement) as the Data Result Registers (ADCRHn:ADCRLn) in the different modes of operation (See Table 21-10). Therefore, the compare function only uses the compare value register bits that are related to the ADC mode of operation.

The compare value 2 registers (ADCCV2H:ADCCV2L) are utilized only when the compare range function is enabled (ACREN=1).

In all modes except 8-bit single-ended conversions, the ADCCV1H register holds the upper bits of the first compare value. In 8-bit single-ended mode, ADCCV1H is not used during compare. In all conversion modes, the ADCCV1L register holds the lower 8 bits of the first compare value. The compare function is further detailed in Section 21.5.6.

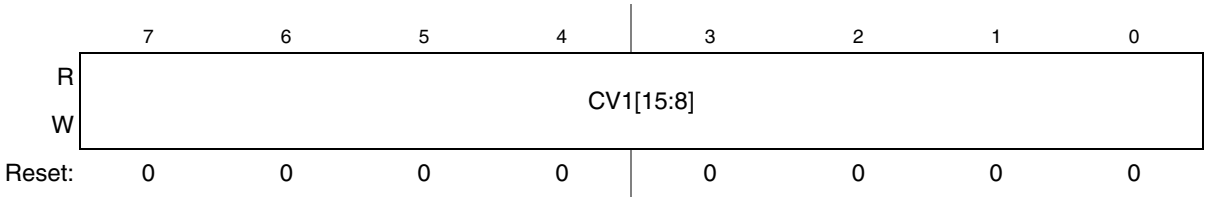


Figure 21-7. Compare Value 1 High Register (ADCCV1H)

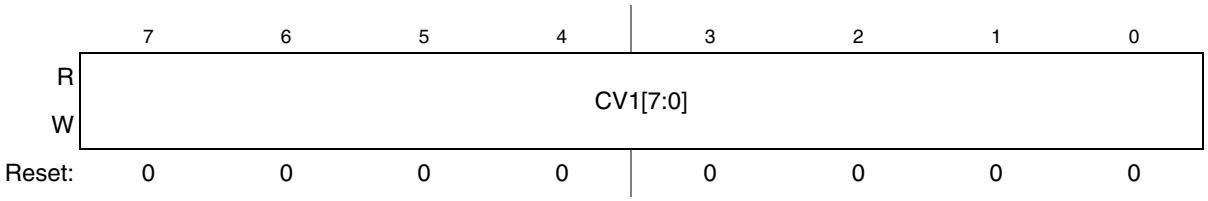


Figure 21-8. Compare Value 1 Low Register(ADCCV1L)

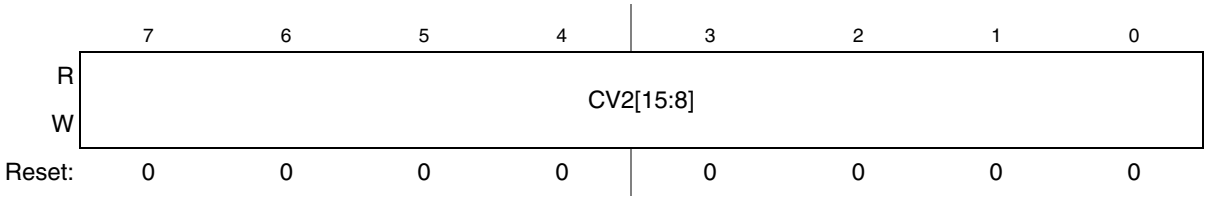


Figure 21-9. Compare Value 2 High Register (ADCCV2H)

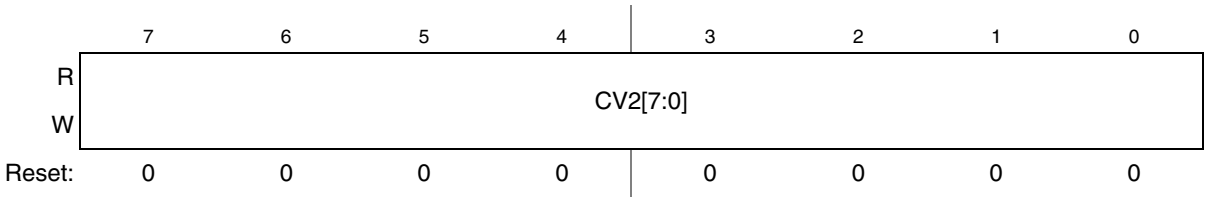


Figure 21-10. Compare Value 2Low Register(ADCCV2L)

21.4.6 Status and Control Register 2 (ADCSC2)

The ADCSC2 register contains the conversion active, hardware/software trigger select, compare function and voltage reference select of the ADC module.

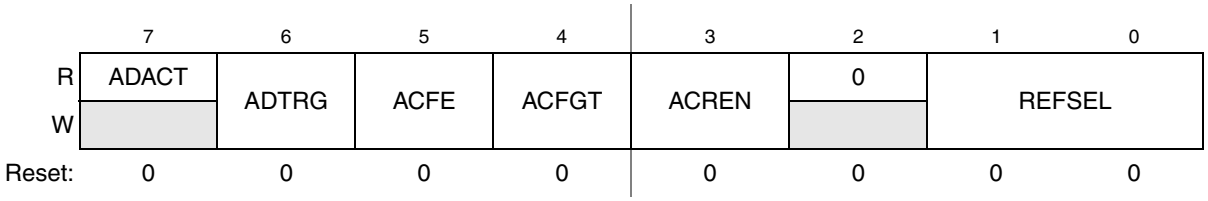


Figure 21-11. Status and Control Register 2 (ADCSC2)

Table 21-11. ADCSC2 Register Field Descriptions

Field	Description
7 ADACT	**Conversion Active** - ADACT indicates that a conversion or hardware averaging is in progress. ADACT is set when a conversion is initiated and cleared when a conversion is completed or aborted. 0 Conversion not in progress 1 Conversion in progress
6 ADTRG	**Conversion Trigger Select** - ADTRG selects the type of trigger used for initiating a conversion. Two types of trigger are selectable: software trigger and hardware trigger. When software trigger is selected, a conversion is initiated following a write to ADCSC1A. When hardware trigger is selected, a conversion is initiated following the assertion of the ADHWT input after a pulse of the ADHWTSn input.Refer to Section 21.5.5.1 for more information on initiating conversions. 0 Software trigger selected 1 Hardware trigger selected
5 ACFE	**Compare Function Enable** - ACFE enables the compare function. 0 Compare function disabled 1 Compare function enabled

Table 21-11. ADCSC2 Register Field Descriptions (continued)

Field	Description
4 ACFGT	**Compare Function Greater Than Enable** - ACFGT configures the compare function to check the conversion result relative to the compare value register(s) (ADCCV1H:ADCCV1L and ADCCV2H:ADCCV2L) based upon the value of ACREN. The ACFE bit must be set for ACFGT to have any effect. The compare function modes are further detailed in Table 21-24 in Section 21.5.6. 0 Configures Less Than Threshold, Outside Range Not Inclusive and Inside Range Not Inclusive functionality based on the values placed in the ADCCV1 and ADCCV2 registers. 1 Configures GreaterThan Or EqualTo Threshold, Outside Range Inclusive and Inside Range Inclusive functionality based on the values placed in the ADCCV1 and ADCCV2 registers.
3 ACREN	**Compare Function Range Enable** - ACREN configures the compare function to check the conversion result of the input being monitored is either between or outside the range formed by the compare value registers (ADCCV1H:ADCCV1L and ADCCV2H:ADCCV2L) determined by the value of ACFGT.The ACFE bit must be set for ACFGT to have any effect. The compare function modes are further detailed in Table 21-24 in Section 21.5.6. 0 Range function disabled. Only the compare value 1 register (ADCCV1H:ADCCV1L) is compared. 1 Range function enabled. Both compare value registers (ADCCV1H:ADCCV1L and ADCCV2H:ADCCV2L) are compared.
1:0 REFSEL [1:0]	**Voltage Reference Selection** - REFSEL bits select the voltage reference source used for conversions.Refer to Section 21.5.3 for more information on voltage reference selection. 00 Default voltage reference pin pair (External pins V_{REFH} and V_{REFL}). 01 Alternate reference pair (V_{ALTH} and V_{ALTL}). This pair may be additional external pins or internal sources depending on MCU configuration.Consult the module introduction for information on the Voltage Reference specific to this MCU. 10 Internal bandgap reference and associated ground reference (V_{BGH} and V_{BGL}). 11 Reserved - Selects default voltage reference (V_{REFH} and V_{REFL}) pin pair.

21.4.7 Status and Control Register 3 (ADCSC3)

The ADCSC3 register controls the calibration, continuous convert and hardware averaging functions of the ADC module.

	7	6	5	4	3	2	1	0
R	CAL	CALF	0	0	ADCO	AVGE	AVGS	
W								
Reset:	0	0	0	0	0	0	0	0

Figure 21-12. Status and Control Register 3 (ADCSC3)

Table 21-12. ADCSC3 Register Field Descriptions

Field	Description
7 CAL	**Calibration - CAL begins the calibration sequence when set. This bit stays set while the calibration is in progress and is cleared when the calibration sequence is complete. The CALF bit must be checked to determine the result of the calibration sequence. Once started, the calibration routine cannot be interrupted by writes to the ADC registers or the results will be invalid and the CALF bit will set. Setting the CAL bit will abort any current conversion.**
6 CALF	**Calibration Failed Flag -** CALF displays the result of the calibration sequence. The calibration sequence will fail if ADTRG = 1, any ADC register is written, or any stop mode is entered before the calibration sequence completes.The CALF bit is cleared by writing a 1 to this bit. 0 Calibration completed normally. 1 Calibration failed. ADC accuracy specifications are not guaranteed.
3 ADCO	**Continuous Conversion Enable** - ADCO enables continuous conversions.Refer to Section 21.5.5.1 for more information on initiating conversions. 0 One conversion or one set of conversions if the hardware average function is enabled (AVGE=1) after initiating a conversion. 1 Continuous conversions or sets of conversions if the hardware average function is enabled (AVGE=1) after initiating a conversion.
2 AVGE	**Hardware average enable - AVGE enables the hardware average function of the ADC.** 0 Hardware average function disabled 1 Hardware average function enabled
1:0 AVGS	**Hardware Average select** - AVGS determine how many ADC conversions will be averaged to create the ADC average result. 00 - 4 Samples averaged 01 - 8 Samples averaged 10 - 16 Samples averaged 11 - 32 Samples averaged

21.4.8 ADC Offset Correction Register (ADCOFSH:ADCOFSL)

The ADC Offset Correction Register (ADCOFSH:ADCOFSL) contains the user-selected or calibration-generated offset error correction value.

This register is a 2’s complement, left justified, 16b value formed by the concatenation of:

- ADCOFSH
- ADCOFSL.

The value in the offset correction registers (ADCOFSH:ADCOFSL) is subtracted from the conversion and the result is transferred into the result registers (ADCRHn:ADCRLn).

NOTE

If the result is above the maximum or below the minimum result value, it is forced to the appropriate limit for the current mode of operation.For additional information, please see Section 21.5.8.

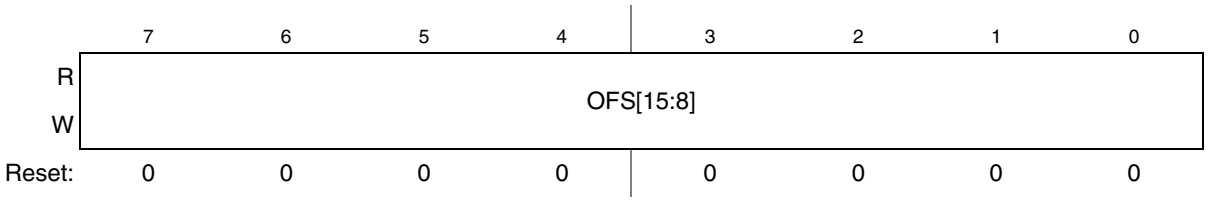


Figure 21-13. Offset Calibration High Register (ADCOFSH)

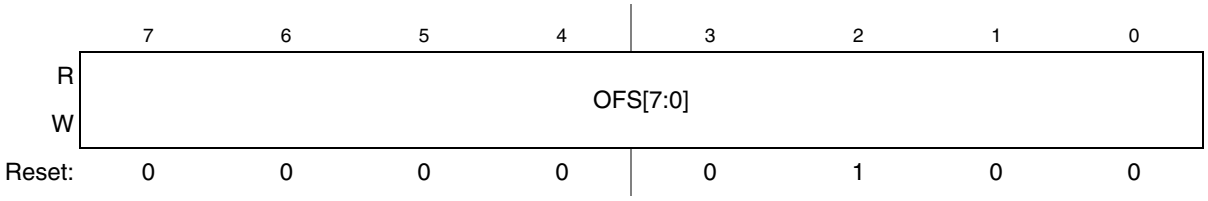


Figure 21-14. Offset Calibration Low Register (ADCOFSL)

21.4.9 ADC Plus-Side Gain Register (ADCPGH:ADCPGL)

The Plus-Side Gain Register (ADCPGH:ADCPGL) contains the gain error correction for the plus-side input in differential mode or the overall conversion in single-ended mode. ADCPGH:ADCPGL represent a 16 bit floating point number representation of the gain adjustment factor, with the decimal point fixed between PG15 and PG14. This register must be written by the user with the value described in the calibration procedure or the gain error specifications may not be met.

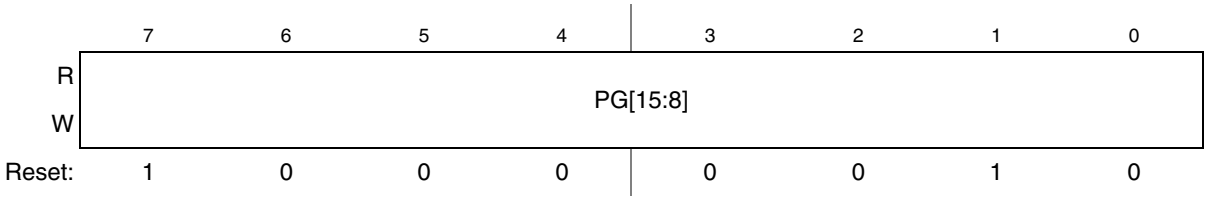


Figure 21-15. ADC Plus Gain High Register (ADCPGH)

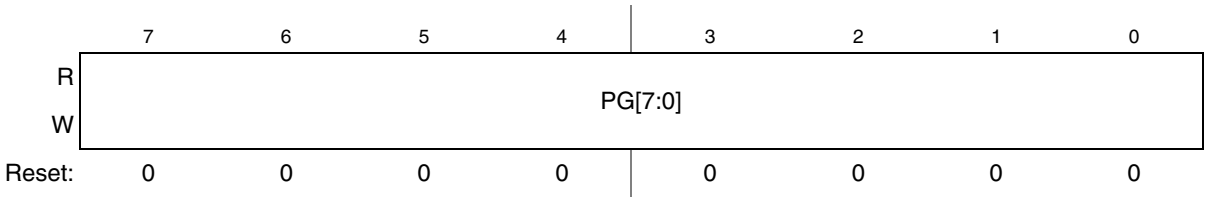


Figure 21-16. ADC Plus Gain Low Register (ADCPGL)

21.4.10 ADC Minus-Side Gain Register (ADCMGH:ADCMGL)

The Minus-Side Gain Register (ADCMGH:ADCMGL) contains the gain error correction for the minus-side input in differential mode. This register is ignored in single-ended mode. ADCMGH:ADCMGL represent a 16 bit floating point number representation of the gain adjustment

factor, with the decimal point fixed between MG15 and MG14. This register must be written by the user with the value described in the calibration procedure or the gain error specifications may not be met.

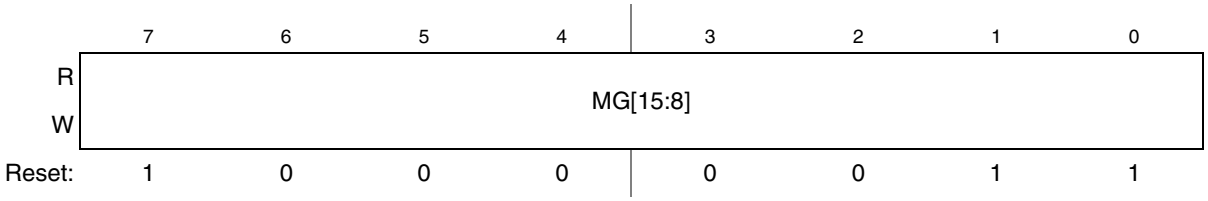


Figure 21-17. ADC Minus-Side Gain Register (ADCMGH)

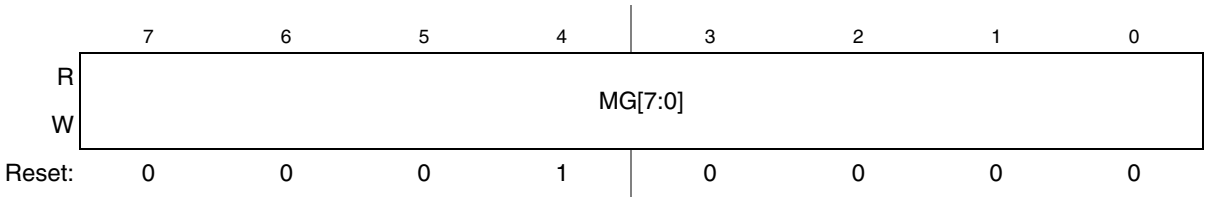


Figure 21-18. ADC Minus-Side Gain Register (ADCMGL)

21.4.11 ADC Plus-Side General Calibration Value Registers (ADCCLPx)

The Plus-Side General Calibration Value Registers (ADCCLPx) contain calibration information that is generated by the calibration function. These registers contain seven calibration values of varying widths: CLP0[5:0], CLP1[6:0], CLP2[7:0], CLP3[8:0], CLP4[9:0], CLPS[5:0], and CLPD[5:0]. ADCCLPx are automatically set once the self calibration sequence is done (CAL is cleared). If these registers are written by the user after calibration, the linearity error specifications may not be met.

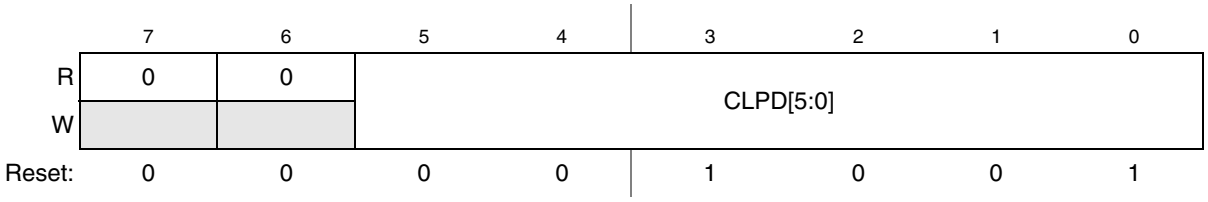


Figure 21-19. Plus-Side General Calibration Register (ADCCLPD)

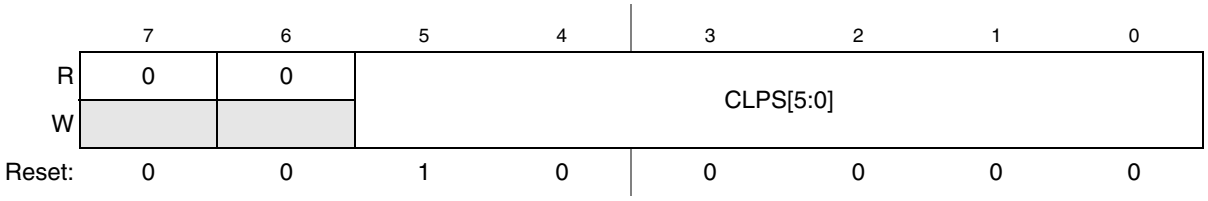


Figure 21-20. Plus-Side General Calibration Register (ADCCLPS)

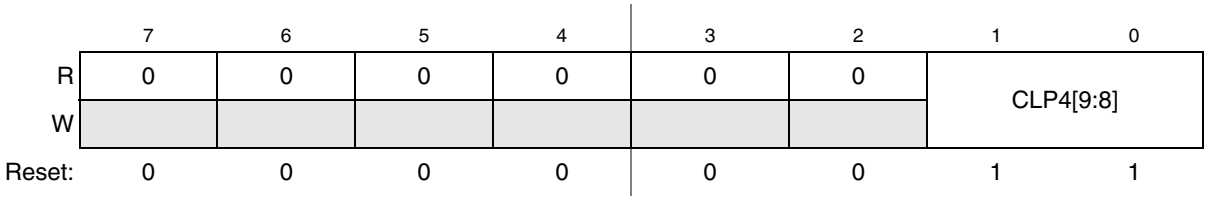


Figure 21-21. Plus-Side General Calibration Register (ADCCLP4H)

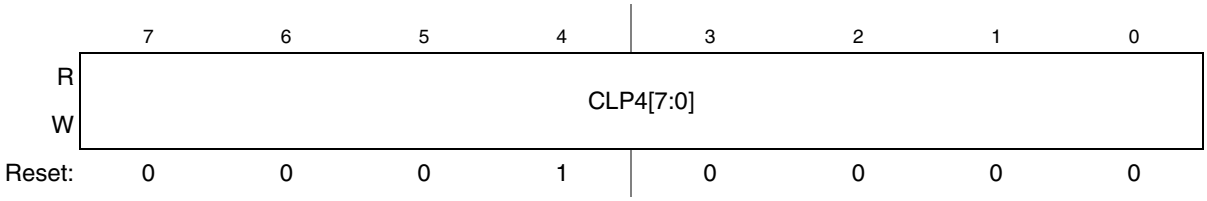


Figure 21-22. Plus-Side General Calibration Register (ADCCLP4L)

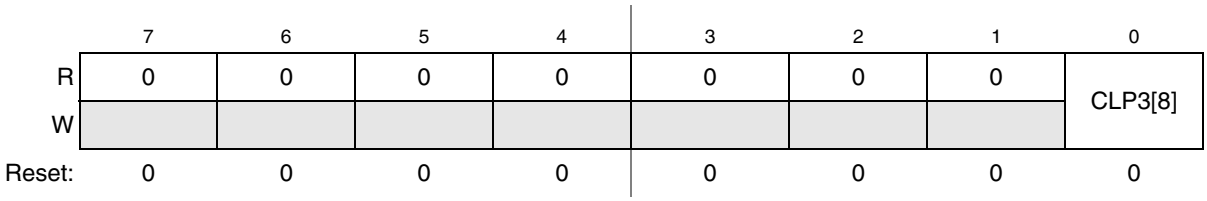


Figure 21-23. Plus-Side General Calibration Register (ADCCLP3H)

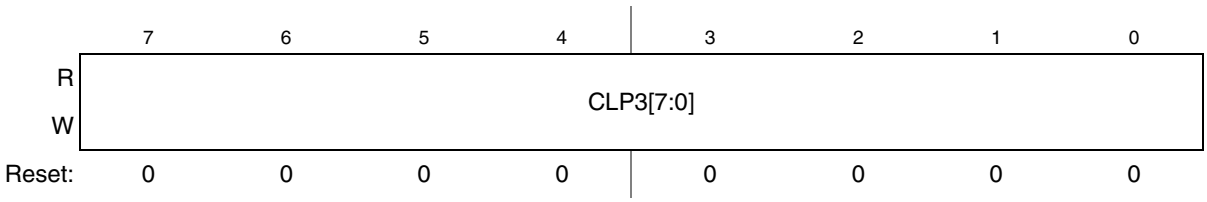


Figure 21-24. Plus-Side General Calibration Register (ADCCLP3L)

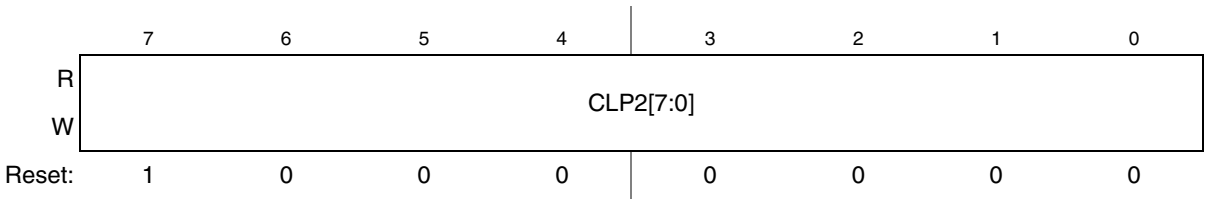


Figure 21-25. Plus-Side General Calibration Register (ADCCLP2)

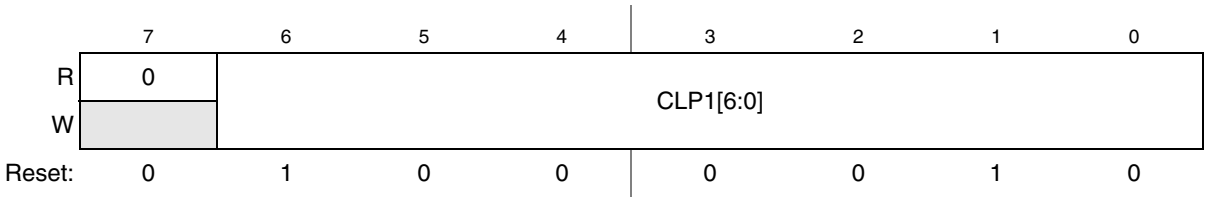


Figure 21-26. Plus-Side General Calibration Register (ADCCLP1)

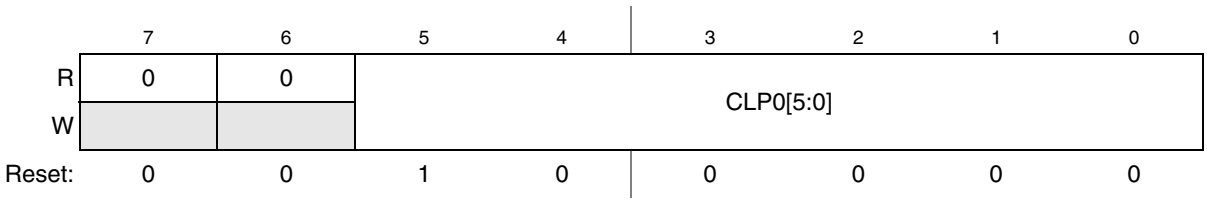


Figure 21-27. Plus-Side General Calibration Register (ADCCLP0)

21.4.12 ADC Minus-Side General Calibration Value Registers (ADCCLMx)

ADCCLMx contain calibration information that is generated by the calibration function. These registers contain seven calibration values of varying widths: CLM0[5:0], CLM1[6:0], CLM2[7:0], CLM3[8:0], CLM4[9:0], CLMS[5:0], and CLMD[5:0]. ADCCLMx are automatically set once the self calibration sequence is done (CAL is cleared). If these registers are written by the user after calibration, the linearity error specifications may not be met.

	7	6	5	4	3	2	1	0
R	0	0	CLMD[5:0]					
W								
Reset:	0	0	0	1	1	1	0	1

Figure 21-28. Minus-Side General Calibration Register (ADCCLMD)

	7	6	5	4	3	2	1	0
R	0	0	CLMS[5:0]					
W								
Reset:	0	0	1	1	0	0	0	1

Figure 21-29. Minus-Side General Calibration Register (ADCCLMS)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	CLM4[9:8]	
W								
Reset:	0	0	0	0	0	0	1	1

Figure 21-30. Minus-Side General Calibration Register (ADCCLM4H)

	7	6	5	4	3	2	1	0
R	CLM4[7:0]							
W								
Reset:	0	0	0	1	0	0	0	0

Figure 21-31. Minus-Side General Calibration Register (ADCCLM4L)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	CLM3[8]
W								
Reset:	0	0	0	0	0	0	0	1

Figure 21-32. Minus-Side General Calibration Register (ADCCLM3H)

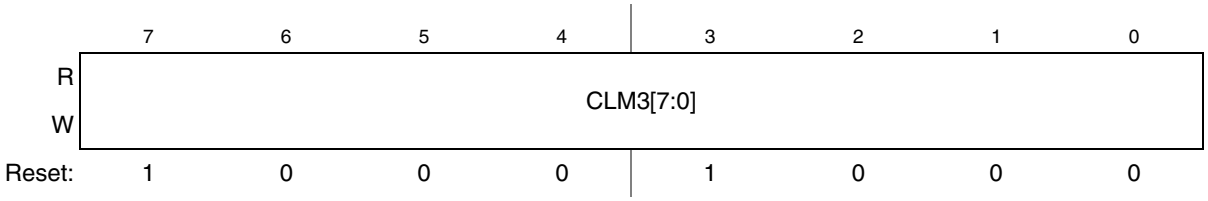


Figure 21-33. Minus-Side General Calibration Register (ADCCLM3L)

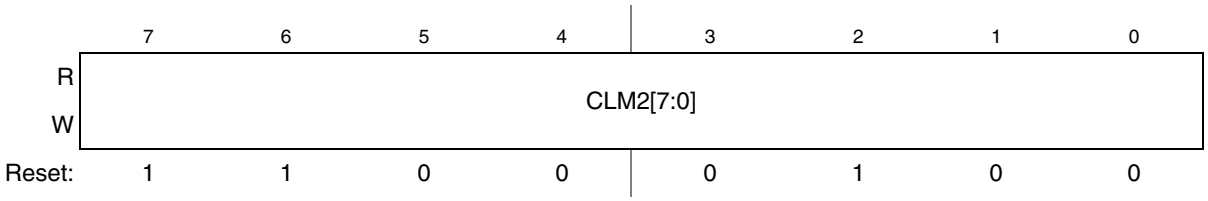


Figure 21-34. Minus-Side General Calibration Register (ADCCLM2)

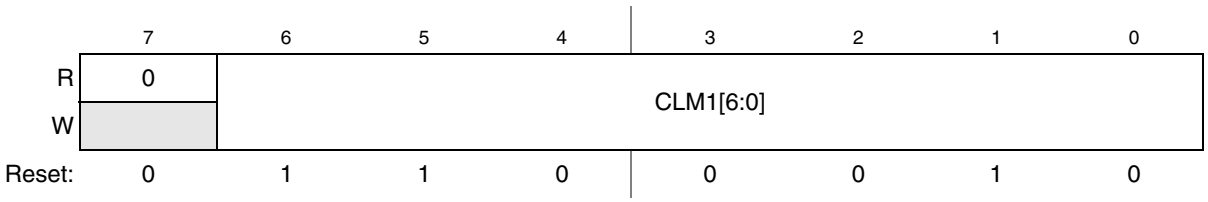


Figure 21-35. Minus-Side General Calibration Register (ADCCLM1)

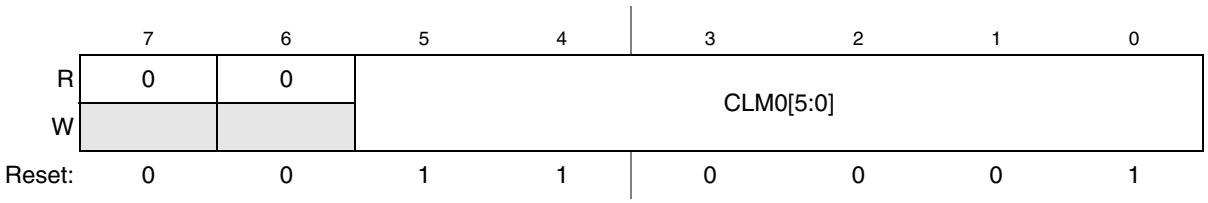


Figure 21-36. Minus-Side General Calibration Register (ADCCLM0)

21.4.13 Pin Control 1 Register (APCTL1)

The pin control registers disable the I/O port control of MCU pins used as analog inputs. APCTL1 is used to control the pins associated with channels 0–7 of the ADC module.

Figure 21-37. Pin Control 1 Register (APCTL1)

Table 21-13. APCTL1 Register Field Descriptions

Field	Description
7 ADPC7	**ADC Pin Control 7 -** ADPC7 controls the pin associated with channel AD7. 0 AD7 pin I/O control enabled 1 AD7 pin I/O control disabled
6 ADPC6	**ADC Pin Control 6** - ADPC6 controls the pin associated with channel AD6. 0 AD6 pin I/O control enabled 1 AD6 pin I/O control disabled
5 ADPC5	**ADC Pin Control 5**- ADPC5 controls the pin associated with channel AD5. 0 AD5 pin I/O control enabled 1 AD5 pin I/O control disabled
4 ADPC4	**ADC Pin Control 4** - ADPC4 controls the pin associated with channel AD4. 0 AD4 pin I/O control enabled 1 AD4 pin I/O control disabled
3 ADPC3	**ADC Pin Control 3** - ADPC3 controls the pin associated with channel AD3. 0 AD3 pin I/O control enabled 1 AD3 pin I/O control disabled
2 ADPC2	**ADC Pin Control 2** - ADPC2 controls the pin associated with channel AD2. 0 AD2 pin I/O control enabled 1 AD2 pin I/O control disabled
1 ADPC1	**ADC Pin Control 1** - ADPC1 controls the pin associated with channel AD1. 0 AD1 pin I/O control enabled 1 AD1 pin I/O control disabled
0 ADPC0	**ADC Pin Control 0** - ADPC0 controls the pin associated with channel AD0. 0 AD0 pin I/O control enabled 1 AD0 pin I/O control disabled

21.4.14 Pin Control 2 Register (APCTL2)

APCTL2 controls channels 8–15 of the ADC module.

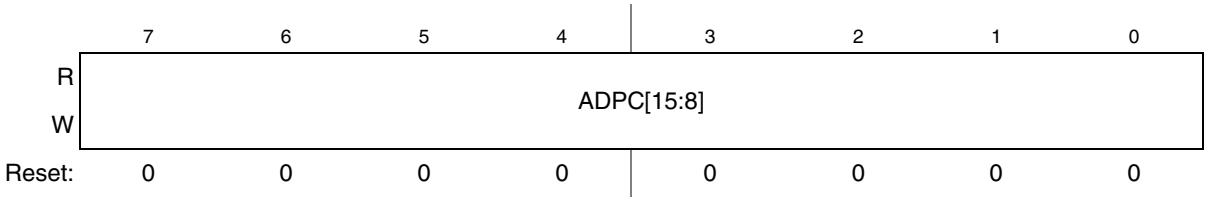


Figure 21-38. Pin Control 2 Register (APCTL2)

Table 21-14. APCTL2 Register Field Descriptions

Field	Description
7 ADPC15	**ADC Pin Control 15 -** ADPC15 controls the pin associated with channel AD15. 0 AD15 pin I/O control enabled 1 AD15 pin I/O control disabled
6 ADPC14	**ADC Pin Control 14** - ADPC14 controls the pin associated with channel AD14. 0 AD14 pin I/O control enabled 1 AD14 pin I/O control disabled
5 ADPC13	**ADC Pin Control 13** - ADPC13 controls the pin associated with channel AD13. 0 AD13 pin I/O control enabled 1 AD13 pin I/O control disabled
4 ADPC12	**ADC Pin Control 12** - ADPC12 controls the pin associated with channel AD12. 0 AD12 pin I/O control enabled 1 AD12 pin I/O control disabled
3 ADPC11	**ADC Pin Control 11** - ADPC11 controls the pin associated with channel AD11. 0 AD11 pin I/O control enabled 1 AD11 pin I/O control disabled
2 ADPC10	**ADC Pin Control 10** - ADPC10 controls the pin associated with channel AD10. 0 AD10 pin I/O control enabled 1 AD10 pin I/O control disabled
1 ADPC9	**ADC Pin Control 9** - ADPC9 controls the pin associated with channel AD9. 0 AD9 pin I/O control enabled 1 AD9 pin I/O control disabled
0 ADPC8	**ADC Pin Control 8** - ADPC8 controls the pin associated with channel AD8. 0 AD8 pin I/O control enabled 1 AD8 pin I/O control disabled

21.4.15 Pin Control 3 Register (APCTL3)

APCTL3 controls channels 23–16 of the ADC module.

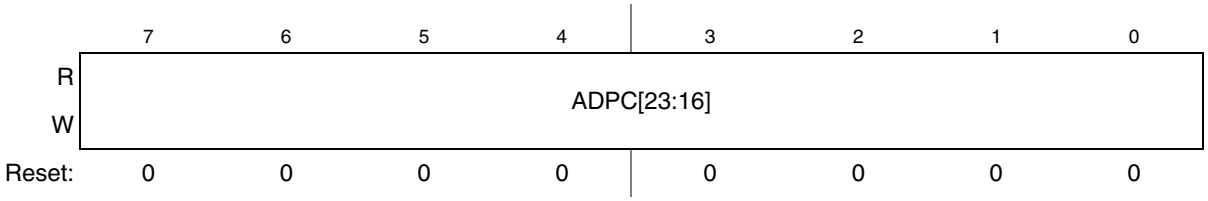


Figure 21-39. Pin Control 3 Register (APCTL3)

Table 21-15. APCTL3 Register Field Descriptions

Field	Description
7 ADPC23	**ADC Pin Control 23** - ADPC23 controls the pin associated with channel AD23. 0 AD23 pin I/O control enabled 1 AD23 pin I/O control disabled
6 ADPC22	**ADC Pin Control 22** - ADPC22 controls the pin associated with channel AD22. 0 AD22 pin I/O control enabled 1 AD22 pin I/O control disabled
5 ADPC21	**ADC Pin Control 21** - ADPC21 controls the pin associated with channel AD21. 0 AD21 pin I/O control enabled 1 AD21 pin I/O control disabled
4 ADPC20	**ADC Pin Control 20** - ADPC20 controls the pin associated with channel AD20. 0 AD20 pin I/O control enabled 1 AD20 pin I/O control disabled
3 ADPC19	**ADC Pin Control 19** - ADPC19 controls the pin associated with channel AD19. 0 AD19 pin I/O control enabled 1 AD19 pin I/O control disabled
2 ADPC18	**ADC Pin Control 18** - ADPC18 controls the pin associated with channel AD18. 0 AD18 pin I/O control enabled 1 AD18 pin I/O control disabled
1 ADPC17	**ADC Pin Control 17** - ADPC17 controls the pin associated with channel AD17. 0 AD17 pin I/O control enabled 1 AD17 pin I/O control disabled
0 ADPC16	**ADC Pin Control 16** - ADPC16 controls the pin associated with channel AD16. 0 AD16 pin I/O control enabled 1 AD16 pin I/O control disabled

21.4.16 Pin Control 4 Register (APCTL4)

APCTL4 controls channels DAD0-DAD3 of the ADC module. When DIFFn=1, channels DAD0-DAD3 use the input pin pairs DADPx and DADMx to form a differential conversion. When DIFFn=0, channels DAD0-DAD3 use only the input pins DADPx and the DADMx input pins are ingored.

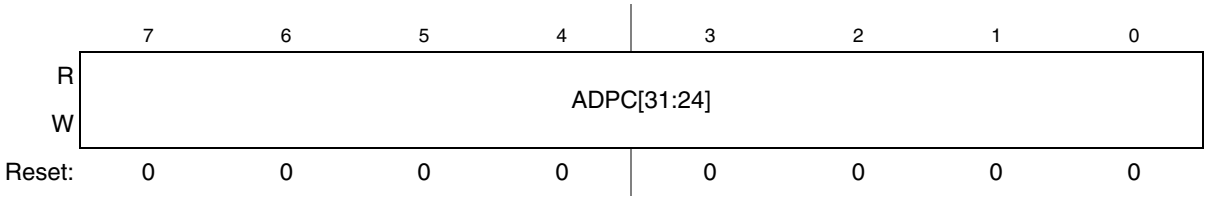


Figure 21-40. Pin Control 4 Register (APCTL4)

Table 21-16. APCTL4 Register Field Descriptions

Field	Description
7 ADPC31	**ADC Pin Control 31** - ADPC31 controls the pin associated with channel AD31. 0 AD31 pin I/O control enabled 1 AD31 pin I/O control disabled
6 ADPC30	**ADC Pin Control 30** - ADPC30 controls the pin associated with channel AD30. 0 AD30 pin I/O control enabled 1 AD30 pin I/O control disabled
5 ADPC29	**ADC Pin Control 29** - ADPC29 controls the pin associated with channel AD29. 0 AD29 pin I/O control enabled 1 AD29 pin I/O control disabled
4 ADPC28	**ADC Pin Control 28** - ADPC28 controls the pin associated with channel AD28. 0 AD28 pin I/O control enabled 1 AD28 pin I/O control disabled
3 ADPC27	**ADC Pin Control 27** - ADPC27 controls the pin associated with channel AD27. 0 AD27 pin I/O control enabled 1 AD27 pin I/O control disabled
2 ADPC26	**ADC Pin Control 26** - ADPC26 controls the pin associated with channel AD26. 0 AD26 pin I/O control enabled 1 AD26 pin I/O control disabled
1 ADPC25	**ADC Pin Control 25** - ADPC25 controls the pin associated with channel AD25. 0 AD25 pin I/O control enabled 1 AD25 pin I/O control disabled
0 ADPC24	**ADC Pin Control 24** - ADPC24 controls the pin associated with channel AD24. 0 AD24 pin I/O control enabled 1 AD24 pin I/O control disabled

21.5 Functional Description

The ADC module is disabled during reset, stop2 or when the ADCHn bits are all high. The module is idle when a conversion has completed and another conversion has not been initiated. When idle and the asynchronous clock output enable is disabled (ADACKEN=0), the module is in its lowest power state. The ADC can perform an analog-to-digital conversion on any of the software selectable channels. All modes perform conversion by a successive approximation algorithm.

To meet accuracy specifications, the ADC module must be calibrated using the on chip calibration function. Calibration is recommended to be done after any reset. See Section 21.5.7 for details on how to perform calibration.

When the conversion is completed, the result is placed in the data registers (ADCRHn and ADCRLn). The conversion complete flag (COCOn) is then set and an interrupt is generated if the respective conversion complete interrupt has been enabled (AIENn=1).

The ADC module has the capability of automatically comparing the result of a conversion with the contents of the compare value registers. The compare function is enabled by setting the ACFE bit and operates with any of the conversion modes and configurations.

The ADC module has the capability of automatically averaging the result of multiple conversions. The hardware average function is enabled by setting the AVGE bit and operates with any of the conversion modes and configurations.

21.5.1 Clock Select and Divide Control

One of four clock sources can be selected as the clock source for the ADC module. This clock source is then divided by a configurable value to generate the input clock to the converter (ADCK). The clock is selected from one of the following sources by means of the ADICLK bits.

- The bus clock, which is equal to the frequency at which software is executed. This is the default selection following reset.
- The bus clock divided by two. For higher bus clock rates, this allows a maximum divide by 16 of the bus clock with using the ADIV bits.
- ALTCLK, as defined for this MCU (See module section introduction).
- The asynchronous clock (ADACK). This clock is generated from a clock source within the ADC module. Conversions are possible using ADACK as the input clock source while the MCU is in stop3 mode. Refer to Section 21.5.5.4 for more information.

Whichever clock is selected, its frequency must fall within the specified frequency range for ADCK. If the available clocks are too slow, the ADC may not perform according to specifications. If the available clocks are too fast, the clock must be divided to the appropriate frequency. This divider is specified by the ADIV bits and can be divide-by 1, 2, 4, or 8.

21.5.2 Input Select and Pin Control

The pin control registers (APCTL1, APCTL2, APCTL3, and APCTL4) disable the I/O port control of the pins used as analog inputs.When a pin control register bit is set, the following conditions are forced for the associated MCU pin:

- The output buffer is forced to its high impedance state.
- The input buffer is disabled. A read of the I/O port returns a zero for any pin with its input buffer disabled.
- The pullup is disabled.

21.5.3 Voltage Reference Selection

The ADC can be configured to accept one of three voltage reference pairs as the reference voltage (V_{REFSH} and V_{REFSL}) used for conversions. Each pair contains a positive reference which must be between the minimum Ref Voltage High (defined in Appendix A) and V_{DDAD}, and a ground reference which must be at the same potential as V_{SSAD}. The three pairs are external (V_{REFH} and V_{REFL}), alternate (V_{ALTH} and V_{ALTL}) and the internal bandgap (V_{BGH} and V_{BGL}). These voltage references are selected using the REFSEL bits in the ADCSC2 register.The alternate (V_{ALTH} and V_{ALTL}) voltage reference pair may select additional external pins or internal sources depending on MCU configuration.Consult the module introduction for information on the Voltage References specific to this MCU.

21.5.4 Hardware Trigger and Channel Selects

The ADC module has a selectable asynchronous hardware conversion trigger, ADHWT, that is enabled when the ADTRG bit is set and a hardware trigger select event (ADHWTSn) has occurred. This source is not available on all MCUs. Consult the module introduction for information on the ADHWT source and the ADHWTSn configurations specific to this MCU.

When the ADHWT source is available and hardware trigger is enabled (ADTRG=1), a conversion is initiated on the rising edge of the ADHWT after a hardware trigger select event (ADHWTSn) has occurred. If a conversion is in progress when a rising edge of a trigger occurs, the rising edge is ignored. In continuous convert configuration, only the initial rising edge to launch continuous conversions is observed and until conversion gets aborted the ADC will continue to do conversions on the same ADC Status and Control register that initiated the conversion. The hardware trigger function operates in conjunction with any of the conversion modes and configurations.

The hardware trigger select event (ADHWTSn) must be set prior to and during the receipt of the ADHWT signal. If these conditions are not met the converter may ignore the trigger or use the incorrect configuration. If a hardware trigger select event gets asserted during a conversion, it must stay asserted until end of current conversion and remain set until the receipt of the ADHWT signal to trigger a new conversion. The channel and status fields selected for the conversion will depend on the active trigger select signal (ADHWTSA active selects ADCSC1A; ADHWTSn active selects ADCSC1n).

NOTE

Asserting more than one hardware trigger select signal (ADHWTSn) at the same time will result in unknown results. To avoid this, only select one hardware trigger select signal (ADHWTSn) prior to the next intended conversion.

When the conversion is completed, the result is placed in the data registers associated with the ADHWTSn received (ADHWTSA active selects ADCRHA:ADCRLA; ADHWTSn active selects ADCRHn:ADCRLn). The conversion complete flag associated with the ADHWTSn received (COCOn) is then set and an interrupt is generated if the respective conversion complete interrupt has been enabled (AIENn=1).

21.5.5 Conversion Control

Conversions can be performed as determined by the MODE bits and the DIFFn bit as shown in Table 21-7.

Conversions can be initiated by a software or hardware trigger. In addition, the ADC module can be configured for low power operation, long sample time, continuous conversion, hardware average and automatic compare of the conversion result to a software determined compare value.

21.5.5.1 Initiating Conversions

A conversion is initiated:

- Following a write to ADCSC1A (with ADCHA bits not all 1’s) if software triggered operation is selected (ADTRG=0).

- Following a hardware trigger (ADHWT) event if hardware triggered operation is selected (ADTRG=1) and a hardware trigger select event (ADHWTSn) has occurred. The channel and status fields selected will depend on the active trigger select signal (ADHWTSA active selects ADCSC1A; ADHWTSn active selects ADCSC1n; if neither is active the off condition is selected).

NOTE

Selecting more than one hardware trigger select signal (ADHWTSn) prior to a conversion completion will result in unknown results. To avoid this, only select one hardware trigger select signal (ADHWTSn) prior to a conversion completion.

- Following the transfer of the result to the data registers when continuous conversion is enabled (ADCO=1).

If continuous conversions are enabled, a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation (ADTRG=0), continuous conversions begin after ADCSC1A is written and continue until aborted. In hardware triggered operation(ADTRG=1 and one ADHWTSn event has occurred), continuous conversions begin after a hardware trigger event and continue until aborted.

If hardware averaging is enabled, a new conversion is automatically initiated after the completion of the current conversion until the correct number of conversions is completed. In software triggered operation, conversions begin after ADCSC1A is written. In hardware triggered operation, conversions begin after a hardware trigger. If continuous conversions are also enabled, a new set of conversions to be averaged are initiated following the last of the selected number of conversions.

21.5.5.2 Completing Conversions

A conversion is completed when the result of the conversion is transferred into the data result registers, ADCRHn and ADCRLn. If the compare functions are disabled, this is indicated by the setting of COCOn. If hardware averaging is enabled, COCOn sets only if the last of the selected number of conversions is complete. If the compare function is enabled, COCOn sets and conversion result data is transferred only if the compare condition is true. If both hardware averaging and compare functions are enabled then COCOn sets only if the last of the selected number of conversions is complete and the compare condition is true. An interrupt is generated if AIENn is high at the time that COCOn is set.In all modes except 8-bit single-ended conversions, a blocking mechanism prevents a new result from overwriting previous data in ADCRHn and ADCRLn if the previous data is in the process of being read (the ADCRHn register has been read but the ADCRLn register has not). When blocking is active, the conversion result data transfer is blocked, COCOn is not set, and the new result is lost. In the case of single conversions with the compare function enabled and the compare condition false, blocking has no effect and ADC operation is terminated. In all other cases of operation, when a conversion result data transfer is blocked, another conversion is initiated regardless of the state of ADCO (single or continuous conversions enabled).

NOTE

If continuous conversions are enabled, the blocking mechanism could result in the loss of data occurring at specific timepoints. To avoid this issue, the data must be read in fewer cycles than an ADC conversion time, accounting for interrupt or software polling loop latency.

If single conversions are enabled, the blocking mechanism could result in several discarded conversions and excess power consumption. To avoid this issue, the data registers must not be read after initiating a single conversion until the conversion completes.

21.5.5.3 Aborting Conversions

Any conversion in progress is aborted when:

- Writing ADCSC1A while ADCSC1A is actively controlling a conversion aborts the current conversion. In software trigger mode (ADTRG=0), a write to ADCSC1A initiates a new conversion (if the ADCHA bits are equal to a value other than all 1s). Writing any of the ADCSC1(B-n) registers while that specific ADCSC1(B-n) register is actively controlling a conversion aborts the current conversion.The ADCSC1(B-n) registers are not used for software trigger operation and therefore writes to the ADCSC1(B-n) registers do not initiate a new conversion.
- A write to any ADC register besides the ADCSC1A:ADCSC1n registers occurs. This indicates a mode of operation change has occurred and the current conversion is therefore invalid.
- The MCU is reset or enters stop2 mode.
- The MCU enters stop3 mode with ADACK not enabled.

When a conversion is aborted, the contents of the data registers, ADCRHn and ADCRLn, are not altered. The data registers continue to be the values transferred after the completion of the last successful conversion. If the conversion was aborted by a reset or stop2, ADCRHA:ADCRLA and ADCRHn:ADCRLn return to their reset states.

21.5.5.4 Power Control

The ADC module remains in its idle state until a conversion is initiated. If ADACK is selected as the conversion clock source but the asynchronous clock output is disabled (ADACKEN=0), the ADACK clock generator will also remain in its idle state (disabled) until a conversion is initiated. If the asynchronous clock output is enabled (ADACKEN=1), it will remain active regardless of the state of the ADC or the MCU power mode.

Power consumption when the ADC is active can be reduced by setting ADLPC. This results in a lower maximum value for f_{ADCK} (see the electrical specifications).

21.5.5.5 Sample Time and Total Conversion Time

The total conversion time depends upon: the sample time (as determined by ADLSMP and ADLSTS bits), the MCU bus frequency, the conversion mode (as determined by MODE and DIFFn bits), the high speed configuration (ADHSC bit), and the frequency of the conversion clock (f_{ADCK}).

The ADHSC bit is used to configure a higher clock input frequency. This will allow faster overall conversion times. In order to meet internal A/D converter timing requirements the ADHSC bit adds additional ADCK cycles. Conversions with ADHSC = 1 take four more ADCK cycles. ADHSC should be used when the ADCLK exceeds the limit for ADHSC = 0.

After the module becomes active, sampling of the input begins. ADLSMP and ADLSTS select between sample times based on the conversion mode that is selected. When sampling is complete, the converter is isolated from the input channel and a successive approximation algorithm is performed to determine the digital value of the analog signal. The result of the conversion is transferred to ADCRHn and ADCRLn upon completion of the conversion algorithm.

If the bus frequency is less than the f_{ADCK} frequency, precise sample time for continuous conversions cannot be guaranteed when short sample is enabled (ADLSMP=0). The maximum total conversion time is determined by the clock source chosen and the divide ratio selected. The clock source is selectable by the ADICLK bits, and the divide ratio is specified by the ADIV bits. The maximum total conversion time for all configurations is summarized in the equation below. Refer to Table 21-17 through Table 21-21 for the variables referenced in the equation.

Conversion Time Equation ***Eqn. 21-1***

$$ConversionTime = SFCAdder + AverageNum \times (BCT + LSTAdder + HSCAdder)$$

Table 21-17. Single or First Continuous Time Adder (SFCAdder)

ADLSMP	ADACKEN	ADICLK	Single or First Continuous Time Adder (SFCAdder)
1	x	0x, 10	3 ADCK cycles + 5 bus clock cycles
1	1	11	3 ADCK cycles + 5 bus clock cycles[1]
1	0	11	5μs + 3 ADCK cycles + 5 bus clock cycles
0	x	0x, 10	5 ADCK cycles + 5 bus clock cycles
0	1	11	5 ADCK cycles + 5 bus clock cycles[1]
0	0	11	5μs + 5 ADCK cycles + 5 bus clock cycles

[1] ADACKEN must be 1 for at least 5us prior to the conversion is initiated to achieve this time

Table 21-18. Average Number Factor (AverageNum)

AVGE	AVGS[1:0]	Average Number Factor (AverageNum)
0	xx	1
1	00	4
1	01	8
1	10	16
1	11	32

Table 21-19. Base Conversion Time (BCT)

Mode	Base Conversion Time (BCT)
8b se	17 ADCK cycles
9b diff	27 ADCK cycles
10b s.e.	20 ADCK cycles
11b diff	30 ADCK cycles
12b s.e.	20 ADCK cycles
13b diff	30 ADCK cycles
16b s.e.	25 ADCK cycles
16b diff	34 ADCK cycles

Table 21-20. Long Sample Time Adder (LSTAdder)

ADLSMP	ADLSTS	Long Sample Time Adder (LSTAdder)
0	xx	0 ADCK cycles
1	00	20 ADCK cycles
1	01	12 ADCK cycles
1	10	6 ADCK cycles
1	11	2 ADCK cycles

Table 21-21. High-Speed Conversion Time Adder (HSCAdder)

ADHSC	High Speed Conversion Time Adder (HSCAdder)
0	0 ADCK cycles
1	4 ADCK cycles

NOTE

The ADCK frequency must be between f_{ADCK} minimum and f_{ADCK} maximum to meet ADC specifications.

21.5.5.6 Conversion Time Examples

The following examples use Equation 21-2 and the information provided in Table 21-17 through Table 21-21.

21.5.5.6.1 Typical conversion time configuration

A typical configuration for ADC conversion is: 10-bit mode, with the bus clock selected as the input clock source, the input clock divide-by-1 ratio selected, and a bus frequency of 8 MHz, long sample time disabled and high speed conversion disabled. The conversion time for a single conversion is calculated by using Equation 21-2 and the information provided in Table 21-17 through Table 21-21. The table below lists the variables of Equation 21-2.

Table 21-22. Typical Conversion Time

Variable	Time
SFCAdder	5 ADCK cycles + 5 bus clock cycles
AverageNum	1
BCT	20 ADCK cycles
LSTAdder	0
HSCAdder	0

The resulting conversion time is generated using the parameters listed in Table 21-24. So, for Bus clock equal to 8 MHz and ADCK equal to 8 MHz, the resulting conversion time is 3.75 us.

21.5.5.6.2 Long conversion time configuration

A configuration for long ADC conversion is: 16-bit differential mode, with the bus clock selected as the input clock source, the input clock divide-by-8 ratio selected, and a bus frequency of 8 MHz, long sample time enabled and configured for longest adder and high speed conversion disabled. Average enabled for 32 conversions. The conversion time for this conversion is calculated by using Equation 21-2 and the information provided in Table 21-17 through Table 21-21. The table below list the variables of Equation 21-2.

Table 21-23. Typical Conversion Time

Variable	Time
SFCAdder	3 ADCK cycles + 5 bus clock cycles
AverageNum	32
BCT	34 ADCK cycles
LSTAdder	20 ADCK cycles
HSCAdder	0

The resulting conversion time is generated using the parameters listed in Table 21-24. So, for Bus clock equal to 8 MHz and ADCK equal to 1 MHz the resulting conversion time is 57.625 μs (AverageNum). This results in a total conversion time of 1.844 ms.

21.5.5.7 Hardware Average Function

The hardware average function can be enabled (AVGE=1) to perform a hardware average of multiple conversions. The number of conversions is determined by the AVGS[1:0] bits, which select 4, 8, 16 or 32 conversions to be averaged. While the hardware average function is in progress, the ADACT bit is set.

After the selected input is sampled and converted, the result is placed in an accumulator from which an average is calculated once the selected number of conversions has been completed. When hardware averaging is selected the completion of a single conversion will not set the COCOn bit.

If the compare function is either disabled or evaluates true, after the selected number of conversions are completed, the average conversion result is transferred into the data result registers, ADCRHn and ADCRLn, and the COCOn bit is set. An ADC interrupt is generated upon the setting of COCOn if the respective ADC interrupt is enabled (AIENn=1).

NOTE

The hardware average function can perform conversions on a channel while the MCU is in wait or stop3 mode. The ADC interrupt wakes the MCU when the hardware average is complete if AIENn was set.

21.5.6 Automatic Compare Function

The compare function can be configured to check if the result is less than or greater-than-or-equal-to a single compare value, or if the result falls within or outside a range determined by two compare values. The compare mode is determined by ACFGT, ACREN and the values in the compare value registers(ADCCV1H:ADCCV1L and ADCCV2H:ADCCV2L). After the input is sampled and converted,

the compare values (ADCCV1H:ADCCV1L and ADCCV2H:ADCCV2L) are used as described in Table 21-24.There are six compare modes as shown in Table 21-24.

Table 21-24. Compare Modes

ACFGT	ACREN	ADCCV1 relative to ADCCV2	Function	Compare Mode Description
0	0	-	Less than threshold	Compare true if the result is less than the ADCCV1 registers.
1	0	-	Greater than or equal to threshold	Compare true if the result is greater than or equal to ADCCV1 registers.
0	1	Less than or equal	Outside range, not inclusive	Compare true if the result is less than ADCCV1 **Or** the result is Greater than ADCCV2
0	1	Greater than	Inside range, not inclusive	Compare true if the result is less than ADCCV1 **And** the result is greater than ADCCV2
1	1	Less Than or equal	Inside range, inclusive	Compare true if the result is greater than or equal to ADCCV1 **And** the result is less than or equal to ADCCV2
1	1	Greater than	Outside range, inclusive	Compare true if the result is greater than or equal to ADCCV1 **Or** the result is less than or equal to ADCCV2

With the ADC range enable bit set, ADCREN =1, if compare value register 1(ADCCV1 value) is less than or equal to the compare value register 2 (ADCCV2 value), setting ACFGT will select a trigger-if-inside-compare-range, inclusive-of-endpoints function. Clearing ACFGT will select a trigger-if-outside-compare-range, not-inclusive-of-endpoints function.

If ADCCV1 is greater than the ADCCV2, setting ACFGT will select a trigger-if-outside-compare-range, inclusive-of-endpoints function. Clearing ACFGT will select a trigger-if-inside-compare-range, not-inclusive-of-endpoints function.

If the condition selected evaluates true, COCOn is set.

Upon completion of a conversion while the compare function is enabled, if the compare condition is not true, COCOn is not set and the conversion result data will not be transferred to the result register. If the hardware averaging function is enabled, the compare function compares the averaged result to the compare values. The same compare function definitions apply. An ADC interrupt is generated upon the setting of COCOn if the respective ADC interrupt is enabled (AIENn=1).

NOTE

The compare function can monitor the voltage on a channel while the MCU is in wait or stop3 mode. The ADC interrupt wakes the MCU when the compare condition is met.

21.5.7 Calibration Function

The ADC contains a self-calibration function that is required to achieve the specified accuracy. Calibration must be run or valid calibration values written after any reset and before a conversion is initiated. The calibration function sets the offset calibration value and the plus-side and minus-side calibration values.

The offset calibration value is automatically stored in the ADC Offset Correction Registers (ADCOFSH and ADCOFSL) and the plus-side and minus-side calibration values are automatically stored in the ADC Plus-Side and Minus-Side Calibration registers (CLPD, CLPS, CLP4, CLP3, CLP2, CLP1, CLP0 and CLMD, CLMS, CLP4, CLM3, CLM2, CLM1, CLM0). The user must configure the ADC correctly prior to calibration, and must generate the plus-side and minus-side gain calibration results and store them in the ADC GAIN registers (ADCPGH and ADCPGL) after the calibration function completes.

Prior to calibration, the user must configure the ADC's clock frequency to be between 2 MHz and 4 MHz, set to high speed mode (ADLPC = 0, ADHSC = 1), and set to maximum averaging (AVGE = 1, AVGS = 11). It is recommended that calibration be run at $V_{DDA} = V_{REFH} \geq 3V$. The input channel, conversion mode continuous function, compare function, resolution mode, and differential/single-ended mode are all ignored during the calibration function.

To initiate calibration, the user sets the CAL bit and the calibration will automatically begin if the ADTRG bit = 0. If ADTRG = 1, the CAL bit will not get set and the calibration fail flag (CALF) will be set. While calibration is active, no ADC register can be written and no stop mode may be entered or the calibration routine will be aborted causing the CAL bit to clear and the CALF bit to set. At the end of a calibration sequence the COCO bit of the ADSC1A register will be set. The AIEN1 bit can be used to allow an interrupt to occur at the end of a calibration sequence. If at the end of calibration routine the CALF bit is not set, the automatic calibration routine completed successfully.

To complete calibration, the user must generate the gain calibration values using the following procedure:

- Initialize (clear) a 16b variable in RAM.
- Add the following plus-side calibration results CLP0, CLP1, CLP2, CLP3, CLP4, and CLPS to the variable.
- Divide the variable by two.
- Set the MSB of the variable.
- The previous two steps can be achieved by setting the carry bit, rotating-right through the carry bit on the high byte and again on the low byte.
- Store the value in the plus-side gain calibration registers ADCPGH and ADCPGL.
- Repeat procedure for the minus-side gain calibration value.

When complete the user may reconfigure and use the ADC as desired. A second calibration may also be performed if desired by clearing and again setting the CAL bit.

Overall the calibration routine may take as many as 14000 ADCK cycles and 100 bus cycles, depending on the results and the clock source chosen. For an 8 MHz clock source this is about 1.7 msec. To reduce this latency, the calibration values (offset, plus- and minus-side gain, and plus- and minus-side calibration values) may be stored in flash after an initial calibration and recovered prior to the first ADC conversion. This should reduce the calibration latency to 20 register store operations on all subsequent power, reset, or stop2 mode recoveries.

21.5.8 User Defined Offset Function

The ADC Offset Correction Register (ADCOFSH:ADCOFSL) contains the user selected or calibration generated offset error correction value. This register is a 2’s complement, left justified, 16b value formed

by the concatenation of ADCOFSH and ADCOFSL. The value in the offset correction registers (ADCOFSH:ADCOFSL) is subtracted from the conversion and the result is transferred into the result registers (ADCRHn:ADCRLn). If the result is above the maximum or below the minimum result value, it is forced to the appropriate limit for the current mode of operation.For additional information please see Section 21.5.8

The formatting of the ADC Offset Correction Register is different from the Data Result Registers (ADCRHn:ADCRLn) to preserve the resolution of the calibration value regardless of the conversion mode selected. Lower order bits are ignored in lower resolution modes. For example, in 8b single-ended mode, the bits OFS[14:7] are subtracted from D[7:0]; bit OFS[15] indicates the sign (negative numbers are effectively added to the result) and bits OFS[6:0] are ignored. The same bits are used in 9b differential mode since bit OFS[15] indicates the sign bit, which maps to bit D[8]. For 16b differential mode, all bits OFS[15:0] are directly subtracted from the conversion result data D[15:0]. Finally, in 16b single-ended mode, there is no bit in the Offset Correction Register corresponding to the least significant result bit D[0], so odd values (-1 or +1, etc.) cannot be subtracted from the result.ADCOFSH is automatically set according to calibration requirements once the self calibration sequence is done (CAL is cleared). Write ADCOFSH:ADCOFSL to override the calibration result if desired. If the Offset Correction Register is written to a value that is different from the calibration value, the ADC error specifications may not be met. It is recommended that the value generated by the calibration function be stored in memory before overwriting with a specified value.

NOTE

There is an effective limit to the values of Offset that can be set. If the magnitude of the offset is too great, the results of the conversions cap off at the limits.

Use the offset calibration function to remove application offsets or DC bias values. The Offset Correction Registers ADCOFSH and ADCOFSL may be written with a number in 2’s complement format and this offset will be subtracted from the result (or hardware averaged value). To add an offset, store the negative offset in 2’s complement format and the effect will be an addition. An offset correction that results in an out-of-range value will be forced to the minimum or maximum value (the minimum value for single-ended conversions is 0x0000; for a differential conversion 0x8000).

To preserve accuracy, the calibrated offset value initially stored in the ADCOFS registers must be added to the user defined offset. For applications which may change the offset repeatedly during operation, it is recommended to store the initial offset calibration value in flash so that it can be recovered and added to any user offset adjustment value -nd the sum stored in the ADCOFS registers.

21.5.9 Temperature Sensor

The ADC module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs. Equation 21-2 provides an approximate transfer function of the temperature sensor.

$$\text{Temp} = 25 - ((V_{TEMP} - V_{TEMP25}) \div m) \qquad \textit{Eqn. 21-2}$$

where:

— V_{TEMP} is the voltage of the temperature sensor channel at the ambient temperature.

— V_{TEMP25} is the voltage of the temperature sensor channel at 25°C.
— m is the hot or cold voltage versus temperature slope in V/°C.

For temperature calculations, use the V_{TEMP25} and m values from the ADC Electricals table.

In application code, the user reads the temperature sensor channel, calculates V_{TEMP}, and compares to V_{TEMP25}. If V_{TEMP} is greater than V_{TEMP25} the cold slope value is applied in Equation 21-2. If V_{TEMP} is less than V_{TEMP25} the hot slope value is applied in Equation 21-2.

For more information on using the temperature sensor, consult AN3031.

21.5.10 MCU Wait Mode Operation

Wait mode is a lower power-consumption standby mode from which recovery is fast because the clock sources remain active. If a conversion is in progress when the MCU enters wait mode, it continues until completion. Conversions can be initiated while the MCU is in wait mode by means of the hardware trigger or if continuous conversions are enabled.

The bus clock, bus clock divided by two, and ADACK are available as conversion clock sources while in wait mode. The use of ALTCLK as the conversion clock source in wait is dependent on the definition of ALTCLK for this MCU. Consult the module introduction for information on ALTCLK specific to this MCU.

If the compare and hardware averaging functions are disabled, a conversion complete event sets the COCOn and generates an ADC interrupt to wake the MCU from wait mode if the respective ADC interrupt is enabled (AIENn=1). If the hardware averaging function is enabled the COCOn will set (and generate an interrupt if enabled) when the selected number of conversions are complete. If the compare function is enabled the COCOn will set (and generate an interrupt if enabled) only if the compare conditions are met. If a single conversion is selected and the compare trigger is not met, the ADC will return to its idle state and cannot wake the MCU from wait mode unless a new conversion is initiated by the hardware trigger.

21.5.11 MCU Stop3 Mode Operation

Stop mode is a low power-consumption standby mode during which most or all clock sources on the MCU are disabled.

21.5.11.1 Stop3 Mode With ADACK Disabled

If the asynchronous clock, ADACK, is not selected as the conversion clock, executing a stop instruction aborts the current conversion and places the ADC in its idle state. The contents of the ADC registers, including ADCRHn and ADCRLn, are unaffected by stop3 mode. After exiting from stop3 mode, a software or hardware trigger is required to resume conversions.

21.5.11.2 Stop3 Mode With ADACK Enabled

If ADACK is selected as the conversion clock, the ADC continues operation during stop3 mode. For guaranteed ADC operation, the MCU's voltage regulator must remain active during stop3 mode. Consult the module introduction for configuration information for this MCU.

If a conversion is in progress when the MCU enters stop3 mode, it continues until completion. Conversions can be initiated while the MCU is in stop3 mode by means of the hardware trigger or if continuous conversions are enabled.

If the compare and hardware averaging functions are disabled, a conversion complete event sets the COCOn and generates an ADC interrupt to wake the MCU from stop3 mode if the respective ADC interrupt is enabled (AIENn = 1). The result register will contain the data from the first completed conversion that occurred during stop3 mode. If the hardware averaging function is enabled the COCOn will set (and generate an interrupt if enabled) when the selected number of conversions are complete. If the compare function is enabled the COCOn will set (and generate an interrupt if enabled) only if the compare conditions are met. If a single conversion is selected and the compare is not true, the ADC will return to its idle state and cannot wake the MCU from stop3 mode unless a new conversion is initiated by another hardware trigger.

NOTE

The ADC module can wake the system from low-power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this scenario, software should ensure the conversion result data transfer blocking mechanism (discussed in Section 21.5.5.2, “Completing Conversions) is cleared when entering stop3 and continuing ADC conversions.

21.5.12 MCU Stop2 Mode Operation

The ADC module is automatically disabled when the MCU enters stop2 mode. All module registers contain their reset values following exit from stop2. Therefore, the module must be re-enabled and re-configured following exit from stop2.

21.6 Initialization Information

This section gives an example that provides some basic direction on how to initialize and configure the ADC module. You can configure the module for 8-, 10-, 12-, or 16-bit single-ended resolution or 9-, 11-, 13-, or 16-bit differential resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. Refer to Table 21-6, Table 21-7, and Table 21-8 for information used in this example.

NOTE

Hexadecimal values designated by a preceding 0x, binary values designated by a preceding %, and decimal values have no preceding character.

21.6.1 ADC Module Initialization Example

21.6.1.1 Initialization Sequence

Before the ADC module can be used to complete conversions, an initialization procedure must be performed. A typical sequence is as follows:

1. Calibrate the ADC by following the calibration instructions in Section 21.5.7.
2. Update the configuration register (ADCCFG) to select the input clock source and the divide ratio used to generate the internal clock, ADCK. This register is also used for selecting sample time and low-power configuration.
3. Update status and control register 2 (ADCSC2) to select the conversion trigger (hardware or software) and compare function options, if enabled.
4. Update status and control register 3 (ADSC3) to select whether conversions will be continuous or completed only once (ADCO) and to select whether to perform hardware averaging.
5. Update status and control register (ADCSC1:ADCSC1n) to select whether conversions will be single-ended or differential and to enable or disable conversion complete interrupts. The input channel on which conversions will be performed is also selected here.

21.6.1.2 Pseudo-Code Example

In this example, the ADC module is set up with interrupts enabled to perform a single 10-bit conversion at low power with a long sample time on input channel 1, where the internal ADCK clock is derived from the bus clock divided by 1.

ADCCFG = 0x98 (%10011000)

```
Bit 7    ADLPC    1        Configures for low power (lowers maximum clock speed.
Bit 6:5  ADIV     00       Sets the ADCK to the input clock ÷ 1.
Bit 4    ADLSMP   1        Configures for long sample time.
Bit 3:2  MODE     10       Sets mode at 10-bit conversions.
Bit 1:0  ADICLK   00       Selects bus clock as input clock source.
```

ADCSC2 = 0x00 (%00000000)

```
Bit 7    ADACT    0        Flag indicates if a conversion is in progress.
Bit 6    ADTRG    0        Software trigger selected.
Bit 5    ACFE     0        Compare function disabled.
Bit 4    ACFGT    0        Not used in this example.
Bit 3:2           00       Reserved, always reads zero.
Bit 1:0           00       Reserved for Freescale's internal use; always write zero.
```

ADCSC1A = 0x41 (%01000001)

```
Bit 7    COCOA    0        Read-only flag which is set when a conversion completes.
Bit 6    AIENA    1        Conversion complete interrupt enabled.
Bit 5    ADCOA    0        One conversion only (continuous conversions disabled).
Bit 4:0  ADCHA    00001    Input channel 1 selected as ADC input channel.
```

ADCRHA/LA = 0xxx

```
Holds results of conversion. Read high byte (ADCRHA) before low byte (ADCRLA) so that
conversion data cannot be overwritten with data from the next conversion.
```

ADCCVH/L = 0xxx

Holds compare value when compare function enabled.

APCTL1=0x02

AD1 pin I/O control disabled. All other AD pins remain general purpose I/O pins.

APCTL2=0x00

All other AD pins remain general purpose I/O pins.

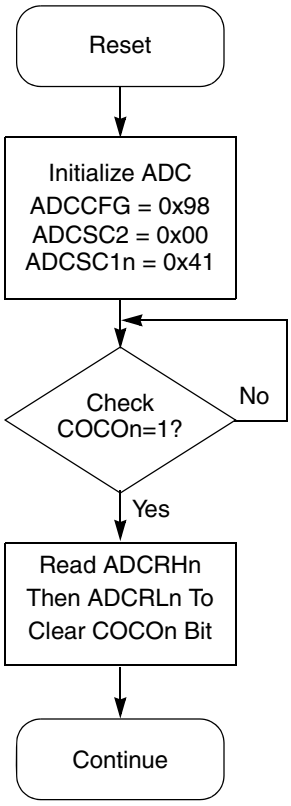


Figure 21-41. Initialization Flowchart for Example

21.7 Application Information

This section contains information for using the ADC module in applications. The ADC has been designed to be integrated into a microcontroller for use in embedded control applications requiring an A/D converter.

21.7.1 External Pins and Routing

The following sections discuss the external pins associated with the ADC module and how they should be used for best results.

21.7.1.1 Analog Supply Pins

The ADC module has analog power and ground supplies (V_{DDAD} and V_{SSAD}) available as separate pins on some devices. V_{SSAD} is shared on the same pin as the MCU digital V_{SS} on some devices. On other devices, V_{SSAD} and V_{DDAD} are shared with the MCU digital supply pins. In these cases, there are separate pads for the analog supplies bonded to the same pin as the corresponding digital supply so that some degree of isolation between the supplies is maintained.

When available on a separate pin, both V_{DDAD} and V_{SSAD} must be connected to the same voltage potential as their corresponding MCU digital supply (V_{DD} and V_{SS}) and must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

If separate power supplies are used for analog and digital power, the ground connection between these supplies must be at the V_{SSAD} pin. This should be the only ground connection between these supplies if possible. The V_{SSAD} pin makes a good single point ground location.

21.7.1.2 Analog Voltage Reference Pins

In addition to the analog supplies, the ADC module has connections for two reference voltage inputs used by the converter, V_{REFSH} and V_{REFSL}. V_{REFSH} is the high reference voltage for the converter. V_{REFSL} is the low reference voltage for the converter.

The ADC can be configured to accept one of three voltage reference pairs for V_{REFSH} and V_{REFSL}. Each pair contains a positive reference and a ground reference. The three pairs are external (V_{REFH} and V_{REFL}), alternate (V_{ALTH} and V_{ALTL}) and the internal bandgap (V_{BGH} and V_{BGL}). These voltage references are selected using the REFSEL bits in the ADCSC2 register. The alternate (V_{ALTH} and V_{ALTL}) voltage reference pair may select additional external pins or internal sources depending on MCU configuration. Consult the module introduction for information on the Voltage References specific to this MCU.

In some packages, the external or alternate pairs are connected in the package to V_{DDAD} and V_{SSAD}, respectively. One of these positive references may be shared on the same pin as V_{DDAD} on some devices. One of these ground references may be shared on the same pin as V_{SSAD} on some devices.

If externally available, the positive reference may be connected to the same potential as V_{DDAD} or may be driven by an external source to a level between the minimum Ref Voltage High (defined in Appendix A) and the V_{DDAD} potential (the positive reference must never exceed V_{DDAD}). If externally available, the ground reference must be connected to the same voltage potential as V_{SSAD}. The voltage reference pairs must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

AC current in the form of current spikes required to supply charge to the capacitor array at each successive approximation step is drawn through the V_{REFH} and V_{REFL} loop. The best external component to meet this current demand is a 0.1 μF capacitor with good high frequency characteristics. This capacitor is connected between V_{REFH} and V_{REFL} and must be placed as near as possible to the package pins. Resistance in the path is not recommended because the current causes a voltage drop that could result in conversion errors. Inductance in this path must be minimum (parasitic only).

21.7.1.3 Analog Input Pins

The external analog inputs are typically shared with digital I/O pins on MCU devices. The pin I/O control is disabled by setting the appropriate control bit in one of the pin control registers. Conversions can be performed on inputs without the associated pin control register bit set. It is recommended that the pin control register bit always be set when using a pin as an analog input. This avoids problems with contention because the output buffer is in its high impedance state and the pullup is disabled. Also, the input buffer draws DC current when its input is not at V_{DD} or V_{SS}. Setting the pin control register bits for all pins used as analog inputs should be done to achieve lowest operating current.

Empirical data shows that capacitors on the analog inputs improve performance in the presence of noise or when the source impedance is high. Use of 0.01 μF capacitors with good high-frequency characteristics is sufficient. These capacitors are not necessary in all cases, but when used they must be placed as near as possible to the package pins and be referenced to V_{SSA}.

For proper conversion, the input voltage must fall between V_{REFH} and V_{REFL}. If the input is equal to or exceeds V_{REFH}, the converter circuit converts the signal to 0xFFF (full scale 12-bit representation), 0x3FF (full scale 10-bit representation) or 0xFF (full scale 8-bit representation). If the input is equal to or less than V_{REFL}, the converter circuit converts it to 0x000. Input voltages between V_{REFH} and V_{REFL} are straight-line linear conversions. There is a brief current associated with V_{REFL} when the sampling capacitor is charging.

For minimal loss of accuracy due to current injection, pins adjacent to the analog input pins should not be transitioning during conversions.

21.7.2 Sources of Error

Several sources of error exist for A/D conversions. These are discussed in the following sections.

21.7.2.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately 7kΩ and input capacitance of approximately 5.5 pF, sampling to within 1/4LSB (at 12-bit resolution) can be achieved within the minimum sample window (3.5 cycles @ 8 MHz maximum ADCK frequency) provided the resistance of the external analog source (R_{AS}) is kept below 2 kΩ.

Higher source resistances or higher-accuracy sampling is possible by setting ADLSMP and changing the ADLSTS bits (to increase the sample window) or decreasing ADCK frequency to increase sample time.

21.7.2.2 Pin Leakage Error

Leakage on the I/O pins can cause conversion error if the external analog source resistance (R_{AS}) is high. If this error cannot be tolerated by the application, keep R_{AS} lower than $V_{DDAD} / (2^N * I_{LEAK})$ for less than 1/4LSB leakage error (N = 8 in 8-bit, 10 in 10-bit or 12 in 12-bit mode).

21.7.2.3 Noise-Induced Errors

System noise that occurs during the sample or conversion process can affect the accuracy of the conversion. The ADC accuracy numbers are guaranteed as specified only if the following conditions are met:

- There is a 0.1 μF low-ESR capacitor from V_{REFH} to V_{REFL}.
- There is a 0.1 μF low-ESR capacitor from V_{DDAD} to V_{SSAD}.
- If inductive isolation is used from the primary supply, an additional 1 μF capacitor is placed from V_{DDAD} to V_{SSAD}.
- V_{SSAD} (and V_{REFL}, if connected) is connected to V_{SS} at a quiet point in the ground plane.
- Operate the MCU in wait or stop3 mode before initiating (hardware triggered conversions) or immediately after initiating (hardware or software triggered conversions) the ADC conversion.
  - For software triggered conversions, immediately follow the write to ADCSC1 with a wait instruction or stop instruction.
  - For stop3 mode operation, select ADACK as the clock source. Operation in stop3 reduces V_{DD} noise but increases effective conversion time due to stop recovery.
- There is no I/O switching, input or output, on the MCU during the conversion.

There are some situations where external system activity causes radiated or conducted noise emissions or excessive V_{DD} noise is coupled into the ADC. In these situations, or when the MCU cannot be placed in wait or stop3 or I/O activity cannot be halted, these recommended actions may reduce the effect of noise on the accuracy:

- Place a 0.01 μF capacitor (C_{AS}) on the selected input channel to V_{REFL} or V_{SSAD} (this improves noise issues, but affects the sample rate based on the external analog source resistance).
- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock (ADACK) and averaging. Noise that is synchronous to ADCK cannot be averaged out.

21.7.2.4 Code Width and Quantization Error

The ADC quantizes the ideal straight-line transfer function into 4096 steps (in 12-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N bit converter (in this case N can be 8, 10 or 12), defined as 1LSB, is:

$$1\ lsb = (V_{REFH} - V_{REFL}) / 2^N \qquad \textit{Eqn. 21-3}$$

There is an inherent quantization error due to the digitalization of the result. For 8-bit or 10-bit conversions the code transitions when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be ± 1/2 lsb in 8- or 10-bit mode. As a consequence, however, the code width of the first (0x000) conversion is only 1/2 lsb and the code width of the last (0xFF or 0x3FF) is 1.5 lsb.

For 12-bit conversions the code transitions only after the full code width is present, so the quantization error is –1 lsb to 0 lsb and the code width of each step is 1 lsb.

21.7.2.5 Linearity Errors

The ADC may also exhibit non-linearity of several forms. Every effort has been made to reduce these errors but the system should be aware of them because they affect overall accuracy. These errors are:

- Zero-scale error (E_{ZS}) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width (1/2 lsb in 8-bit or 10-bit modes and 1 lsb in 12-bit mode). If the first conversion is 0x001, the difference between the actual 0x001 code width and its ideal (1 lsb) is used.
- Full-scale error (E_{FS}) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width (1.5 lsb in 8-bit or 10-bit modes and 1LSB in 12-bit mode). If the last conversion is 0x3FE, the difference between the actual 0x3FE code width and its ideal (1LSB) is used.
- Differential non-linearity (DNL) — This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.
- Integral non-linearity (INL) — This error is defined as the highest-value the (absolute value of the) running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total unadjusted error (TUE) — This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function and includes all forms of error.

21.7.2.6 Code Jitter, Non-Monotonicity, and Missing Codes

Analog-to-digital converters are susceptible to three special forms of error. These are code jitter, non-monotonicity, and missing codes.

Code jitter is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code (and vice-versa). However, even small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around ±1/2 lsb in 8-bit or 10-bit mode, or around 2 lsb in 12-bit mode, and increases with noise.

This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in Section 21.7.2.3 reduces this error.

Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage. Missing codes are those values never converted for any input value.

In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and have no missing codes.

Chapter 22
Programmable Delay Block (PDB)

22.1 Introduction

22.1.1 Overview

Many applications need to synchronize the time at which multiple ADC samples are taken with respect to an external trigger or event. The primary function of the programmable delay block is simply to provide controllable delays from the either an external trigger, or a programmable interval tick, to the sample trigger input of one or more ADCs.

22.1.2 Features

- Positive transition of trigger_in will initiate the counter
- Four configurable channels
  - — Each channel can supply two pre-trigger events to two ADC trigger select inputs.
  - — Each trigger output is individually controlled.
- Continuous trigger or single shot mode supported
- Bypass mode supported
- Each trigger output can be independently enabled.
- One programmable delay interrupt
- One sequence error interrupt

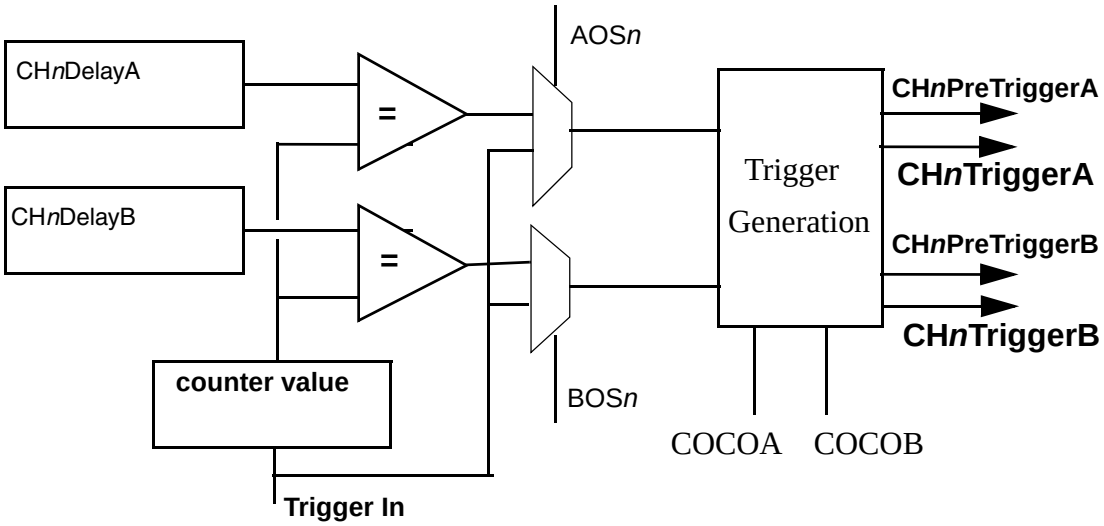
22.1.3 Modes of Operation

Modes of operation include:

- Disabled: Counter is off and all pre-trigger and trigger outputs are low.
- Enabled OneShot: Counter is enabled & restarted at count zero upon receiving a positive edge on the trigger input. Each TriggerA and TriggerB will see only one output trigger per input trigger.
- Enabled Continuous: Counter is enabled & restarted at count zero. The counter will be rolled over to zero again when the count reaches the value specified in the MOD register, and counting restarted. This enables a continuous stream of triggers out as a result of a single trigger input.
- Bypassed: The input trigger bypasses the PDB logic entirely. It IS possible to bypass only one of the trigger output; therefore this mode can be used in conjunction with any of the above.

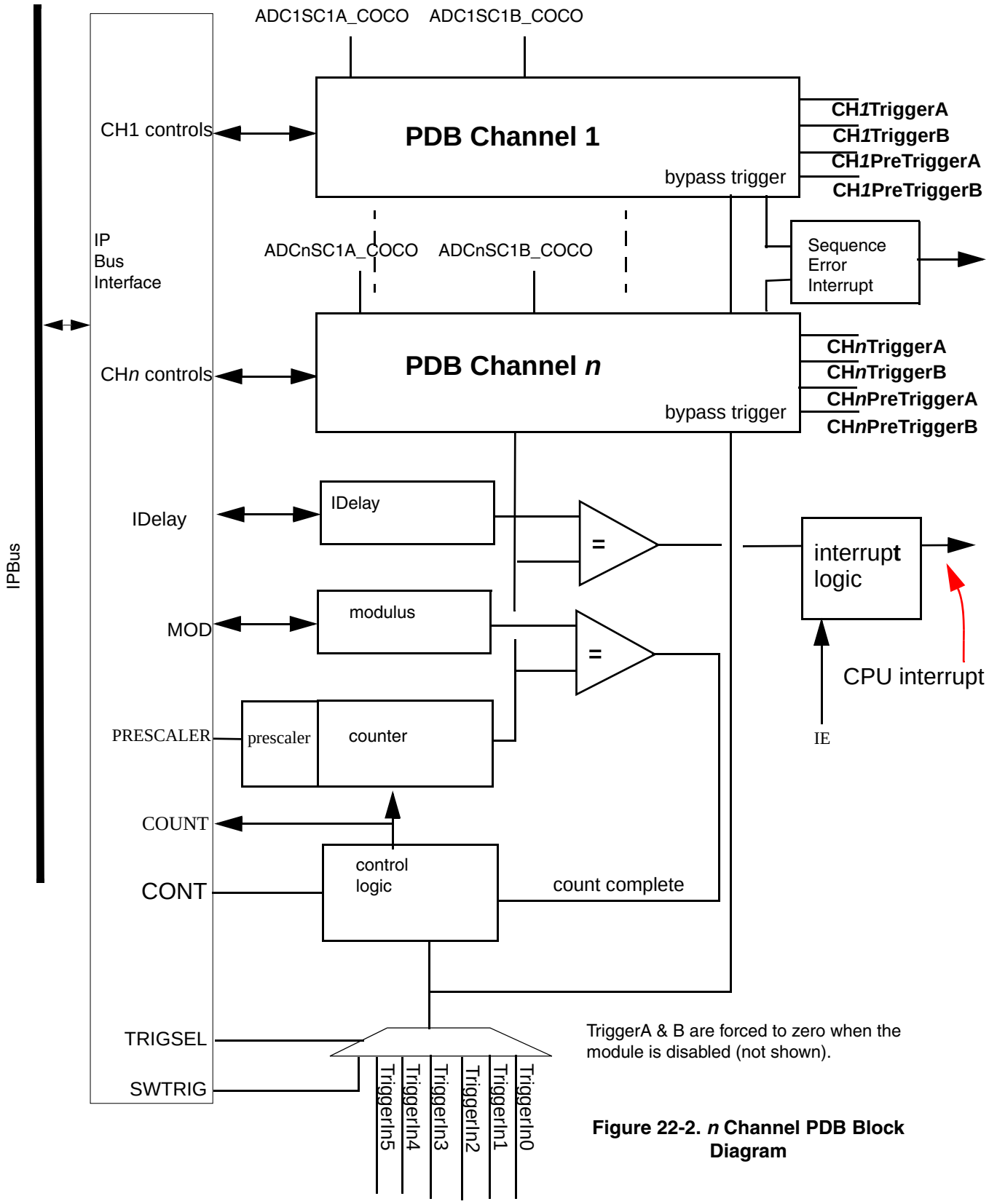
22.1.4 Block Diagram

Figure 22-1 and Figure 22-2 illustrate the basic structure of the PDB block.



Not shown are the ENA and ENB controls for each channel. These can be used to gate off the channel outputs.

Figure 22-1. PDB Channel Block Diagram



TriggerA & B are forced to zero when the module is disabled (not shown).

Figure 22-2. *n* Channel PDB Block Diagram

22.2 Memory Map and Registers

22.2.1 Memory Map

The memory map of PDB will expand as the number of channels increases above 1. On MCF51EM256 series MCU has 4 channels built in the PDB. In Table 22-1, *n* is the channel number for channel.

Offset (bytes)	Register	Description
0x00	PDBSC	PDB Status & Control Register
0x02	PDBMOD	PDB Counter Modulus Register
0x04	PDBCNT	PDB Counter Value (READ ONLY)
0x06	PDBIDLY	PDB Comparison Value for Interrupt Timer
0x08 + (n x 0x08)	PDBCH*n*CR	PDB Channel n Control Register
0x08 + (n x 0x08) + 2	PDBCH*n*DLYA	PDB Channel n Delay A Register
0x08 + (n x 0x08) + 4	PDBCH*n*DLYB	PDB Channel n Delay B Register
0x08 + (n x 0x08) + 6	Reserved	Reserved

Table 22-1. PDB Memory Map

Note that the memory map extends four extra registers per PDB channel.

22.2.2 Registers Descriptions

22.2.2.1 PDB Status and Control Register (PDBSC)

This register contains status and control bits for the Programmable Delay Block. The counter is enabled if EN has been set to one.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PRESCALER			RESERVED					IF	CONT	SWTRIG	TRIGSEL			IE	EN
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Reserved or unused

Figure 22-3. PDB Status and Control Register (PDBSC)

Table 22-2. PDBSC Register Field Descriptions

Field	Description
15:13 PRESCALER	**Clock Prescaler Select** 000 = timer uses peripheral clock 001 = timer uses peripheral clock / 2 010 = timer uses peripheral clock / 4 011 = timer uses peripheral clock / 8 100 = timer uses peripheral clock / 16 101 = timer uses peripheral clock / 32 110 = timer uses peripheral clock / 64 111 = timer uses peripheral clock / 128
12:8 RESERVED	**RESERVED**. Write these register bits as zero.
7 IF	**Interrupt Flag** 0 = A comparison event has not been detected (The count value has equaled IDELAY at some point) 1 = A comparison event has been detected (In this, or a previous PDB cycle, the count has equaled IDELAY) If the module is enabled (EN=1) and the interrupt enable (IE) bit has been set, an interrupt will be issued when the IF bit is set. This bit can be cleared by writing a one to it. The IF bit will be set (when COUNT=IDELAY) regardless of whether or not the interrupt has been enabled.
6 CONT	**Continuous Mode Enable** 0 = Module is in OneShot mode 1 = Module is in continuous mode
5 SWTRIG	**Software Trigger** — When TRIGSEL=3'b111 and the module is enabled, writing a one to this field will trigger a reset and restart of the counter. Alternately, if TriggerA or TriggerB are bypassed, it will propagate there immediately. This bit always reads as zero.
4:2 TRIGSEL	**Input Trigger Select** 000 = Comparator 1 output 001 = Comparator 2 output 100 = EXTRIG 111 = SWTRIG All other values are reserved.
1 IE	**IDelay match Interrupt Enable** 0 = IDelay Interrupt is not enabled 1 = IDelay Interrupt is enabled Note: This bit does not affect the sequence error interrupt. The sequence error interrupt is always enabled, when the PDB is enabled.
0 EN	**PDB Enable** 0 = Module Counter is Disabled 1 = Module Counter is Enabled Setting EN=0 will set the count register to zero.

22.2.2.2 PDB Modulus Register (PDBMOD)

This register specifies the period of the counter in terms of PDB source clock cycles. When the counter reaches this value, it will be reset back to all zeros. If PDBSC_CONT is set to one, the count will begin anew.

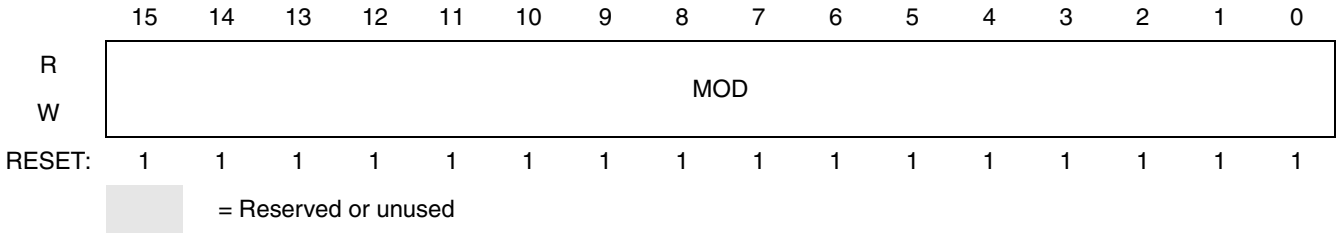


Figure 22-4. PDB Modulus Register (PDBMOD)

22.2.2.3 PDB Counter Register (PDBCNT)

This register can be used to read the current value of the counter. It is READ ONLY.

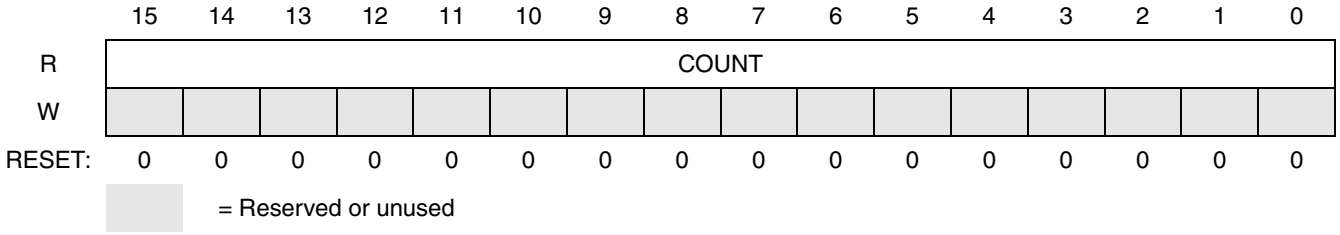


Figure 22-5. PDB Counter Register (PDBCNT)

This register is set to zero when PDBSC_EN = 0. The count does not start until a hardware or software trigger event occurs.

22.2.2.4 PDB Interrupt Delay Register (PDBIDLY)

This register can be used to read and write the value used to schedule the PDB IDelay interrupt. This feature can be used to schedule an independent interrupt at some point in the PDB cycle.

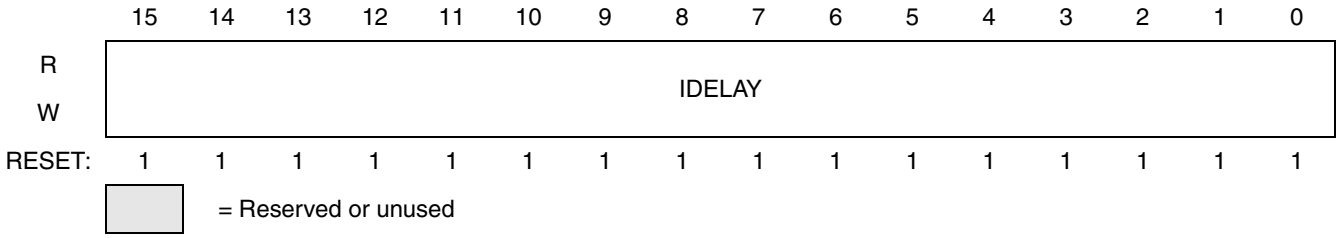


Figure 22-6. PDB Interrupt Delay Register (PDBIDLY)

PDBSC_IE must be set in order for an interrupt to be issued as a result of the count value equaling IDELAY. However, PDBSC_IF will be set whenever COUNT=IDELAY.

22.2.2.5 PDB Channel *n* Control Register (PDBCH*n*CR)

This register contains status and control bits for the channel *n* of the Programmable Delay Block. The channel outputs are enabled if either ENA or ENB have been set to one.

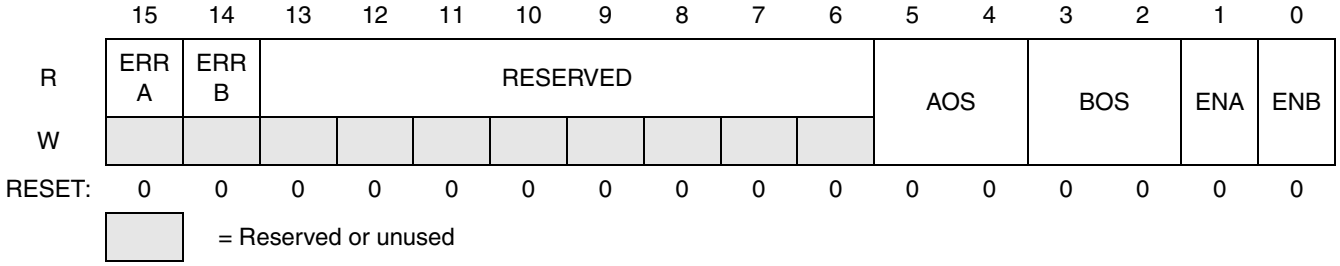


Figure 22-7. Programmable Delay Block Channel *n* Control Register (PDBCH*n*CR)

Table 22-3. PDBCH*n*CR Register Field Descriptions

Field	Description
15 ERRA	**Sequence error on TriggerA** 0 = No sequence error on TriggerA 1 = Sequence error is detected on TriggerA. The Delay A is timed out before the previous ADC conversion, which is triggered by TriggerB, is done. A PDB sequence error interrupt is generated when this bit is set. Write 1 clear this bit.
14 ERRB	**Sequence error on TriggerB** 0 = No sequence error on TriggerB 1 = Sequence error is detected on TriggerB. The Delay B is timed out before the previous ADC conversion, which is triggered by TriggerA, is done. A PDB sequence error interrupt is generated when this bit is set. Write 1 clear this bit.
13:6 RESERVED	**RESERVED**. Write these register bits as zero.
5:4 AOS	**Channel *n* TriggerA Output Select** 00 = Counter delay is bypassed 01 = TriggerA is function of Delay A 10 = Reserved 11 = Reserved
3:2 BOS	**Channel *n* TriggerB Output Select** 00 = Counter delay is bypassed 01 = TriggerB is function of Delay B 10 = Reserved 11 = Reserved
1 ENA	**TriggerA Enable** 0 = PreTriggerA and TriggerA outputs are disabled (and forced to zero) 1 = PreTriggerA and TriggerA outputs are enabled
0 ENB	**TriggerB Enable** 0 = PreTriggerB and TriggerB outputs are disabled (and forced to zero) 1 = PreTriggerB and TriggerB outputs are enabled

22.2.2.6 PDB Channel *n* Delay A & Delay B Registers (PDBCH*n*DLYA & PDBCH*n*DLYB)

These registers are used to specify the delay from assertion of TriggerIn to assertion of PreTriggerA and PreTriggerB out for a given channel. These registers are repeated for each channel present on a given instance of the PDB. These delays are only applicable if the module is enabled and the output trigger in question has not been bypassed. In each case, the delay is in terms of PDB source clock cycles.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DELAY A															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Reserved or unused

Figure 22-8. PDB Channel *n* Delay A Register (PDBCh*n*DLYA

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DELAY B															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Reserved or unused

Figure 22-9. PDB Channel *n* Delay B Register (PDBCH*n*DLYB)

22.2.3 Functional Description

The PDB contains a single counter whose output is compared against a minimum of three different digital values. For each channel, DelayA and DelayB determine the time between assertion of the trigger input to the point at which changes in the trigger output signals are initiated. These times are defined as:

- trigger input to Pre-TriggerA = (prescaler X DelayA) + 2 bus clock cycles
- trigger input to Pre-TriggerB = (prescaler X DelayB) + 2 bus clock cycles
- Add one additional bus clock cycle to determine the time at which the trigger outputs change.

Pre-TriggerA and Pre-TriggerB are used to precondition the ADC blocks two bus clock periods prior to the actual measurement trigger. The ADC16V1 on MCF51EM256 series MCU contains two sets of control and result registers, allowing them to operate in a ping-pong fashion, alternating conversions between two different analog sources (per converter). The Pre-Trigger signals are used to specify which signal will be sampled next. When PreTriggerA and TriggerA is asserted, the ADC conversion is triggered with set A of the control and results registers. When PreTriggerB and TriggerB is asserted, the ADC conversion is triggered with set B of the control and results registers.

The signals shown in Figure 22-10 would be used to operate on any of the 4 ADCs. The trigger delays for the ADCs are independently set via the DELAYA and DELAYB parameters.

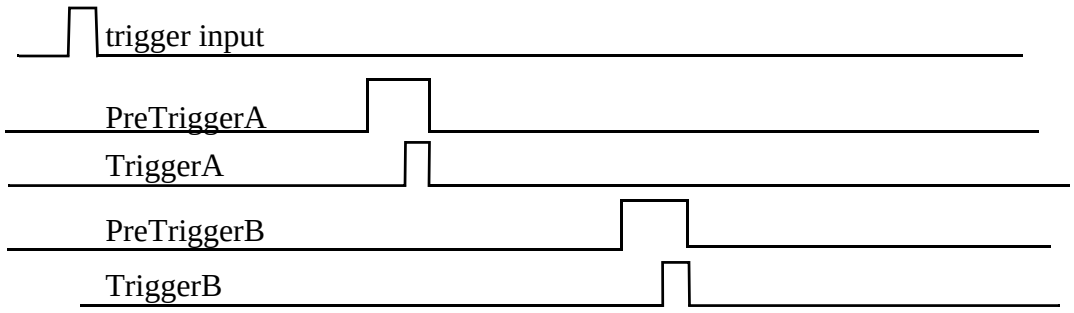


Figure 22-10. Decoupled A & B Trigger Generation

The value, modulus, is used to reset the counter back to zero at the end of the count. If PDBSC_CONT is set, the counter will then resume a new count. Otherwise, the timer operation will cease until the next trigger input event occurs.

In MCF51EM256 series MCUs, the PDB Channel n PreTriggerA and PreTriggerB is connected to ADCn Trigger Select Events ADHWTSA and ADHWTSB correspondingly. Either TriggerA or TriggerB can trigger the ADC conversion. When TriggerA triggers the ADC conversion, control and results register set A is used; when TriggerB triggers the ADC conversion, control and results register set B is used.

The Delay A and Delay B registers should be configured to make the next trigger asserted after the previous ADC conversion is finished.

When one conversion, triggered by TriggerA is in progress, the TriggerB output is suppressed, until the ADCnSC1A_COCO bit is set. If Delay B is timed out during the ADC conversion triggered by TriggerA, the Sequence Error bit PDBCH*n*SC_ERRB will be set.

When one conversion, triggered by TriggerB is in progress, the TriggerA output is suppressed, until the ADCnSC1B_COCO bit is set. If Delay A is timed out during the ADC conversion triggered by TriggerB, the Sequence Error bit PDBCH*n*SC_ERRA will be set.

A PDB sequence error interrupt will be generated, if any of the PDBCH*n*SC_ERRA or PDBCH*n*SC_ERRB on any of the 4 channels is set. The sequence error interrupt cannot be disabled in PDB module.

Chapter 23
Voltage Reference (VREF)

23.1 Introduction

The Voltage Reference (VREF) is intended to supply an accurate voltage output that is trimmable by an 8-bit register in 0.5 mV steps. This reference can be used to provide a reference voltage to external devices or internal analog peripherals such as the ADC, DAC, or ACMP. The voltage reference has two operation modes that provide different levels of load regulation and power consumption. Figure 23-1 is a block diagram of the Voltage Reference.

23.1.1 Overview

The Voltage Reference optimizes the existing bandgap and bandgap buffer system. Unity gain amplifiers ease the design and keep power consumption low. The 8-bit trim register is user accessible so that the voltage reference can be trimmed in application.

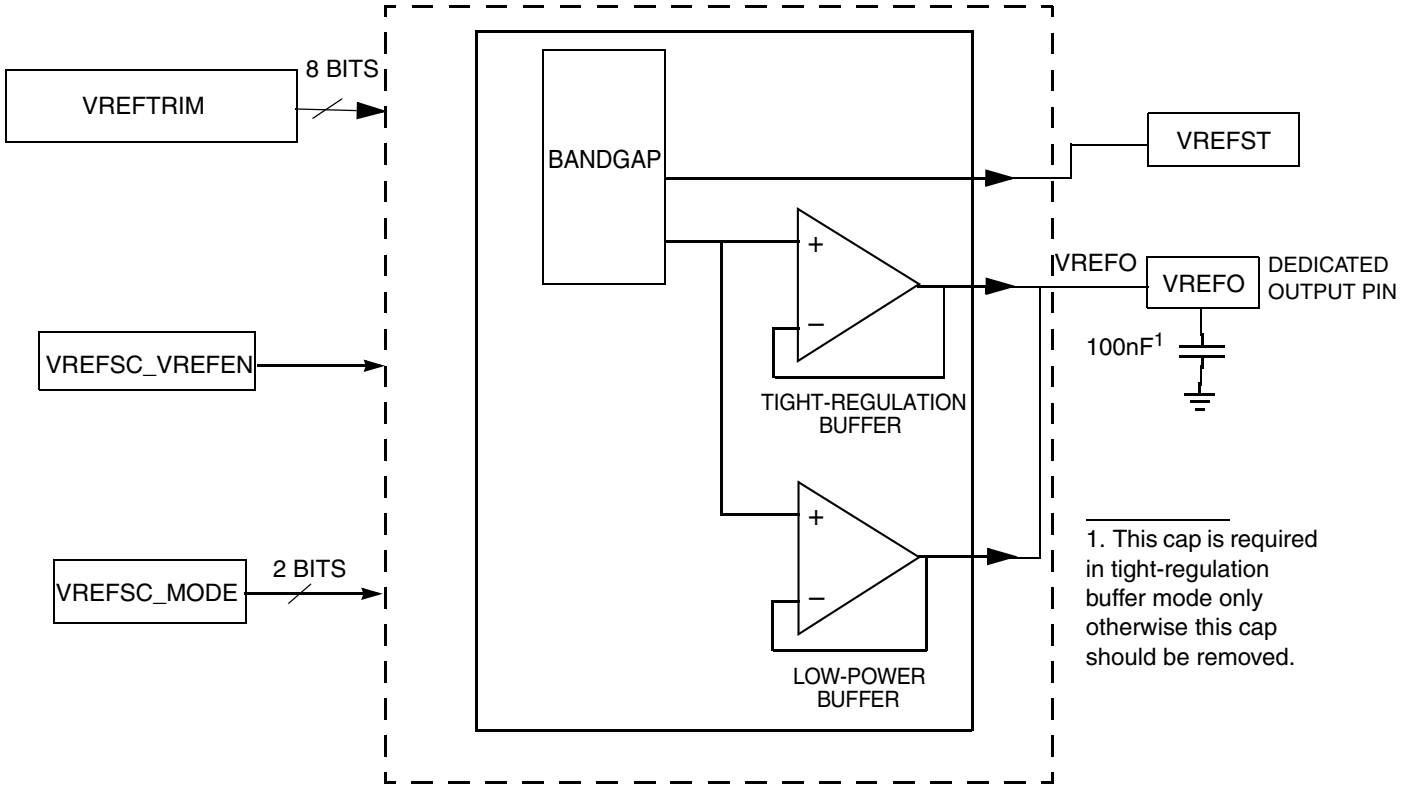


Figure 23-1. Voltage Reference Block Diagram

The voltage reference contains trim resolution of 0.5mV per step. The buffer has modes with improved high-current swing output. In addition, a low-power buffer mode is available for use with ADCs or DACs. The voltage reference can be output on a dedicated output pin[1].

23.1.2 Features

The Voltage Reference module has the following features:

- Programmable trim register with 0.5mV steps, automatically loaded with factory trimmed value upon reset
- Programmable mode selection:
  - — Off
  - — Bandgap out (or stabilization delay)
  - — Low-power buffer mode
  - — Tight-regulation buffer mode
- 1.2 V output at room temperature, 40 ppm/C
- Dedicated output pin VREFO
- Load regulation in tight-regulation mode of 100 uV/mA max
- PSR of 0±0.1 mV DC and -60dB AC

23.1.3 Modes of Operation

The Voltage Reference continues normal operation in Run, Wait, LPRun, and LPWait modes. The Voltage Reference module can be enabled to operate in STOP3 mode.

23.1.4 External Signal Description

Table 23-1 shows the Voltage Reference signals properties.

Table 23-1. Signal Properties

Name	Function	I/O	Reset	Pull Up
VREFO	Internally generated voltage reference output	O	—	—

1. In Tight-Regulation buffer mode, a 100nF capacitor is required to be connected between the VREFO pad and the ground.

23.2 Memory Map and Register Definition

23.2.1 VREF Trim Register (VREFTRM)

The VREFTRM register contains eight bits that contain the trim data for the Voltage Reference as described in Table 23-2.

Register address: Base + 0x00

	7	6	5	4	3	2	1	0
R W	TRM[7:0]							
Reset:	X	X	X	X	X	X	X	X

Figure 23-2. VREF Trim Register (VREFTRM)

Table 23-2. VREFTRM Register Field Descriptions

Field	Description
7:0 TRM[7:0]	Trim Bits 7:0 These bits change the resulting VREF output by ±0.5mV for each step.

Table 23-3. VREFTRM Register Settings

TRM7	TRM6	TRM5	TRM4	TRM3	TRM2	TRM1	TRM0	VREF (mV)
1	0	0	0	0	0	0	0	max
1	0	0	0	0	0	0	1	max-0.5
1	0	0	0	0	0	1	0	max-1.0
1	0	0	0	0	0	1	1	max-1.5
1	. . . . .							. . . . .
1	1	1	1	1	1	0	0	mid+1.5
1	1	1	1	1	1	0	1	mid+1.0
1	1	1	1	1	1	1	0	mid+0.5
1	1	1	1	1	1	1	1	mid
0	0	0	0	0	0	0	0	mid-0.5
0	0	0	0	0	0	0	1	mid-1.0
0	0	0	0	0	0	1	0	mid-1.5
0	0	0	0	0	0	1	1	mid-2.0
0	. . . . .							. . . . .
0	1	1	1	1	1	0	0	min+1.5
0	1	1	1	1	1	0	1	min+1.0
0	1	1	1	1	1	1	0	min+0.5
0	1	1	1	1	1	1	1	min

23.2.2 VREF Status and Control Register (VREFSC)

The VREF status and control register contains the control bits used to enable the internal voltage reference and select the buffer mode to be used.

Register address: Base + 0x01

	7	6	5	4	3	2	1	0
R	VREFEN	0	0	0	0	VREFST	MODE	
W								
Reset:	0	0	0	0	0	0	0	0

Figure 23-3. Voltage Reference Control Register (VREFSC)

Table 23-4. VREFSC Register Field Descriptions

Field	Description
7 VREFEN	**Internal Voltage Reference Enable** — This bit is used to enable the Voltage Reference module. 0 the Voltage Reference module is disabled 1 the Voltage Reference module is enabled
2 VREFST	**Internal Voltage Reference Stable** — This bit indicates that the Voltage Reference module has completed its startup and stabilization. 0 Voltage Reference module is disabled or not stable 1 Voltage Reference module is stable
1:0 MODE[1:0]	**Mode selection** — These bits are used to select the modes of operation for the Voltage Reference module. 00 Bandgap on only, for stabilization and startup 01 Low-power buffer enabled 10 Tight-regulation buffer enabled 11 RESERVED

23.3 Functional Description

The Voltage Reference is a bandgap buffer system. Unity gain amplifiers are used.

The Voltage Reference can be used in two main cases. It can be used as a reference to analog peripherals such as an ADC channel or analog comparator input. For this case, the low-power buffer can be used. When the tight-regulation buffer is enabled, VREFO can be used both internally and externally.

Table 23-5 shows all possible functional configurations of the Voltage Reference.

Table 23-5. Voltage Reference Functional Configurations

VREFEN	MODE[1:0]	Configuration	Functionality
0	X	Voltage Reference Disabled	Off
1	00	Voltage Reference Enabled, Bandgap on only	Startup and Standby
1	01	Voltage Reference Enabled, Low-Power buffer on	Can be used for internal peripherals only and VREFO pin should not be loaded
1	10	Voltage Reference Enabled, Tight-Regulation buffer on	Can be used both internally and externally. A 100nF capacitor is required on VREFO and 10mA max drive strength is allowed.
1	11	Voltage Reference Disabled	RESERVED

23.3.1 Voltage Reference Disabled, VREFEN=0

When VREFEN=0, the Voltage Reference is disabled, all bandgap and buffers are disabled. The Voltage Reference is in off mode.

23.3.2 Voltage Reference Enabled, VREFEN=1

When VREFEN=1, the Voltage Reference is enabled, and different modes should be set by the Mode[1:0] bits.

23.3.2.1 Mode[1:0]=00

The internal bandgap is on to generate an accurate voltage output that can be trimmed by the bits TRM[7:0] in 0.5 mV steps. The bandgap requires some time for startup and stabilization. The VREFST bit can be monitored to determine if the stabilization and startup is complete.

Both low-power buffer and tight-regulation buffer are disabled in this mode, and there is no buffered voltage output. The Voltage Reference is in standby mode.

23.3.2.2 Mode[1:0]=01

The internal bandgap is on.

The low-power buffer is enabled to generate a buffered internal voltage. It can be used as a reference to internal analog peripherals such as an ADC channel or analog comparator input.

23.3.2.3 Mode[1:0]=10

The internal bandgap is on.

The tight-regulation buffer is enabled to generate a buffered voltage to VREFO with load regulation less than 100 uV/mA. A 100nF capacitor is required on VREFO pin and it is allowed to drive 10 mA maximum current. VREFO can be used internally and/or externally.

23.3.2.4 Mode[1:0]=11

RESERVED

23.4 Initialization Information

The Voltage Reference requires some time for startup and stabilization. Once the VREFEN bit is set, the VREFST bit can be monitored to determine if the stabilization and startup is complete.

When the Voltage Reference is already enabled and stabilized, changing the Mode selection bits (MODE[1:0]) will not clear the VREFST bit, but there will be some startup time before the output voltage is stabilized when the low-power buffer or tight-regulation buffer is enabled, and there will be some setting time when a step change of the load current occurs.

Chapter 24
LCD Driver Module

24.1 Introduction

The LCD driver module is a CMOS charge pump voltage inverter that is designed for low-voltage, low-power operation. The LCD driver module is designed to generate the appropriate waveforms to drive multiplexed numeric, alpha-numeric, or custom LCD panels. Depending on LCD module hardware and software configuration, the LCD panels can be either 3 V or 5 V. The LCD module also has several timing and control settings that can be software configured depending on the applications requirements. Timing and control consists of registers and control logic for:

- LCD frame frequency
- Duty cycle select
- Frontplane/backplane select and enable
- Blink modes and frequency
- Operation in low-power modes

100 pin members of the MCF51EM256 series family have a total of 44 available LCD signals pins. Up to eight of these can be configured for back plane use. 80 pin members of the family have 37 LCD signal pins available. OSCOUT1 that comes from the IRTC is the OSCOUT input to the LCD module and the OSCOUT2 that comes from the ICS external crystal is the alternate clock. Several possible LCD configurations are summarized in Table 24-1.

Table 24-1. Example Configurations by Package Type

100 pin device 44 LCD pins available	80 pin device 37 LCD pins available	No. of Backplanes	No. of Frontplanes	No. of Segments
X		8	36	288
X		2	40	80
X	X	8	25	200

24.1.1 Clock Sources for LCD

On this device, OSCOUT1 is connected to the OSCOUT clock input and OSCOUT2 is connected to the alternate clock input. Refer to section 21.5.1 for details on LCD clock selection.

24.1.2 User Considerations

In LCD module, only LCDC0, LCDC1, LCDSUPPLY, LCDRVC, LCDBCTL, and LCDS registers can be read, while the rest of the LCD registers are write only. Because of this, application software is responsible for keeping track the current values by shadowing values in RAM variables. While powered from the line voltage, the application may choose to update all segments rather than track current values since power consumption is not an issue. However, during blackout conditions, if the meter is required to maintain and update the display, the need for power efficient updates is greater.

Shadowing the registers in RAM will add some s/w overhead, however it should add little overhead to execution time.

CAUTION

LCD registers require several cycles between write accesses. Only byte writes should be used to write to these registers. Finally, consecutive writes must be separated by at least one non-write cycle.

NOTE

Refer to Table 21-1 for the ADC and channel that the LCD V_{LL1} is connected to.

MCF51EM256 series do not have the VLCD pin, any reference to it must be ignored.

24.1.3 Features

The LCD module driver features include:

- LCD waveforms functional in LP_{Run}, LP_{Wait}, wait, stop2 and stop3 low-power modes
- 44 LCD (LCD[43:0])pins with selectable frontplane/backplane configuration
  - — Generate up to 43 frontplane signals
  - — Generate up to 8 backplanes signals
- Programmable LCD frame frequency
- Programmable blink modes and frequency
  - — All segments blank during blink period
  - — Alternate display for each LCD segment in x4 or less mode
  - — Blink operation in low-power modes
- Programmable LCD power supply switch, making it an ideal solution for battery-powered and board-level applications
  - — Charge pump requires only four external capacitors
  - — Internal LCD power using V_{DD} (1.8to 3.6 V)
  - — Internal V_{IREG} regulated power supply option for 3 or 5V LCD glassExternal V_{LL3} power supply option (3V)
- Internal regulated voltage source with a 4-bit trim register to apply contrast control
- Integrated charge pump for generating LCD bias voltages
  - — Hardware configurable to drive 3-V or 5-V LCD panels
  - — On-chip generation of bias voltages
- Waveform storage registers LCDWF
- Backplane reassignment to assist in vertical scrolling on dot-matrix displays
- Software configurable LCD frame frequency interrupt
- Internal ADC channel is connected to V_{LL1} to monitor its magnitudes. This feature allows software to adjust the contrast.

24.1.4 Modes of Operation

The LCD module supports the following operation modes:

Table 24-2. LCD-Module Operation Modes

Mode	Operation
Stop2	Depending on the state of the LCDSTP bit, the LCD module can operate an LCD panel in stop2 mode. If LCDSTP = 1, LCD module clock generation is turned off and the LCD module enters a power conservation state and is disabled.If LCDSTP = 0, the LCD module can operate an LCD panel in stop2, and the LCD module continues to display the current LCD panel contents based on the LCD operation prior to the stop2 event. If the LCD is enabled in stop2, the selected LCD clock source, OSCOUT, must be enabled to operate in stop2. The LCD frame interrupt does not cause the MCU to exit stop2.
Stop3	Depending on the state of the LCDSTP bit, the LCD module can operate an LCD panel in stop3 mode. If LCDSTP = 1, LCD module clock generation is turned off and the LCD module enters a power conservation state and is disabled. If LCDSTP = 0, the LCD module can operate an LCD panel in stop3, and the LCD module continues displaying the current LCD panel contents based on the LCD operation prior to the stop3 event. If the LCD is enabled in stop3, the selected LCD clock source, OSCOUT or the Alternate Clock, must be enabled to operate in stop3. In stop3 mode the LCD frame interrupt can cause the MCU to exit stop3.
LP_{Wait}, Wait	Depending on the configuration, the LCD module can operate an LCD panel in wait mode. If LCDWAI = 1, the LCD module clock generation is turned off and the LCD module enters a power-conservation state and is disabled. If LCDWAI = 0, the LCD module can operate an LCD panel in wait, and the LCD module continues displaying the current LCD panel contents base on the LCDWF registers. In wait mode, the LCD frame interrupt can cause the MCU to exit wait.

Stop2 provides the lowest power consumption state where the LCD module is functional. To operate the LCD in stop2 mode, use an external crystal.

24.1.5 Block Diagram

Figure 24-1 is a block diagram of the LCD module.

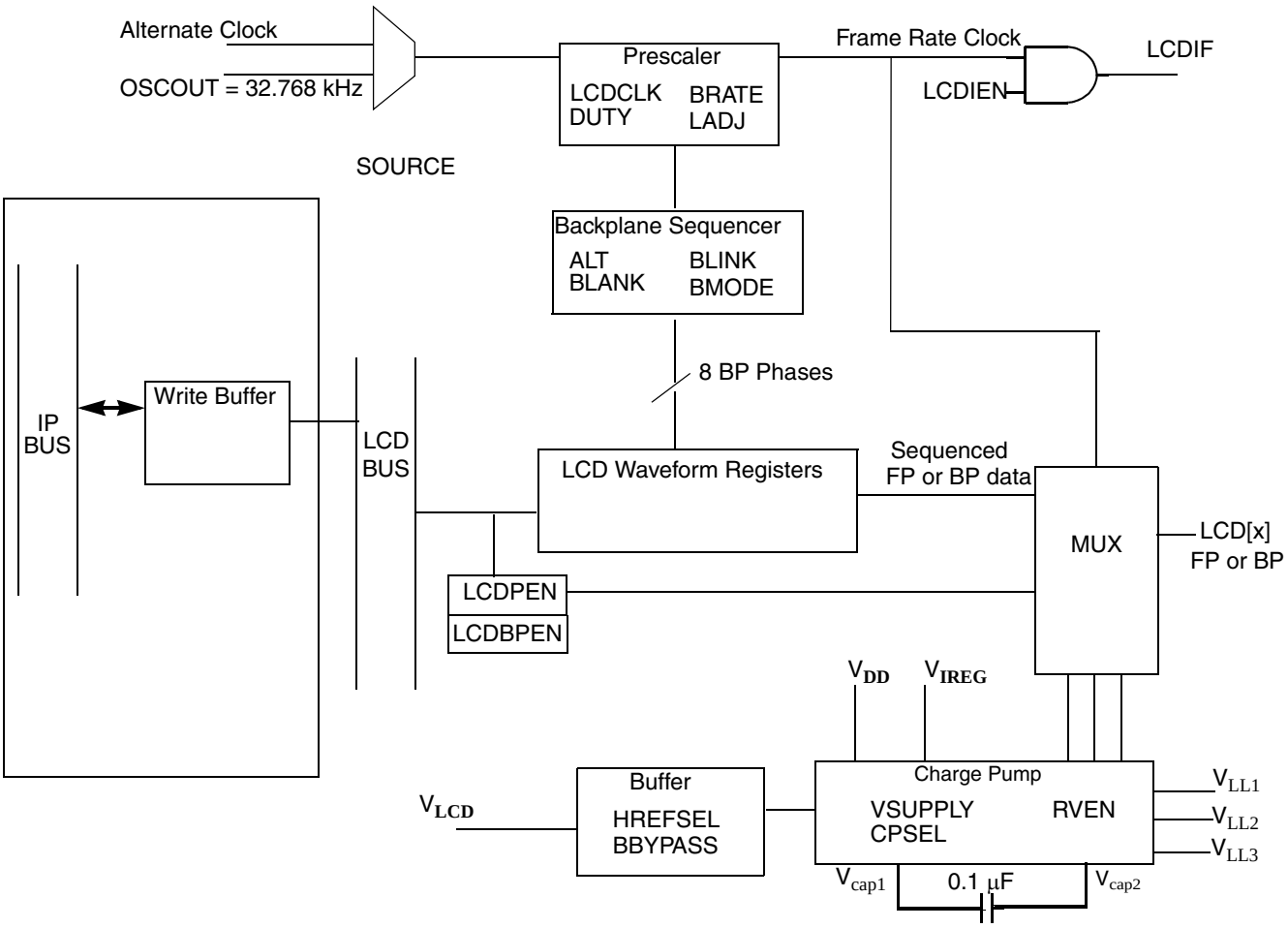


Figure 24-1. LCD Driver Block Diagram

24.2 External Signal Description

The LCD module has several external pins dedicated to power supply and LCD frontplane/backplane signaling. The LCD module can be configured to support eight backplane signals. The table below itemizes all the LCD external pins. See the Pins and Connections chapter for device-specific pin configurations.

Table 24-4. Signal Properties

Name	Port	Function	Reset State
44 LCD frontplane/backplane	LCD[43:0]	Switchable frontplane/backplane driver that connects directly to the display LCD[43:0] can operate as GPIO pins	High impedance
LCD bias voltages	V_{LL1}, V_{LL2}, V_{LL3}	LCD bias voltages	—
LCD charge pump capacitance	V_{cap1}, V_{cap2}	Charge pump capacitor pins	—

24.2.1 LCD[43:0]

When LCD functionality is enabled by the PEN[43:0] bits in the LCDPEN registers, the corresponding LCD[43:0] pin will generate a frontplane or backplane waveform depending on the configuration of the backplane-enable bit field (BPEN[43:0]).

24.2.2 V_{LL1}, V_{LL2}, V_{LL3}

V_{LL1}, V_{LL2}, and V_{LL3} are bias voltages for the LCD module driver waveforms which can be internally generated using the internal charge pump (when enabled). The charge pump can also be configured to accept V_{LL3} as an input and generate V_{LL1} and V_{LL2} for 3 V glass opperation. V_{LL3} should never be set to a voltage other than V_{DD}. Refer to VSUPPLY[1:0] bits explanation.

24.2.3 V_{cap1}, V_{cap2}

The charge pump capacitor is used to transfer charge from the input supply to the regulated output. Use a ceramic capacitor.

24.3 Memory Map and Register Definition

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.

24.3.1 LCD Control Register 0 (LCDC0)

	7	6	5	4	3	2	1	0
R W	LCDEN	SOURCE	LCLK2	LCLK1	LCLK0	DUTY2	DUTY1	DUTY0
Reset	0	0	0	0	0	0	1	1

= Unimplemented or Reserved

Figure 24-2. LCD Control Register 0 (LCDC0)

Read: anytime

Write: LCDEN anytime. Do not change SOURCE LCLCK OR DUTY while LCDEN = 1.

Table 24-5. LCDC0 Field Descriptions

Field	Description
7 LCDEN	**LCD Driver Enable** — LCDEN starts LCD-module-waveform generator. 0 All frontplane and backplane pins are disabled. The LCD module system is also disabled, and all LCD waveform generation clocks are stopped. V_{LL3} is connected to V_{DD} internally 1 LCD module driver system is enabled and frontplane and backplane waveforms are generated. All LCD pins enabled using the LCD pin enable register (LCDPEN[x]) will output an LCD module driver waveform.The backplane pins will output an LCD module driver backplane waveform based on the settings of DUTY[2:0]. Chargepump or resistor bias is enabled.
6 SOURCE	**LCD Clock Source Select** — The LCD module has two possible clock sources. This bit is used to select which clock source is the basis for LCDCLK. 0 Selects the (external clock reference) as the LCD clock source. 1 Selects the alternate clock as the LCD clock source.
5:3 LCLK[2:0]	**LCD Clock Prescaler** — Used as a clock divider to generate the LCD module frame frequency as shown in Equation 24-1. LCD-module-duty-cycle configuration is used to determine the LCD module frame frequency. LCD module frame frequency calculations are provided in Table 24-15./-21. ***Eqn. 24-1*** $\text{LCD Module Frame Frequency} = \frac{\text{LCDCLK}}{((\text{DUTY}+1) \times 8 \times (4 + \text{LCLK[2:0]}) \times Y)}$ where 30< LCDCLK < 39.063 kHz where Y = 2,2,3,3,4,5,8,16 chosen by module duty cycle configuration
2:0 DUTY[2:0]	**LCD Duty Select** — DUTY[2:0] bits select the duty cycle of the LCD module driver. 000 Use 1 BP (1/1 duty cycle). 001 Use 2 BP (1/2 duty cycle). 010 Use 3 BP (1/3 duty cycle). 011 Use 4 BP (1/4 duty cycle). (Default) 100 Use 5 BP (1/5 duty cycle). 101 Use 6 BP (1/6 duty cycle). 110 Use 7 BP (1/7 duty cycle). 111 Use 8 BP (1/8 duty cycle).

24.3.2 LCD Control Register 1 (LCDC1)

	7	6	5	4	3	2	1	0
R	LCDIEN	0	0	0	0	FCDEN	LCDWAI	LCDSTP
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 24-3. LCD Control Register 1 (LCDC1)

Read: anytimeWrite: anytime

Table 24-6. LCDC1 Field Descriptions

Field	Description
7 LCDIEN	**LCD Module Frame Frequency Interrupt Enable** — Enables an LCD interrupt event that coincides with the LCD module frame frequency. 0 No interrupt request is generated by this event. 1 The start of the LCD module frame causes an LCD module frame frequency interrupt request.
2 FCDEN	**Full Complementary Drive Enable** — This bit allows digital functionality that are shared with LCD pins to operate as full complementary if the other conditions necessary have been met. The other conditions are: VSUPPLY = 11 and RVEN = 0. 0 GPIO shared with LCD operate as open drain outputs, input levels and internal pullup resistors are referenced to V_{DD}. 1 If VSUPPLY = 11 and RVEN = 0, GPIO shared with LCD operate as full complementary outputs. Input levels and internal pullup resistors are referenced to V_{LL3}.
1 LCDWAI	**LCD Module Driver and Charge Pump Stop While in Wait Mode** 0 Allows the LCD driver and charge pump to continue running during wait mode. 1 Disables the LCD driver and charge pump when MCU goes into wait mode.
0 LCDSTP	**LCD Module Driver and Charge Pump Stop While in Stop2 or Stop3 Mode** 0 Allows LCD module driver and charge pump to continue running during stop2 or stop3. 1 Disables LCD module driver and charge pump when MCU goes into stop2 or stop3.

24.3.3 LCD Voltage Supply Register (LCDSUPPLY)

	7	6	5	4	3	2	1	0
R	CPSEL	HREFSEL	LADJ1	LADJ0	0	BBYPASS	VSUPPLY1	VSUPPLY0
W								
Reset	0	0	1	1	0	1	0	1

Unimplemented or Reserved

Figure 24-4. LCD Voltage Supply Register (LCDSUPPLY)

Read: anytime

Write: anytime.

For proper operation, do not modify VSUPPLY[1:0] while the LCDEN bit is asserted. VSUPPLY[1:0] must also be configured according to the external hardware power supply configuration.

Table 24-7. LCDSUPPLY Field Descriptions

Field	Description
7 CPSEL	**Charge Pump or Resistor Bias Select** — Selects LCD module charge pump or a resistor network to supply the LCD voltages V_{LL1}, V_{LL2}, and V_{LL3}. See Figure 24-15 for more detail. 0 LCD charge pump is disabled. Resistor network selected (The internal 1/3-bias is forced.) 1 LCD charge pump is selected. Resistor network disabled (The internal 1/3-bias is forced.)
6 HREFSEL	**High Reference Select**— When using the V_{LCD} or V_{IREG} inputs, this bit configures internal circuits to supply V_{LL1}. 0 Divide input, V_{LCD} IN = V_{LCDEXT} * 2/3, V_{IREG} = 1.0V 1 Do not divide the input, V_{LCD} IN = V_{LCDEXT} * 3/3, V_{IREG} = 1.67 V
5:4 LADJ[1:0]	**LCD Module Load Adjust —** The LCD load adjust bits are used to configure the LCD module to handle different LCD glass capacitance. **For CPSEL = 1** **Adjust the clock source for the charge pump.** Higher loads require higher charge pump clock rates. 00 - Fastest clock source for charge pump (LCD glass capacitance 8000pf or lower) 01 - Intermediate clock source for charge pump (LCD glass capacitance 6000pf or lower)) 10 - Intermediate clock source for charge pump (LCD glass capacitance 4000pf or lower) 11 - Slowest clock source for charge pump (LCD glass capacitance 2000pf or lower) For CPSEL = 0 Adjust the resistor bias network for different LCD glass capacitance 00 - Low Load (LCD glass capacitance 2000pf or lower) 01 - Low Load (LCD glass capacitance 2000pf or lower) 10 - High Load (LCD glass capacitance 8000pf or lower) 11 - High Load (LCD glass capacitance 8000pf or lower)
2 BBYPASS	**Op Amp Control** — Determines whether the internal LCD op amp buffer is bypassed. 0 Buffered mode 1 Unbuffered mode
1:0 VSUPPLY[1:0]	**Voltage Supply Control** — Configures whether the LCD module power supply is external or internal. Avoid modifying this bit field while the LCD module is enabled (e.g., LCDEN = 1). See Figure 24-15 for more detail. 00 Drive V_{LL2} internally from V_{DD} 01 Drive V_{LL3} internally from V_{DD} 10 11 Drive V_{LL3} externally

24.3.4 LCD Regulated Voltage Control Register (LCDRVC)

	7	6	5	4	3	2	1	0
R	RVEN	0	0	0	RVTRIM3	RVTRIM2	RVTRIM1	RVTRIM0
W								
Reset	0	0	0	0	1	0	0	0

Unimplemented or Reserved

Figure 24-5. LCD Regulated Voltage Control Register (LCDRVC)

Read: anytime.

Write: anytime.

The regulated voltage can be used to generate a reference signal to the LCD charge pump for 3V or 5V LCD operation dependant on the HREFSEL bit.

Table 24-8. LCDRVC Field Descriptions

Field	Description
7 RVEN	**Regulated Voltage Enable** — Enables internal voltage regulator, Must have charge pump enabled. 0 Regulated voltage disabled. 1 Regulated voltage enabled.
3:0 RVTRIM[3:0]	**Regulated Voltage Trim**—This 4 bit trim register is used to adjust the regulated input. Each bit in the register has equal weight. The Regulated input is changed by 1.5% for each count.

24.3.5 LCD Blink Control Register (LCDBCTL)

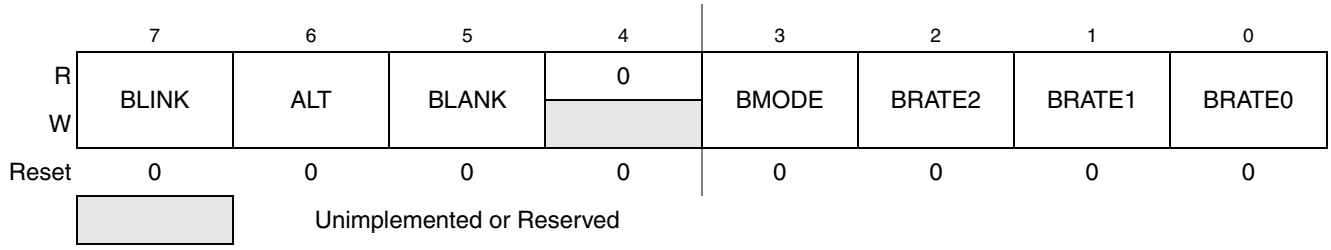


Figure 24-6. LCD Blink Control Register (LCDBCTL)

Read: anytime

Write: anytime

Table 24-9. LCDBCTL Field Descriptions

Field	Description
7 BLINK	**Blink Command** — Starts or stops LCD module blinking 0 Disables blinking 1 Starts blinking at blinking frequency specified by LCD blink rate calculation (see Equation 24-2)
6 ALT	**Alternate Display Mode** — For four backplanes or less the LCD backplane sequencer changes to output an alternate display. ALT bit is ignored if Duty is 5 or greater. 0 Normal Display 1 Alternate display mode
5 BLANK	Blank Display Mode — Asserting this bit clears all segments in the LCD display. 0 Normal or Alternate Display 1 Blank Display Mode

Table 24-9. LCDBCTL Field Descriptions (continued)

Field	Description
3 BMODE	**Blink Mode** — Selects the blink mode displayed during the blink period. See Table 24-9 for more information on how BMODE affects the LCD display. 0 Display blank during the blink period 1 Display alternate display during blink period (Ignored if duty is 5 or greater)
2:0 BRATE[2:0]	**Blink-Rate Configuration**— Selects frequency at which the LCD display blinks when the BLINK is asserted. Equation 24-2 shows how BRATE[2:0] bit field is used in the LCD blink-rate calculation. Equation 24-2 provides an expression for the LCD module blink rate ***Eqn. 24-2*** $\text{LCD module blink rate} = \frac{\text{LCDCLK}}{2^{(12+\text{ BRATE[2:0]})}}$ LCD module blink rate calculations are provided in 24.4.3.2/-28.

24.3.6 LCD Status Register (LCDS)

	7	6	5	4	3	2	1	0
R	LCDIF	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

Unimplemented or Reserved

Figure 24-7. LCD Status Register (LCDS)

Read: anytime

Write: anytime

Table 24-10. LCDS Field Descriptions

Field	Description
7 LCDIF	**LCD Interrupt Flag** — LCDIF indicates an interrupt condition occurred. To clear the interrupt write a 1 to LCDIF. 0 interrupt condition has not occurred. 1 interrupt condition has occurred.

24.3.7 LCD Pin Enable Registers 0–7 (LCDPEN0–LCDPEN7)

When LCDEN = 1, these bits enable the corresponding LCD pin for LCD operation.

These registers should only be written with instructions that perform byte writes, using instructions that perform word writes will lead to invalid data being placed in the register. Initialize these registers before enabling the LCD module. Exiting stop2 mode does not require reinitializing the LCDPEN registers.

		7	6	5	4	3	2	1	0
LCDPEN0	R W	PEN7	PEN6	PEN5	PEN4	PEN3	PEN2	PEN1	PEN0
	Reset	Indeterminate after reset							
LCDPEN1	R W	PEN15	PEN14	PEN13	PEN12	PEN11	PEN10	PEN9	PEN8
	Reset	Indeterminate after reset							
LCDPEN2	R W	PEN23	PEN22	PEN21	PEN20	PEN19	PEN18	PEN17	PEN16
	Reset	Indeterminate after reset							
LCDPEN3	R W	PEN31	PEN30	PEN29	PEN28	PEN27	PEN26	PEN25	PEN24
	Reset	Indeterminate after reset							
LCDPEN4	R W	PEN39	PEN38	PEN37	PEN36	PEN35	PEN34	PEN33	PEN32
	Reset	Indeterminate after reset							
LCDPEN5	R W					PEN43	PEN42	PEN41	PEN40
	Reset	Indeterminate after reset							

Unimplemented or Reserved

Figure 24-8. LCD Pin Enable Registers 0–7 (LCDPEN0–LCDPEN7)

Read: anytime

Write: anytime

Table 24-11. LCDPEN0–LCDPEN7 Field Descriptions

Field	Description
PEN[43:0]	**LCD Pin Enable** — The PEN[43:0] bit enables the LCD[43:0] pin for LCD operation. Each LCD[43:0] pin can be configured as a backplane or a frontplane based on the corresponding BPEN[n] bit in the Backplane Enable Register (LCDBPEN[7:0]). If LCDEN = 0, these bits have no effect on the state of the I/O pins. Set PEN[63:0] bits before LCDEN is set. 0 LCD operation disabled on LCDnn. 1 LCD operation enabled on LCDnn.

24.3.8 Backplane Enable Registers 0–7 (BPEN0–BPEN7)

When LCDPEN[n] = 1, these bits configure the corresponding LCD pin to operate as an LCD backplane or an LCD frontplane. Most applications set a maximum of eight of these bits. Initialize these registers before enabling the LCD module. Exiting stop2 mode does not require reinitializing the LCDBPEN registers.

These registers should only be written with instructions that perform byte writes, using instructions that perform word writes will lead to invalid data being placed in the register.

		7	6	5	4	3	2	1	0
LCDBPEN0	R W	BPEN7	BPEN6	BPEN5	BPEN4	BPEN3	BPEN2	BPEN1	BPEN0
	Reset	Indeterminate after reset							
LCDBPEN1	R W	BPEN15	BPEN14	BPEN13	BPEN12	BPEN11	BPEN10	BPEN9	BPEN8
	Reset	Indeterminate after reset							
LCDBPEN2	R W	BPEN23	BPEN22	BPEN21	BPEN20	BPEN19	BPEN18	BPEN17	BPEN16
	Reset	Indeterminate after reset							
LCDBPEN3	R W	BPEN31	BPEN30	BPEN29	BPEN28	BPEN27	BPEN26	BPEN25	BPEN24
	Reset	Indeterminate after reset							
LCDBPEN4	R W	BPEN39	BPEN38	BPEN37	BPEN36	BPEN35	BPEN34	BPEN33	BPEN32
	Reset	Indeterminate after reset							
LCDBPEN5	R W					BPEN43	BPEN42	BPEN41	BPEN40
	Reset	Indeterminate after reset							

Unimplemented or Reserved

Figure 24-9. Backplane Enable Registers 0–7 (BPEN0–BPEN7)

Read: anytime

Write: anytime

Table 24-12. LCDBPEN0–LCDBPEN Field Descriptions

Field	Description
BPEN[43:0]	**Backplane Enable** — The BPEN[43:0] bit configures the LCD[43:0] pin to operate as an LCD backplane or LCD frontplane. If LCDEN = 0, these bits have no effect on the state of the I/O pins. It is recommended to set BPEN[63:0] bits before LCDEN is set. 0 Frontplane operation enabled on LCD[n]. 1 Backplane operation enabled on LCD[n].

24.3.9 LCD Waveform Registers (LCDWF[43:0])

Each frontplane segment is associated with a backplane phase (A-H). For an LCD pin configured as a frontplante the LCDWF registers control the on/off state for frontplane segments.

For an LCD pin configured as a backplane the LCDWF registers controls the phase (A-H) in which the associated backplane pin is active.

These registers should only be written with instructions that perform byte writes, using instructions that perform word writes will lead to invalid data being placed in the register.

After reset, the LCDWF contents are indeterminate as indicated by Figure 24-10. Exiting stop2 mode does not require reinitializing the LCDWF registers.

		7	6	5	4	3	2	1	0
LCDWF0	R W	BPHLCD0	BPGLCD0	BPFLCD0	BPELCD0	BPDLCD0	BPCLCD0	BPBLCD0	BPALCD0
	Reset	Indeterminate after reset							
LCDWF1	R W	BPHLCD1	BPGLCD1	BPFLCD1	BPELCD1	BPDLCD1	BPCLCD1	BPBLCD1	BPALCD1
	Reset	Indeterminate after reset							
LCDWF2	R W	BPHLCD2	BPGLCD2	BPFLCD2	BPELCD2	BPDLCD2	BPCLCD2	BPBLCD2	BPALCD2
	Reset	Indeterminate after reset							
LCDWF3	R W	BPHLCD3	BPGLCD3	BPFLCD3	BPELCD3	BPDLCD3	BPCLCD3	BPBLCD3	BPALCD3
	Reset	Indeterminate after reset							
LCDWF4	R W	BPHLCD4	BPGLCD4	BPFLCD4	BPELCD4	BPDLCD4	BPCLCD4	BPBLCD4	BPALCD4
	Reset	Indeterminate after reset							
LCDWF5	R W	BPHLCD5	BPGLCD5	BPFLCD5	BPELCD5	BPDLCD5	BPCLCD5	BPBLCD5	BPALCD5
	Reset	Indeterminate after reset							
LCDWF6	R W	BPHLCD6	BPGLCD6	BPFLCD6	BPELCD6	BPDLCD6	BPCLCD6	BPBLCD6	BPALCD6
	Reset	Indeterminate after reset							
LCDWF7	R W	BPHLCD7	BPGLCD7	BPFLCD7	BPELCD7	BPDLCD7	BPCLCD7	BPBLCD7	BPALCD7
	Reset	Indeterminate after reset							
LCDWF8	R W	BPHLCD8	BPGLCD8	BPFLCD8	BPELCD8	BPDLCD8	BPCLCD8	BPBLCD8	BPALCD8
	Reset	Indeterminate after reset							
LCDWF9	R W	BPHLCD9	BPGLCD9	BPFLCD9	BPELCD9	BPDLCD9	BPCLCD9	BPBLCD9	BPALCD9
	Reset	Indeterminate after reset							
LCDWF10	R W	BPHLCD10	BPGLCD10	BPFLCD10	BPELCD10	BPDLCD10	BPCLCD10	BPBLCD10	BPALCD10
	Reset	Indeterminate after reset							
LCDWF11	R W	BPHLCD11	BPGLCD11	BPFLCD11	BPELCD11	BPDLCD11	BPCLCD11	BPBLCD11	BPALCD11
	Reset	Indeterminate after reset							
LCDWF12	R W	BPHLCD12	BPGLCD12	BPFLCD12	BPELCD12	BPDLCD12	BPCLCD12	BPBLCD12	BPALCD12
	Reset	Indeterminate after reset							

Figure 24-10. LCD Waveform Registers (LCDWF[:0])

		7	6	5	4	3	2	1	0
LCDWF13	R W	BPHLCD13	BPGLCD13	BPFLCD13	BPELCD13	BPDLCD13	BPCLCD13	BPBLCD13	BPALCD13
	Reset	Indeterminate after reset							
LCDWF14	R W	BPHLCD14	BPGLCD14	BPFLCD14	BPELCD14	BPDLCD14	BPCLCD14	BPBLCD14	BPALCD14
	Reset	Indeterminate after reset							
LCDWF15	R W	BPHLCD15	BPGLCD15	BPFLCD15	BPELCD15	BPDLCD15	BPCLCD15	BPBLCD15	BPALCD15
	Reset	Indeterminate after reset							
LCDWF16	R W	BPHLCD16	BPGLCD16	BPFLCD16	BPELCD16	BPDLCD16	BPCLCD16	BPBLCD16	BPALCD16
	Reset	Indeterminate after reset							
LCDWF17	R W	BPHLCD17	BPGLCD17	BPFLCD17	BPELCD17	BPDLCD17	BPCLCD17	BPBLCD17	BPALCD17
	Reset	Indeterminate after reset							
LCDWF18	R W	BPHLCD18	BPGLCD18	BPFLCD18	BPELCD18	BPDLCD18	BPCLCD18	BPBLCD18	BPALCD18
	Reset	Indeterminate after reset							
LCDWF19	R W	BPHLCD19	BPGLCD19	BPFLCD19	BPELCD19	BPDLCD19	BPCLCD19	BPBLCD19	BPALCD19
	Reset	Indeterminate after reset							
LCDWF20	R W	BPHLCD20	BPGLCD20	BPFLCD20	BPELCD20	BPDLCD20	BPCLCD20	BPBLCD20	BPALCD20
	Reset	Indeterminate after reset							
LCDWF21	R W	BPHLCD21	BPGLCD21	BPFLCD21	BPELCD21	BPDLCD21	BPCLCD21	BPBLCD21	BPALCD21
	Reset	Indeterminate after reset							
LCDWF22	R W	BPHLCD22	BPGLCD22	BPFLCD22	BPELCD22	BPDLCD22	BPCLCD22	BPBLCD22	BPALCD22
	Reset	Indeterminate after reset							
LCDWF23	R W	BPHLCD23	BPGLCD23	BPFLCD23	BPELCD23	BPDLCD23	BPCLCD23	BPBLCD23	BPALCD23
	Reset	Indeterminate after reset							
LCDWF24	R W	BPHLCD24	BPGLCD24	BPFLCD24	BPELCD24	BPDLCD24	BPCLCD24	BPBLCD24	BPALCD24
	Reset	Indeterminate after reset							
LCDWF25	R W	BPHLCD25	BPGLCD25	BPFLCD25	BPELCD25	BPDLCD25	BPCLCD25	BPBLCD25	BPALCD25
	Reset	Indeterminate after reset							

Figure 24-10. LCD Waveform Registers (LCDWF[:0]) (continued)

		7	6	5	4	3	2	1	0
LCDWF26	R W	BPHLCD26	BPGLCD26	BPFLCD26	BPELCD26	BPDLCD26	BPCLCD26	BPBLCD26	BPALCD26
	Reset	Indeterminate after reset							
LCDWF27	R W	BPHLCD27	BPGLCD27	BPFLCD27	BPELCD27	BPDLCD27	BPCLCD27	BPBLCD27	BPALCD27
	Reset	Indeterminate after reset							
LCDWF28	R W	BPHLCD28	BPGLCD28	BPFLCD28	BPELCD28	BPDLCD28	BPCLCD28	BPBLCD28	BPALCD28
	Reset	Indeterminate after reset							
LCDWF29	R W	BPHLCD29	BPGLCD29	BPFLCD29	BPELCD29	BPDLCD29	BPCLCD29	BPBLCD29	BPALCD29
	Reset	Indeterminate after reset							
LCDWF30	R W	BPHLCD30	BPGLCD30	BPFLCD30	BPELCD30	BPDLCD30	BPCLCD30	BPBLCD30	BPALCD30
	Reset	Indeterminate after reset							
LCDWF31	R W	BPHLCD31	BPGLCD31	BPFLCD31	BPELCD31	BPDLCD31	BPCLCD31	BPBLCD31	BPALCD31
	Reset	Indeterminate after reset							
LCDWF32	R W	BPHLCD32	BPGLCD32	BPFLCD32	BPELCD32	BPDLCD32	BPCLCD32	BPBLCD32	BPALCD32
	Reset	Indeterminate after reset							
LCDWF33	R W	BPHLCD33	BPGLCD33	BPFLCD33	BPELCD33	BPDLCD33	BPCLCD33	BPBLCD33	BPALCD33
	Reset	Indeterminate after reset							
LCDWF34	R W	BPHLCD34	BPGLCD34	BPFLCD34	BPELCD34	BPDLCD34	BPCLCD34	BPBLCD34	BPALCD34
	Reset	Indeterminate after reset							
LCDWF35	R W	BPHLCD35	BPGLCD35	BPFLCD35	BPELCD35	BPDLCD35	BPCLCD35	BPBLCD35	BPALCD35
	Reset	Indeterminate after reset							
LCDWF36	R W	BPHLCD36	BPGLCD36	BPFLCD36	BPELCD36	BPDLCD36	BPCLCD36	BPBLCD36	BPALCD36
	Reset	Indeterminate after reset							
LCDWF37	R W	BPHLCD37	BPGLCD37	BPFLCD37	BPELCD37	BPDLCD37	BPCLCD37	BPBLCD37	BPALCD37
	Reset	Indeterminate after reset							
LCDWF38	R W	BPHLCD38	BPGLCD38	BPFLCD38	BPELCD38	BPDLCD38	BPCLCD38	BPBLCD38	BPALCD38
	Reset	Indeterminate after reset							

Figure 24-10. LCD Waveform Registers (LCDWF[:0]) (continued)

		7	6	5	4	3	2	1	0
LCDWF39	R W	BPHLCD39	BPGLCD39	BPFLCD39	BPELCD39	BPDLCD39	BPCLCD39	BPBLCD39	BPALCD39
	Reset	Indeterminate after reset							
LCDWF40	R W	BPHLCD40	BPGLCD40	BPFLCD40	BPELCD40	BPDLCD40	BPCLCD40	BPBLCD40	BPALCD40
	Reset	Indeterminate after reset							
LCDWF41	R W	BPHLCD41	BPGLCD41	BPFLCD41	BPELCD41	BPDLCD41	BPCLCD41	BPBLCD41	BPALCD41
	Reset	Indeterminate after reset							
LCDWF42	R W	BPHLCD42	BPGLCD42	BPFLCD42	BPELCD42	BPDLCD42	BPCLCD42	BPBLCD42	BPALCD42
	Reset	Indeterminate after reset							
LCDWF43	R W	BPHLCD43	BPGLCD43	BPFLCD43	BPELCD43	BPDLCD43	BPCLCD43	BPBLCD43	BPALCD43
	Reset	Indeterminate after reset							

Figure 24-10. LCD Waveform Registers (LCDWF[:0]) (continued)

Table 24-13. LCDWF Field Descriptions

Field	Description
BP[y]LCD[x]	**Segment-on-Frontplane Operation** — If the LCD[x] pin is enabled and configured to operate as a frontplane, the BP[y]LCD[x] bit in the LCDWF registers controls the on/off state for the LCD segment connected between LCD[x] and BP[y].BP[y] corresponds to an LCD[:0] pin enabled and configured to operate as a backplane that is active in phase [y]. Asserting BP[y]LCD[x] displays (turns on) the LCD segment connected between LCD[x] and BP[y]. 0 LCD segment off 1 LCD segment on **Segment-on-Backplane Operation** — If the LCD[x] pin is enabled and configured to operate as a backplane, the BP[y] LCD[x] bit in the LCDWF registers controls the phase (A-H) in which the LCD[x] pin is active.Backplane phase assignment is done using this method. 0 LCD backplane inactive for phase[y] 1 LCD backplane active for phase[y].

24.4 Functional Description

This section provides a complete functional description of the <<BLOCK NAME>> block, detailing the operation of the design from the end-user perspective.

Before enabling the LCD module by asserting the LCDEN bit in the LCDC0 register, configure the LCD module based on the end application requirements. Out of reset, the LCD module is configured with default settings, but these settings are not optimal for every application. The LCD module provides several versatile configuration settings and options to support varied implementation requirements, including:

- Frame frequency

- Duty cycle (number of backplanes)
- Backplane assignment (which LCD[43:0] pins operate as backplanes)
- Frame frequency interrupt enable
- Blinking frequency and options
- Power-supply configurations

The LCD module also provides an LCD pin enable control. Setting the LCD pin enable bit (PEN[x] in the LCDPEN[y] register) for a particular LCD[y] pin enables the LCD module functionality of that pin once the LCDEN bit is set. When the BPEN[x] bit in the LCDBPEN[y] is set, the associated pin operates as a backplane. The LCDWF registers can then activate (display) the corresponding LCD segments on an LCD panel.

The LCDWF registers control the on/off state for the segments controlled by the LCD pins defined as front planes and the active phase for the backplanes. Blank display modes do not use the data from the LCDWF registers. When using the LCDWF register for frontplane operation, writing a 0 turns the segment off.

For pins enabled as backplane, the phase of the backplane (A-H) is assigned by the LCDWF register for the corresponding backplane pin.

24.4.1 LCD Driver Description

The LCD module driver has 8 modes of operation:

- 1/1 duty (1 backplane) (Phase A), 1/3 bias (4 voltage levels)
- 1/2 duty (2 backplanes) (Phase A, B), 1/3 bias (4 voltage levels)
- 1/3 duty (3 backplanes) (Phase A, B, C), 1/3 bias (4 voltage levels)
- 1/4 duty (4 backplanes) (Phase A, B, C, D), 1/3 bias (4 voltage levels)
- 1/5 duty (5 backplanes) (Phase A, B, C, D, E), 1/3 bias (4 voltage levels)
- 1/6 duty (6 backplanes) (Phase A, B, C, D, E, F), 1/3 bias (4 voltage levels)
- 1/7 duty (7 backplanes) (Phase A, B, C, D, E, F, G), 1/3 bias (4 voltage levels)
- 1/8 duty (8 backplanes) (Phase A, B, C, D, E, F, G, H), 1/3 bias (4 voltage levels)

All modes are 1/3 bias. These modes of operation are described in more detail in the following sections.

24.4.1.1 LCD Duty Cycle

The denominator of the duty cycle indicates the number of LCD panel segments capable of being driven by each individual frontplane output driver. Depending on the duty cycle, the LCD waveform drive can be categorized as static or multiplexed.

In static-driving method, the LCD is driven with two square waveforms. The static-driving method is the most basic method to drive an LCD panel, but because each frontplane driver can drive only one LCD segment, static driving limits the LCD segments that can be driven with a given number of frontplane pins. In static mode, only one backplane is required.

In multiplexed mode, the LCD waveforms are multi-level and depend on the bias mode. Multiplex mode, depending on the number of backplanes, can drive multiple LCD segments with a single frontplane driver.

This reduces the number of driver circuits and connections to LCD segments. For multiplex mode operation, at least two backplane drivers are needed. The LCD module is optimized for multiplex mode.

The duty cycle indicates the amount of time the LCD panel segment is energized during each LCD module frame cycle. The denominator of the duty cycle indicates the number of backplanes that are being used to drive an LCD panel.

The duty cycle is used by the backplane phase generator to set the phase outputs. The phase outputs A-H are driven according to the sequence shown below. The sequence is repeated at the LCD frame frequency. The duty cycle is configured using the DUTY[2:0] bit field in the LCDC0 register, as shown in Table 24-14.

Table 24-14. LCD Module Duty Cycle Modes

Duty	LCDC0 Register			Number of Backplanes	Phase Sequence
	DUTY2	DUTY1	DUTY0		
1/1	0	0	0	1	A
1/2	0	0	1	2	A B
1/3	0	1	0	3	A B C
1/4	0	1	1	4	A B C D
1/5	1	0	0	5	A B C D E
1/6	1	0	1	6	A B C D E F
1/7	1	1	0	7	A B C D E F G
1/8	1	1	1	8	A B C D E F G H

24.4.1.2 LCD Bias

Because a single frontplane driver is configured to drive more and more individual LCD segments, 3 voltage levels are required to generate the appropriate waveforms to drive the segment. The LCD module is designed to operate using the 1/3 bias mode.

24.4.1.3 LCD Module Base Clock and Frame Frequency

The LCD module is optimized to operate using a 32.768 kHz clock input. Two clock sources are available to the LCD module, which are selectable by configuring the SOURCE bit in the LCDC0 register. The two clock sources include:

- External crystal —OSCOUT (SOURCE = 0)
- Alternate clock (SOURCE = 1)

Figure 24-11 shows the LCD clock tree. The clock tree shows the two possible clock sources and the LCD frame frequency and blink frequency clock source. The LCD blink frequency is discussed in Section 24.4.3.2, “Blink Frequency.”

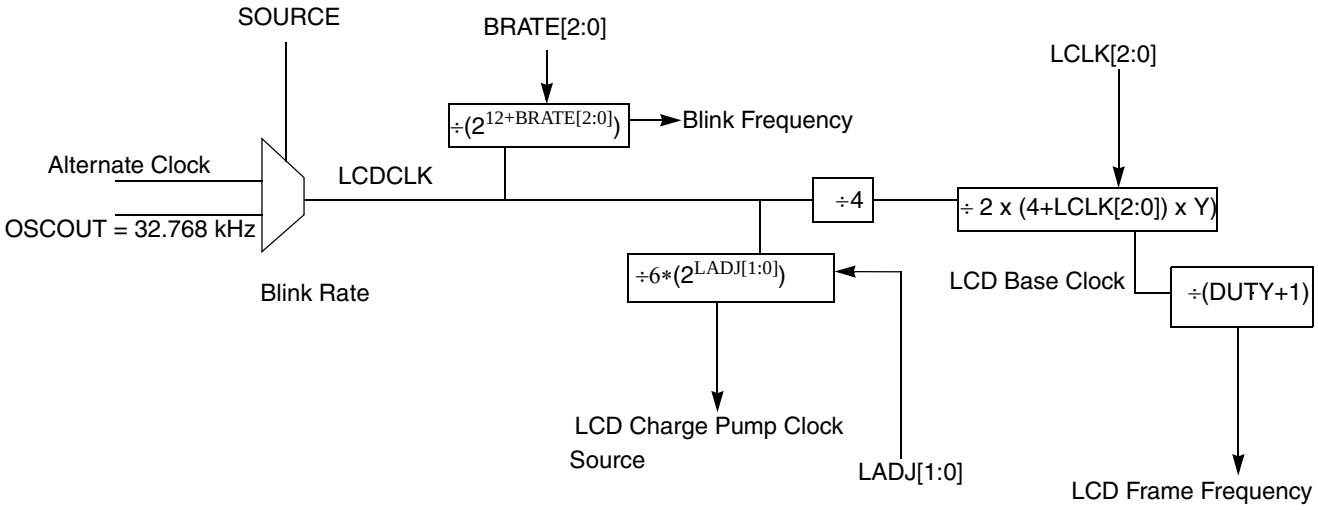


Figure 24-11. LCD Clock Tree

An external 32.768 kHz clock input is required to achieve lowest power consumption.

The value of LCDCLK is important because it is used to generate the LCD module frame frequency. Equation 24-1 provides an expression for the LCD module frame frequency calculation.

The LCD module frame frequency is a function of the LCD module duty cycle as shown in Equation 24-1. Table 24-16 and Table 24-15 show LCD module frame frequency calculations that consider several possible LCD module configurations of LCLK[2:0] and DUTY[2:0].

The LCD module frame frequency is defined as the number of times the LCD segments are energized per second. The LCD module frame frequency must be selected to prevent the LCD display from flickering (LCD module frame frequency is too low) or ghosting (LCD module frame frequency is too high). To avoid these issues, an LCD module frame frequency in the range of 28 to 58 Hz is required. LCD module frame frequencies less than 28 Hz or greater than 58 Hz are out of specification, and so are invalid. Selecting lower values for the LCD base and frame frequency results in lower current consumption for the LCD module.

The LCD module base clock frequency is the LCD module frame frequency multiplied by the number of backplane phases that are being generated. The number of backplane phases is selected using the DUTY[2:0] bits. The LCD module base clock is used by the backplane sequencer to generate the LCD waveform data for the enabled phases (A-H).

Table 24-15. LCD Module Frame Frequency Calculations[1]

Duty Cycle	1/1	1/2	1/3	1/4	1/5	1/6	1/7	1/8
Y	16	8	5	4	3	3	2	2
LCLK[2:0]								
0	64	64	68.3	64	68.3	56.9	73.1	64
1	51.2	51.2	54.6	51.2	54.6	45.5	58.5	51.2
2	42.7	42.7	45.5	42.7	45.5	37.9	48.8	42.7
3	36.6	36.6	39	36.6	39	32.5	41.8	36.6
4	32	32	34.1	32	34.1	28.4	36.6	32
5	28.4	28.4	30.3	28.4	30.3	25.3	32.5	28.4
6	25.6	25.6	27.3	25.6	27.3	22.8	29.3	25.6
7	23.3	23.3	24.8	23.3	24.8	20.7	26.6	23.3

1 LCD clock input ~ 32.768 kHz

Shaded table entries are out of specification and invalid.

Table 24-16. LCD Module Frame Frequency Calculations[1]

Duty Cycle	1/1	1/2	1/3	1/4	1/5	1/6	1/7	1/8
Y	16	8	5	4	3	3	2	2
LCLK[2:0]								
0	76.3	76.3	81.4	76.3	81.4	67.8	87.2	76.3
1	61	61	65.1	61	65.1	54.3	69.8	61
2	50.9	50.9	54.3	50.9	54.3	45.2	58.1	50.9
3	43.6	43.6	46.5	43.6	46.5	38.8	49.8	43.6
4	38.1	38.1	40.7	38.1	40.7	33.9	43.6	38.1
5	33.9	33.9	36.2	33.9	36.2	30.1	38.8	33.9
6	30.5	30.5	32.6	30.5	32.6	27.1	34.9	30.5
7	27.7	27.7	29.6	27.7	29.6	24.7	31.7	27.7

1 LCD clock input ~ 39.063 kHz

Shaded table entries are out of specification and invalid.

24.4.1.4 LCD Waveform Examples

This section shows the timing examples of the LCD output waveforms for the several modes of operation. As shown in Table 24-17, all examples use 1/3 bias mode.

Table 24-17. Configurations for Example LCD Waveforms

	Bias Mode	DUTY[2:0]	Duty Cycle
Example 1	1/3	001	1/2
Example 2		011	1/4
Example 3		111	1/8

24.4.1.4.1 1/2 Duty Multiplexed with 1/3 Bias Mode (Low-power Waveform)

Duty=1/2:DUTY[2:0] = 001

LCD pin 0 (LCD[0])and LCD pin 1, LCD[1] enabled as backplanes:

BPEN0 =1 and BPEN1 =1 in the LCDBPEN0

LCD[0] assigned to Phase A: LCDWF0 = 0x01

LCD[1] assigned to Phase B: LCDWF1 = 0x02

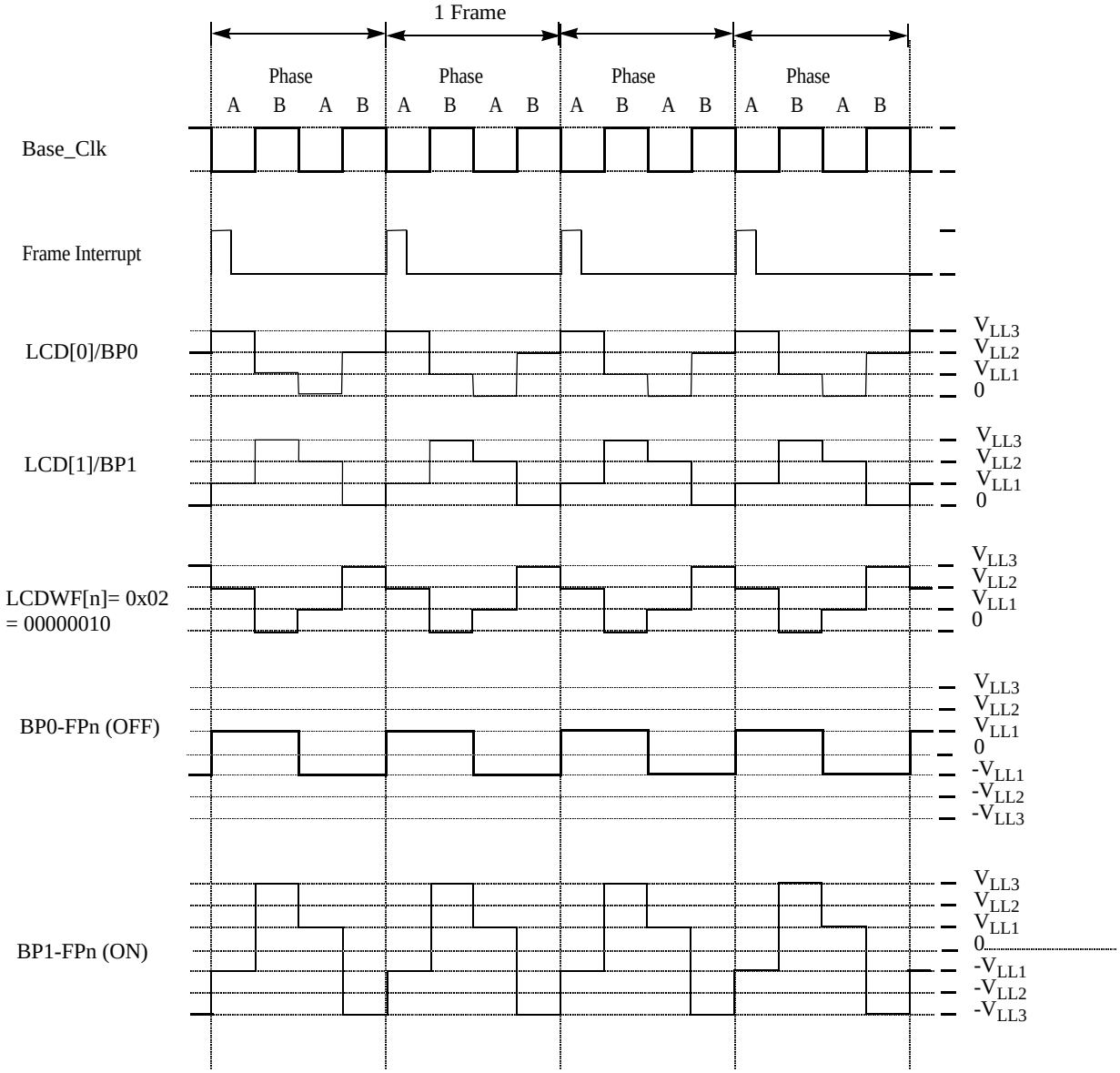


Figure 24-12. 1/2 Duty and 1/3 Bias (Low-Power Waveform)

24.4.1.4.2 1/4 Duty Multiplexed with 1/3 Bias Mode (Low-power Waveform)

Duty = 1/4: DUTY[2:0] = 011

LCD pins 0 – 3 enabled as backplanes: LCDBPEN0 = 0x0F

LCD[0] assigned to Phase A: LCDWF0 = 0x01

LCD[1] assigned to Phase B: LCDWF1 = 0x02

LCD[2] assigned to Phase C: LCDWF2 = 0x04

LCD[3] assigned to Phase D: LCDWF3 = 0x08

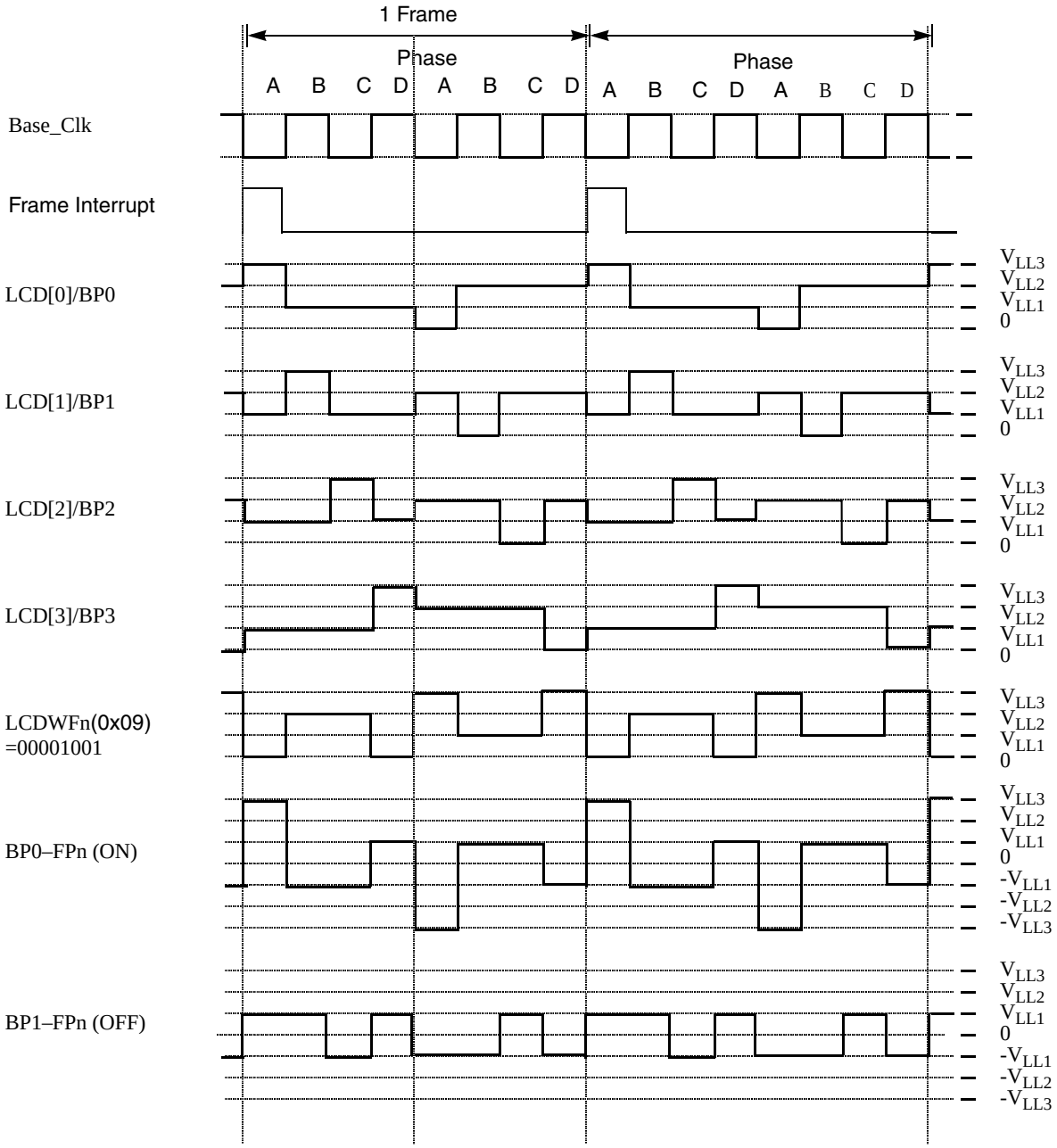


Figure 24-13. 1/4 Duty and 1/3 Bias (Low-Power Waveform)

24.4.1.4.3 1/8 Duty Multiplexed with 1/3 Bias Mode (Low-power Waveform)

Duty = 1/8:DUTY[2:0] = 111

LCD pins 0 – 7 enabled as backplanes: LCDBPEN0 = 0xFF

LCD[0] assigned to Phase A: LCDWF0 = 0x01

LCD[1] assigned to Phase B: LCDWF1 = 0x02

LCD[2] assigned to Phase C: LCDWF2 = 0x04

LCD[3] assigned to Phase D: LCDWF3 = 0x08

LCD[4] assigned to Phase E: LCDWF4 = 0x10

LCD[5] assigned to Phase F: LCDWF5 = 0x20

LCD[6] assigned to Phase G: LCDWF6 = 0x40

LCD[7] assigned to Phase H: LCDWF7 = 0x80

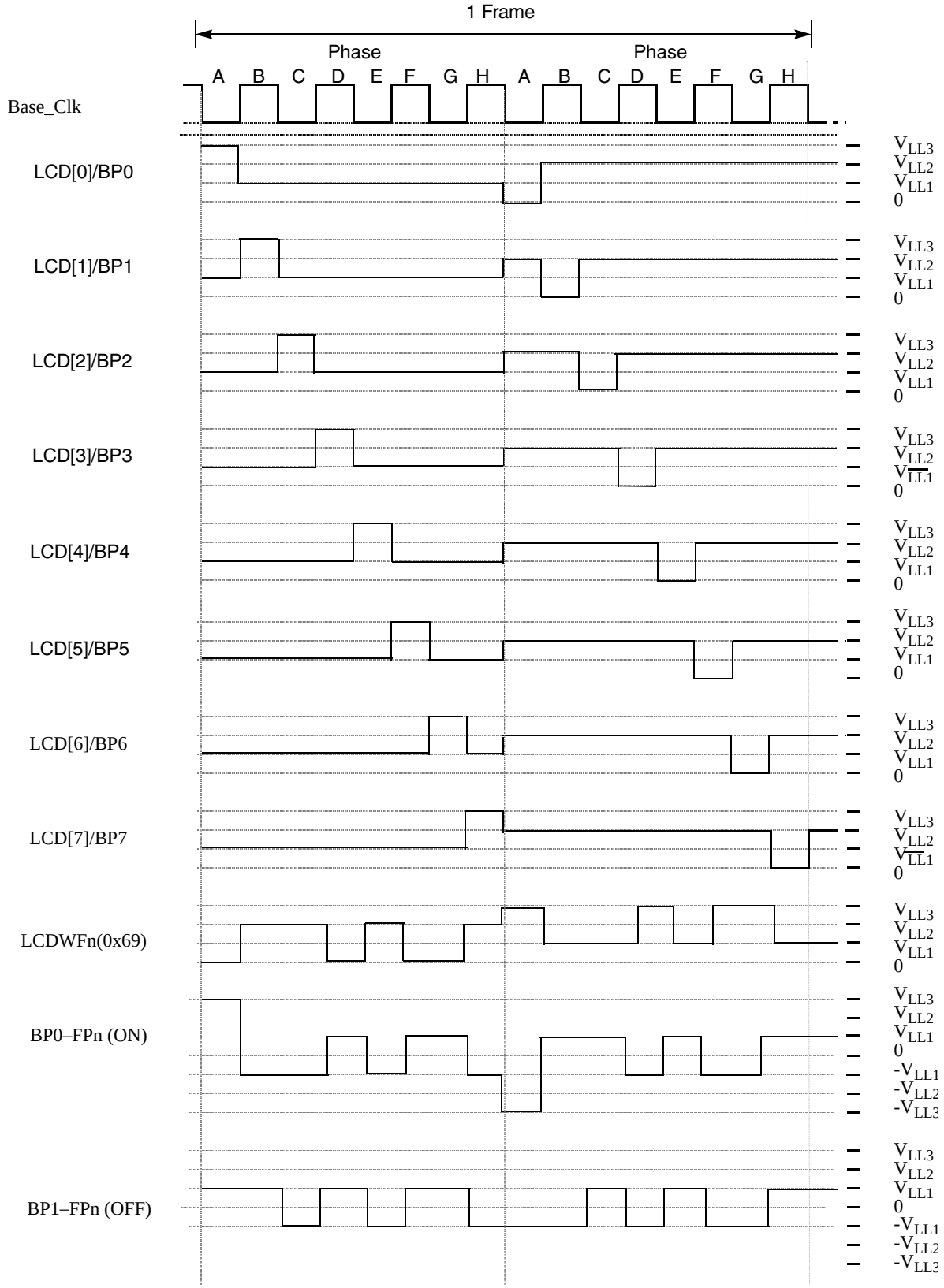


Figure 24-14. 1/8 Duty and 1/3 Bias (Low-power Waveform)

24.4.2 LCDWF Registers

For a segment on the LCD panel to be displayed, data must be written to the LCDWF registers. For LCD pins enabled as frontplanes, each bit in the LCDWF registers corresponds to a segment on an LCD panel. The different phases A-H represent the different backplanes of the LCD panel. The selected LCD duty cycle controls the number of implemented phases. Refer to Table 24-14 for normal LCD operation the phases follow the sequence shown.

For LCD pins enabled as a backplane, the LCDWF assigns the phase in which the backplane pin is active. This is how backplane assignment is done.

An example of normal operation follows: enable LCD pin 0 to operate as backplane 0. Enable the LCD pin 0 by setting PEN0 bit in the LCDPEN0 register. Configure LCD pin 0 as a backplane pin by setting the BPEN0 bit in the LCDBPEN0 register. Finally, the BPALCD0 bit in the LCDWF0 is set to associate LCD pin 0 with backplane phase A.This will configure LCD0 to operate as a backplane that is active in Phase A.

For LCD pins enabled as a frontplane, writing a 1 to a given LCDWF location results in the corresponding display segment being driven with the differential root mean square (RMS) voltage necessary to turn the segment on during the phase selected. Writing a 0 to a given location results in the corresponding display segment being driven with the differential RMS voltage necessary to turn the segment off during the phase selected.

24.4.3 LCD Display Modes

The LCD module can be configured to implement several different display modes. The bits ALT and BLANK in the LCD-blink-control register (LCDBCTL) configure the different display modes. In normal display mode (default), LCD segments are controlled by the data placed in the LCDWF registers, as described in Section 24.4.2, “LCDWF Registers.” For blank-display mode, the LCDWF data is bypassed and the frontplane and backplane pins are configured to clear all segments.

For alternate-display mode, the backplane sequence is modified for duty cycles of 1/4, 1/3, 1/2, and 1/1. For four backplanes or less, the backplane sequence is modified as shown below. The altered sequence allows two complete displays to be placed in the LDCDWF registers. The first display is placed in phases A-D and the second in phases E-H in the case of four backplanes. If the LCD duty cycle is five backplanes or greater, the ALT bit is ignored and creates a blank display. Refer to Table 24-19 for additional information.

Using the alternate display function an inverse display can be accomplished for x4 mode and less by placing inverse data in the alternate phases of the LCDWF registers.

Table 24-18. Alternate Display Backplane Sequence

Duty	Backplane Sequence	Alt. Backplane Sequence
1/1	A	E
1/2	A B	E F

Table 24-18. Alternate Display Backplane Sequence (continued)

Duty	Backplane Sequence	Alt. Backplane Sequence
1/3	A B C	E F G
1/4	A B C D	E F G H

24.4.3.1 LCD Blink Modes

The blink mode is used as a means of alternating among different LCD display modes at a defined frequency. The LCD module can be configured to implement two blink modes. The BMODE bit in the LCD-blink-control register (LCDBCTL) configures the different blink modes. Blink modes are activated by setting the BLINK bit in the LCDBCTL register. If BLINK = 0, the LCD module operates normally as described Section 24.4.3, “LCD Display Modes”. If BLINK = 1, BMODE bit configures the blinking operation. During a blink, the display data driven by the LCD module changes to the mode selected by the BMODE bit. The BMODE bit selects two different blink modes, blank and alternate modes operate in the same way, as defined in Section 24.4.3, “LCD Display Modes.” The table below shows the interaction between display modes and blink modes. If the LCD duty cycle is five backplanes or greater, BMODE = 1 is ignored and will revert to create a blank display during the blink period.

Table 24-19. Display Mode Interaction

BLANK	ALT	BMODE	LCD Duty	BLINK = 1	
				Normal Period	Blink Period
0	0	0	1-4	Normal Display	Blank Display
0	0	1	1-4	Normal Display	Alternate display
0	1	0	1-4	Alternate display	Blank Display
0	1	1	1-4	Alternate Display	Alternate display
1	X	0	1-4	Blank Display	Blank DIsplay
1	X	1	1-4	Blank Display	Alternate display
0	X	X	5-8	Normal Display	Blank Display
1	X	X	5-8	Blank Display	Blank Display

24.4.3.2 Blink Frequency

The LCD clock is the basis for the calculation of the LCD module blink frequency. The LCD module blink frequency is equal to the LCD clock (LCDCLK) divided by the factor selected by the BRATE[2:0] bits. Table 24-20 shows LCD module blink frequency calculations for all values of BRATE[2:0] at a few common LCDCLK selections.

Table 24-20. Blink Frequency Calculations
(Blink Rate = LCD Clock(Hz) ÷ Blink Divider)

BRATE[2:0]	0	1	2	3	4	5	6	7
LCD Clock	**Blink Frequency (Hz)**							
30 khz	7.32	3.66	1.831	.916	.46	.23	.11	.06
32.768 khz	8	4	2	1	.5	.25	.13	.06
39.063 khz	9.54	4.77	2.38	1.19	.6	.30	.15	.075

24.4.4 LCD Charge Pump, Voltage Divider, and Power Supply Operation

This section describes the LCD charge pump, voltage divider, and LCD power supply configuration options. Figure 24-15 provides a block diagram for the LCD charge pump and the resistor divider network.

The LCD bias voltages (V_{LL1}, V_{LL2} and V_{LL3}) can be generated by the LCD charge pump or a resistor divider network that is connected using the CPSEL bit. The input source to the LCD charge pump is controlled by the VSUPPLY[1:0] bit field.

VSUPPLY[1:0] indicates the state of internal signals used to configure power switches as shown in the table in Figure 24-15. The block diagram in Figure 24-15 illustrates several potential operational modes for the LCD module including configuration of the LCD module power supply source using V_{DD},or an external supply on the V_{LL3} . V_{LL3} should never exceed V_{DD}.

Upon Reset the VSUPPLY[1:0] bits are configured to connect V_{LL3} to V_{DD}. This configuration should be changed to match the application requirements before the LCD module is enabled.

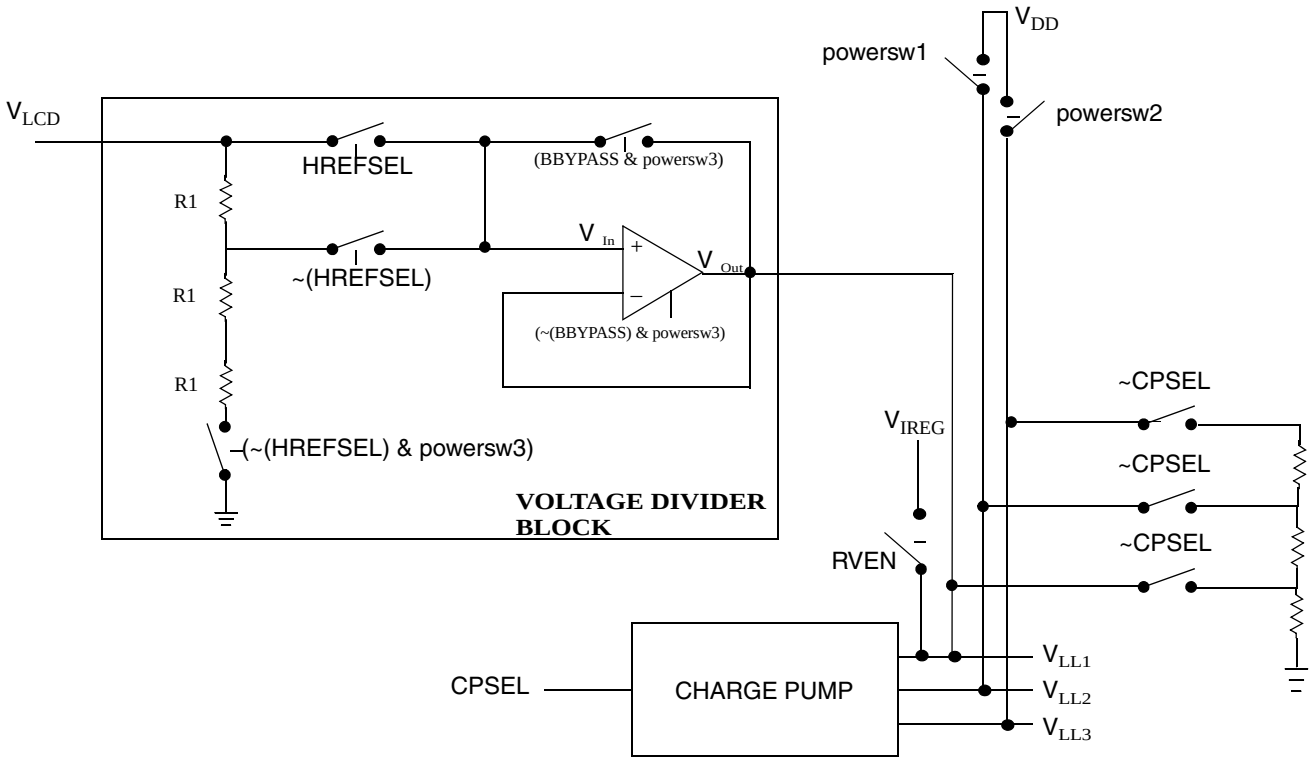


Table 24-21. LCD Power Supply Configuration

VSUPPLY[1:0]	Configuration	powersw1	powersw2	powersw3
00	Drive V_{LL2} internally from V_{DD}	1	0	0
01	Drive V_{LL3} internally from V_{DD}	0	1	0
10	Drive V_{LL1}internally from the V_{LCD} pin	0	0	1
11	Drive V_{LL3} externally from V_{DD} Or Drive V_{LL1} internally from V_{IREG}	0	0	0

Figure 24-15. LCD Charge Pump and V_{LCD} Voltage Divider Block Diagram

Figure 24-15 also illustrates a buffer, a voltage follower with an ideal op amp. The buffer, if enabled, gives $V_{In} = V_{Out}$, and, because the input impedance of the op amp is very high, V_{In} is isolated from V_{Out} This isolation can protect V_{In} from current draw from V_{Out}; however, if the buffer is disabled ((~BBYPASS & powersw3) = 0), the output and input will be configured in a tri-state condition; that is, they will be floating.

NOTE:

The charge pump is optimized for 1/3 bias mode operation only.

During the first 16 timebase clock cycles after the LCDCPEN bit is set, all the LCD frontplane and backplane outputs are disabled, regardless of the state of the LCDEN bit.

The charge pump requires external capacitance for its operation. To provide this external capacitance, the $V_{cap1\ and}$ V_{cap2} external pins are provided. It is recommended that a ceramic capacitor be used. Proper orientation is imperative when using a polarized capacitor. The recommended value for the external capacitor is 0.1 μF.

24.4.4.1 LCD Charge Pump and Voltage Divider

Using the voltage divider and charge pump, the LCD module can effectively double or triple the input voltage placed on the V_{LCD} pin.This LCD module configurability makes the LCD module compatible with both 3-V or 5-V LCD glass.

The LCD module high reference select bit (HREFSEL) in the LCDSUPPLY register configures the LCD module operational mode as a voltage doubler or a voltage tripler. In Figure 24-15, HREFSEL bit signal is used to control switches within the voltage divider block to enable or disable the two-thirds (2/3 * V_{LCD}) voltage divider. If HREFSEL = 0, the LCD module is configured as a voltage doubler, by enabling the voltage divider. With this configuration if V_{LCD} = 1.5 V the bias voltages generated will be:V_{LCD} * 2/3 = 1.5 V * 2/3 = 1 V = V_{LL1}

V_{LL1} * 2 = 2 V = V_{LL2}

V_{LL1} * 3 = 3 V = V_{LL3}

If HREFSEL = 1, the LCD module is configured as a voltage tripler by disabling the voltage divider. (V_{LL1} = V_{LCD}) The HREFSEL configuration depends on the LCD panel operating voltage specification in the application.

24.4.4.1.1 CPSEL: LCD Charge Pump or Resistor Bias Enable

The CPSEL bit in the LCDSUPPLY register selects the charge pump. When the charge pump is selected (CPSEL = 1), V_{LL1}, V_{LL2}, and V_{LL3} are generated internally.

When the charge pump is unselected (CPSEL = 0), V_{LL3} must be supplied and V_{LL1}, V_{LL2} are generated by a resistor bias network.

24.4.4.2 LCD Power Supplyand Voltage BufferConfiguration

The LCD bias voltages can be internally derived from V_{DD}, internally derived from a voltage source (must not exceed V_{DD}) connected to V_{LL3}, internally derived from a regulated voltage source that can be configured to supply 1.0 or 1.67 V (V_{IREG}), or it can be internally derived from a voltage source in the range between .9 to 1.8 V that is applied to the V_{LCD} pin. Table 24-22 provides a more detailed description of the power state of the LCD module which depends on the configuration of the VSUPPLY[1:0],HREFSEL, BBYPASS, CPSEL and RVEN bits.

Table 24-22 shows all possible configurations of the LCD Power Supply. All other combinations of the configuration bits above are not permissible LCD power supply modes and should be avoided.

Table 24-22. LCD Power Supply Configuration

LCD Operational State	LCD Power Supply Configuration	VSUPPLY[1:0]	HREFSEL	BBYPASS	CPSEL	RVEN
V_{LL2} connected to V_{DD} internally for 3 or 5 V glass operation.	For 3 V glass operation V_{DD} must equal 2 V. For 5 V glass operation V_{DD} must equal 3.33 V Charge pump is used to generate V_{LL1} and V_{LL3} V_{LCD} pin is not connected	00	X	X	1	0
V_{LL3} connected to V_{DD} internally for 3 V or 5 V glass operation	For 3 V glass operation V_{DD} must equal 3 V. For 5 V glass operation V_{DD} must equal 5 V. Charge pump is used to generate V_{LL1} and V_{LL2} V_{LCD} pin is not connected	01	X	X	1	0
V_{LCD} * 3/3 connected to V_{LL1} for 3 V glass operation. Unbuffered mode.	For 3 V glass operation V_{LCD} must equal 1 V. V_{LCD} (1 V) is connected to V_{LL1} HREFSEL = 1 to set $V_{LL1} = V_{LCD}$ * 3/3 Charge pump is used to generate V_{LL2}, and V_{LL3}	10	1	1	1	0
V_{LCD} * 2/3 connected to V_{LL1} for 3 V glass operation. Buffered mode.	For 3 V glass operation V_{LCD} must equal 1.5 V. 2/3 V_{LCD} (1 V) is connected to V_{LL1} HREFSEL = 0 to set $V_{LL1} = V_{LCD}$ * 2/3 Charge pump is used to generate V_{LL2}, and V_{LL3}	10	0	0	1	0
V_{LCD} * 3/3 connected to V_{LL1} for 3 V glass operation. Buffered mode.	For 3 V glass operation V_{LCD} must equal 1 V. V_{LCD} (1 V) is connected to V_{LL1} HREFSEL = 1 to set $V_{LL1} = V_{LCD}$ * 3/3 Charge pump is used to generate V_{LL2}, and V_{LL3}	10	1	0	1	0
V_{LCD} connected to V_{LL1} for 5 V glass operation. Unbuffered mode.	For 5 V glass operation V_{LCD} must equal 1.67 V. V_{LCD} (1.67 V) is connected to V_{LL1} HREFSEL = 1 to set $V_{LL1} = V_{LCD}$ * 3/3 Charge pump is used to generate V_{LL2}, and V_{LL3}	10	1	1	1	0

Table 24-22. LCD Power Supply Configuration

LCD Operational State	LCD Power Supply Configuration	VSUPPLY[1:0]	HREFSEL	BBYPASS	CPSEL	RVEN
V_{LCD} connected to V_{LL1} for 5 V glass operation. Buffered mode.	For 5 V glass operation V_{LCD} must equal 1.67 V. V_{LCD} (1.67 V) is connected to V_{LL1} HREFSEL = 1 to set $V_{LL1} = V_{LCD} * 3/3$ Charge pump is used to generate V_{LL2}, and V_{LL3}	10	1	0	1	0
V_{LL3} is driven externally for 3 V LCD Glass operation.	For 3 V glass operation V_{LL3} must equal 3 V. V_{LCD} is not connected Charge pump is used to generate V_{LL1} and V_{LL2}	11	X	X	1	0
V_{LL3} is driven externally for 5 V LCD Glass operation. V_{LL3} must equal V_{DD} This operation is not allowed for 1.8 V to 3.6 V parts	For 5 V glass operation V_{LL3} must equal 5 V. V_{LCD} is not connected Charge pump is used to generate V_{LL1} and V_{LL2}	11	X	X	1	0
V_{LL3} is driven externally for 3 V LCD Glass operation. Resistor Bias Network enabled.	For 3 V glass operation V_{LL3} must equal 3 V. V_{LCD} is not connected. Charge pump is disabled. Resistor Bias network is used to create V_{LL1} and V_{LL2}	11	X	X	0	0
V_{LL3} is driven externally for 5 V LCD Glass operation. Resistor Bias network enabled. V_{LL3} must equal V_{DD} This operation is not allowed for 1.8 V to 3.6 V parts	For 5 V glass operation V_{LL3} must equal 5 V. V_{LCD} is not connected. Charge pump is disabled. Resistor Bias network is used to create V_{LL1} and V_{LL2}.	11	X	X	0	0
V_{IREG} is connected to V_{LL1} for 5 V glass operation. The HREFSEL bit is used to select 1.0 or 1.67 V range for V_{IREG}	For 5 V glass operation HREFSEL = 1, V_{IREG} = 1.67 V. V_{IREG} is connected V_{LL1} internally. V_{LCD} is not connected. Charge pump is used to generate V_{LL2} and V_{LL3} .	11	1	X	1	1

Table 24-22. LCD Power Supply Configuration

LCD Operational State	LCD Power Supply Configuration	VSUPPLY[1:0]	HREFSEL	BBYPASS	CPSEL	RVEN
V_{IREG} is connected to V_{LL1} for 3 V glass operation. The HREFSEL bit is used to select 1.0 or 1.67 V range for V_{IREG}	For 3 V glass operation HREFSEL= 0, V_{IREG} = 1 V. V_{IREG} is connected V_{LL1} internally. V_{LCD} is not connected. Charge pump is used to generate V_{LL2} and V_{LL3} .	11	0	X	1	1

24.4.4.2.1 LCD External Power Supply, VSUPPLY[1:0] = 11

When VSUPPLY[1:0] = 11, powersw1, powersw2, are deasserted. V_{DD} is not available to power the LCD module internally, so the LCD module requires an external power source for $V_{LL1,}$ $V_{LL2,}$ and V_{LL3} when the charge pump is disabled.

If the charge pump is enabled, external power must be applied to V_{LL3}. With this configuration, the charge pump will generate the other LCD bias voltages V_{LL1} and $V_{LL2.}$

Internal V_{IREG}

If the charge pump is enabled the internal regulated voltage, V_{IREG}, can be used as an input to generate the LCD bias voltages.In this state external voltage source should not be connected to V_{LL1}. V_{IREG} is controlled by the LCD regulated voltage control (LCDRVC) register. The figure above, Figure 24-15, shows that V_{IREG} is connected to V_{LL1} when the RVEN bit is set.

V_{LL1} is connected to the internal charge pump.Using the charge pump, the value of V_{LL1} is tripled and output as $V_{LL3.}$$V_{LL3}$, an LCD bias voltage, is equal to the voltage required to energize the LCD panel, $V_{LCDON.}$For 3-V LCD glass, V_{LL3} should be approximately 3 V; while for 5-V LCD glass, V_{LL3} should be approximately 5 V.

The HREFSEL bit in the LCDSUPPLY register is used to set V_{IREG} to approximately 1.0 V or 1.67 V as shown in Table 24-23.The LCDRVC contains trim bits that can be used to make changes to the regulated voltage.The trim can be used to increase or decrease the regulated voltage by 1.5% for each count. A total of ±12% of change can be done to the regulated voltage.Table 24-23 shows the selected LCD bias voltages V_{LL1}, V_{LL2}, and V_{LL3} values based on the value of V_{IREG}.

Table 24-23. LCD bias Voltages VLL1, VLL2, and VLL3 based on V_{IREG}

HREFSELL	V_{IREG}	$V_{LL1} = V_{ref}$	$V_{LL2} = 2 \times V_{ref}$	$V_{LL3} = 3 \times V_{ref}$
HREFSEL = 0	1.0 V	1.0 V	2.0 V	3.0 V
HREFSEL =1	1.67 V	1.67 V	3.33 V	5.0 V

24.4.4.2.2 LCD Internal Power Supply, VSUPPLY[1:0] = 00 or 01

V_{DD} is used as the LCD module power supply when VSUPPLY[1:0] = 00 or 01 (Table 24-24). When powering the LCD module using V_{DD}, the charge pump must be enabled (CPSEL = 1). Table 24-24 provides recommendations regarding configuration of the VSUPPLY[1:0] bit field when using both 3-V and 5-V LCD panels.

Table 24-24. V_{DD} Switch Option

VSUPPLY[1:0]	V_{DD} Switch Option	Recommend Use for 3-V LCD Panels	Recommend Use for 5-V LCD Panels
00	V_{LL2} is generated from V_{DD}	• V_{LL1} = 1 V • V_{DD} = V_{LL2} = 2 V • V_{LL3} = 3 V	• V_{LL1} = 1.67 V • V_{DD} = V_{LL2} = 3.3 V • V_{LL3} = 5 V
01	V_{LL3} is generated from V_{DD}	• V_{LL1} = 1 V • V_{LL2} = 2 V • V_{DD} = V_{LL3} = 3 V	• V_{LL1} = 1.67 V • V_{LL2} = 3.33 V • V_{DD} = V_{LL3} = 5 V

24.4.5 Resets

During a reset, the LCD module system is configured in the default mode. The default mode includes the following settings:

- LCDEN is cleared, thereby forcing all frontplane and backplane driver outputs to the high impedance state.
- 1/4 duty
- 1/3 bias
- LCLK[2:0], VSUPPLY[1:0], BBYPASS, CPSEL, RVEN, and BRATE[2:0] revert to their reset values

24.4.6 Interrupts

When an LCD module frame-frequency interrupt event occurs, the LCDIF bit in the LCDS register is asserted. The LCDIF bit remains asserted until software clears the LCD-module-frame-frequency interrupt. The interrupt can be cleared by software writing a 1 to the LCDIF bit.

If both the LCDIF bit in the LCDS register and the LCDIEN bit in the LCDC1 register are set, an LCD interrupt signal asserts.

24.5 Initialization Section

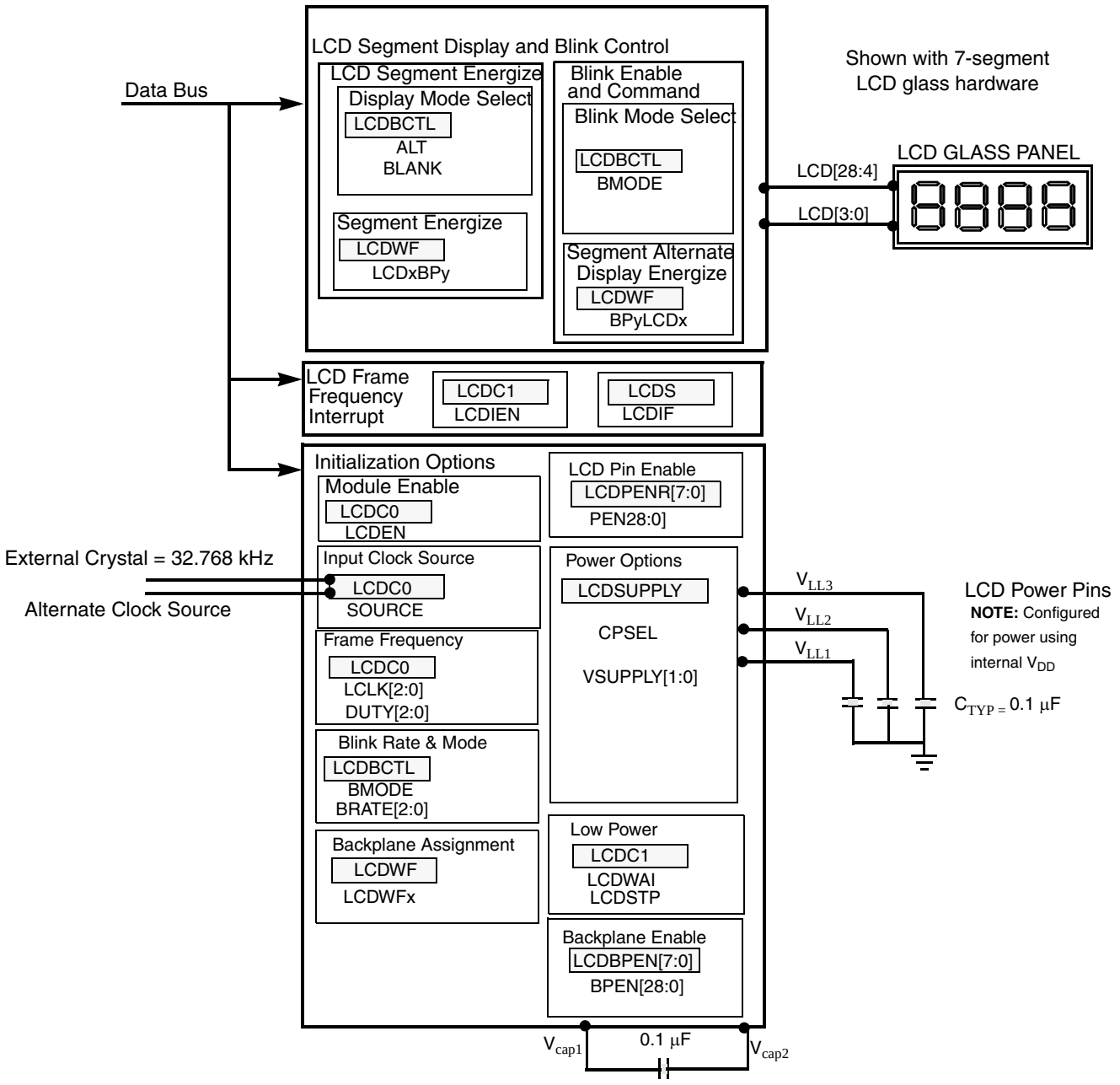
This section provides a recommended initialization sequence for the LCD module and also includes initialization examples for several LCD application scenarios.

24.5.1 Initialization Sequence

The below list provides a recommended initialization sequence for the LCD module.

You must write to all LCDPEN, LCDBPEN, and LCDWF registers to initialize their values after a reset.

1. LCDC0 register
   a) Configure LCD clock source (SOURCE bit).
2. LCDRVC register (If the application uses the internally regulated voltage)
   a) Select 1.0 V or 1.67 V for 3 or 5 V glass (HREFSEL).
   b) Enable regulated voltage (RVEN).
   c) Trim the regulated voltage (RVTRIM).
3. LCDSUPPLY register
   a) Enable charge pump (CPSEL bit).
   b) Configure LCD module for doubler or tripler mode (HREFSEL bit).
   c) Configure charge pump clock (LADJ[1:0]).
   d) Configure op amp switch (BBYPASS bit).
   e) Configure LCD power supply (VSUPPLY[1:0]).
4. LCDC1 register
   a) Configure LCD frame frequency interrupt (LCDIEN bit).
   b) Configure LCD behavior in low power mode (LCDWAI and LCDSTP bits).
5. LCDC0 register
   a) Configure LCD duty cycle (DUTY[2:0]).
   b) Select and configure LCD frame frequency (LCLK[2:0]).
6. LCDBCTL register
   a) Configure display mode (ALT and BLANK bits).
   b) Configure blink mode (BMODE).
   c) Configure blink frequency (BRATE[2:0]).
7. LCDPEN[7:0] register
   a) Enable LCD module pins (PEN[43:0] bits).
8. LCDBPEN[7:0]
   a) Enable LCD pins to operate as an LCD backplane (BPEN[43:0]).
9. LCDC0 register
   a) Enable LCD module (LCDEN bit).



LCD charge pump capacitance

Chapter 25
Programmable Reference Analog Comparator (PRACMP)

25.1 Introduction

The PRACMP is a CMOS comparator with a programmable reference input. The comparator has up to eight input pins, each of them can be compared with any input pins.There is also a internal programmable reference generator which divides the V_{In} into 32 levels, the V_{In} can be selected from two external sources. Output of this reference generator can be one of the eight inputs of comparator. The comparator circuit is designed to operate across the full range of the supply voltage (rail-to-rail operation).

Table 25-1. PRACMP Pins Mapping

PRACMP Internal Connection	Pins
External input 0	PRACMPxP0
External input 1	PRACMPxP1
External input 2	PRACMPxP2
External input 3	PRACMPxP3
External input 4	PRACMPxP4
External input 5	V_{ss}
External input 6	V_{ss}

NOTE

The PRACMP output can be connected internally to the SCI Rx input for infra-red reception implementation. For further information please refer to Figure 2-4 and Section 7.7.16, “Internal Peripheral Select Register 2 (SIMIPS2).”

The PRACMP register PRACMPxC2 has no effect in MCF51EMxx devices. For details to enable the PRACMP pins, refer to Section 4.7, “Pin Mux Controls.”

25.1.1 Features

PRACMP features include:

- On-chip programmable reference generator output (1/32 V_{in} to V_{in}, step is 1/32 V_{in}, V_{in} can be selected from external Vdd and internal regulated Vdd)
- Typically 5 mV of input offset
- Less than 40 μA in enable mode and less than 1 nA in disable mode (excluding programmable reference generator)
- Fixed ACMP hysteresis which is from 3 mV to 20 mV

- Up to eight selectable comparator inputs; each input can be compared with any input by any polarity sequence
- Selectable interrupt on rising edge, falling edge, or either rising or falling edges of comparator output
- Remains operational in stop3 mode and wait mode

25.1.2 Modes of Operation

This section defines the PRACMP operation in wait, stop, and background debug modes.

25.1.2.1 Operation in Wait Mode

The PRACMP continues to operate in wait mode if enabled. The interrupt can wake up the MCU if enabled.

25.1.2.2 Operation in Stop Mode

The PRACMP (including PRG and ACMP) continues to operate in stop3 mode if enabled. If ACIEN is set, a PRACMP interrupt still can be generated to wake the MCU up from stop3 mode.

If the stop3 is exited by an interrupt, the PRACMP remains the setting before entering the stop. If stop3 is exited with a reset, the PRACMP goes into its reset.

The user should turn it off if its output is not used as a reference input of ACMP to save power, because the PRG consumes additional power.

In stop2 mode, the PRACMP is shut down completely. Any waking up from stop2 and stop1 brings PRACMP to its reset state.

25.1.2.3 Operation in Background Mode

When the MCU is in active background debug mode, the PRACMP continues operating normally.

25.1.3 Block Diagram

The block diagram of the PRACMP module is shown in Figure 25-1.

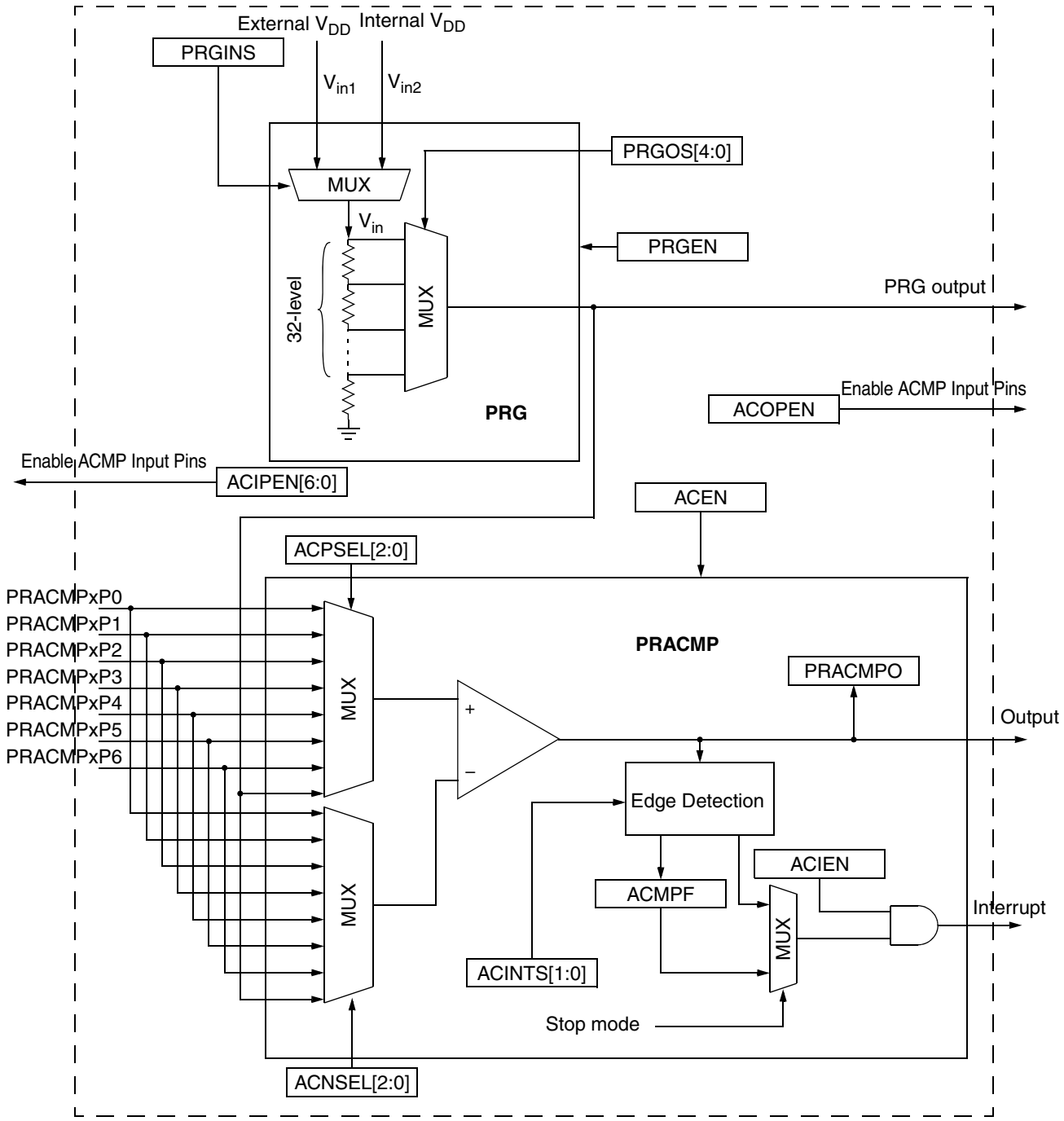


Figure 25-1. PRACMP Block Diagram

25.2 External Signal Description

The output of PRACMP can also be mapped to an external pin. When the output is mapped to an external pin, register bit ACOPE controls the pin to enable/disable the PRACMP output function.

25.3 Memory Map and Register Definition

Table 25-2 is the memory map of the programmable reference analog comparator (PRACMP).

Table 25-2. Module Memory Map

Address	Use	Access
Base + $0	PRACMP Control and Status Register (PRACMPxCS)	Read/Write
Base + $1	PRACMP Control Register 0 (PRACMPxC0)	Read/Write
Base + $2	PRACMP Control Register 1 (PRACMPxC1)	Read/Write
Base + $3	PRACMP Control Register 2 (PRACMPxC2)	Read/Write

Refer to the direct-page register summary in the memory chapter of this reference manual for the absolute address assignments for all PRACMP registers.

25.3.1 PRACMP Control and Status Register (PRACMPxCS)

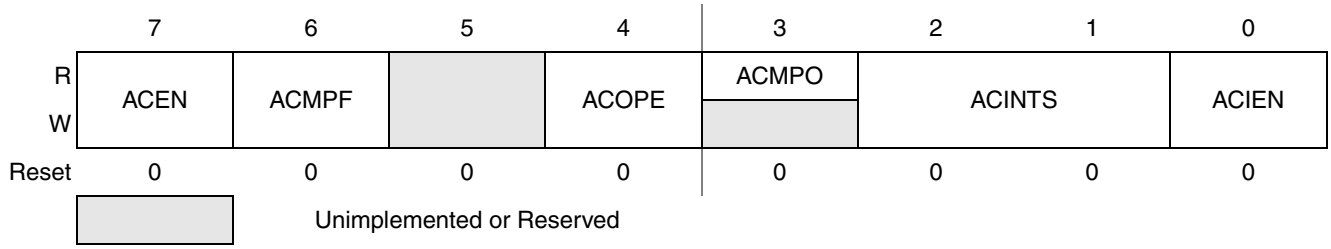


Figure 25-2. PRACMPx Control and Status Register (PRACMPxCS)

Table 25-3. PRACMPxCS Descriptions

Field	Description
7 ACEN	**ACMP enable control bit** 0 The ACMP is disabled 1 The ACMP is enabled
6 ACMPF	**ACMP Interrupt Flag Bit** — Synchronously set by hardware when ACMP output has a valid edge defined by ACINTS[1:0]. The setting of this bit lags the ACMPO 2 bus clocks. Clear ACMPF bit by writing a 0 to this bit. Writing a 1 to this bit has not effect.
4 ACOPE	**Analog Comparator Output Pin Enable** — ACOPE enables the comparator output to be placed onto the external pin, ACMPxO. 0 Analog comparator output is not available on ACMPx1O. 1 Analog comparator output is driven out on ACMPx1O.
3 ACMPO	**ACMP Output Bit** — ACMP output is synchronized by bus clock to form this bit. It changes following the ACMP output. This bit is a read only bit. Set when the output of the ACMP is high Cleared when the output of the ACMP is low After any reset or when the ACMP is disabled, this bit is read as 0.

Table 25-3. PRACMPxCS Descriptions (continued)

Field	Description
2:1 ACINTS [1:0]	**ACMP Interrupt Select** — Determines the sensitivity modes of the interrupt trigger. 00 ACMP interrupt on output falling or rising edge 01 ACMP interrupt on output falling edge 10 ACMP interrupt on output rising edge 11 Reserved
0 ACIEN	**ACMP Interrupt Enable** — Enables an ACMP CPU interrupt 1 Enable the ACMP Interrupt 0 Disable the ACMP Interrupt

25.3.2 PRACMP Control Register 0 (PRACMPxC0)

	7	6	5	4	3	2	1	0
R W		ACPSEL				ACNSEL		
Reset	0	0	0	0	0	0	0	0

Unimplemented or Reserved

Figure 25-3. PRACMPxControl Register 0 (PRACMPxC0)

Table 25-4. PRACMPxC0 Field Descriptions

Field	Description
6:4 ACPSEL[2:0]	**ACMP Positive Input Select** 000 External reference 0 001 External reference 1 010 External reference 2 011 External reference 3 100 External reference 4 101 External reference 5 110 External reference 6 111 Internal PRG output
2:0 ACNSEL[2:0]	**ACMP Negative Input Select** 000 External reference 0 001 External reference 1 010 External reference 2 011 External reference 3 100 External reference 4 101 External reference 5 110 External reference 6 111 Internal PRG output

25.3.3 PRACMP Control Register 1 (PRACMPxC1)

	7	6	5	4	3	2	1	0
R W	PRGEN	PRGINS		PRGOS4	PRGOS3	PRGOS2	PRGOS1	PRGOS0
Reset	0	0	0	0	0	0	0	0

Unimplemented or Reserved

Figure 25-4. PRACMPx Control Register 1 (PRACMPxC1)

Table 25-5. PRACMPxC1 Field Descriptions

Field	Description
7 PRGEN	**Programmable Reference Generator Enable** — The PRGEN bit starts the Programmable Reference Generator operation. 0 The PRG system is disabled 1 The PRG system is enabled
6 PRGINS	**Programmable Reference Generator Input Selection** 0 The PRG selects V_{in2} (internal regulated power supply)as the reference voltage 1 The PRG selects V_{in1} (external power supply)as the reference voltage
4:0 PRGOS[4:0]	**Programmable Reference Generator Output Selection** — The output voltage is selected by the following formula: V_{output}= (V_{in}/32)x(PRGOS[4:0]+1) The V_{output} range is from V_{in}/32 to V_{in}, the step is V_{in}/32

Table 25-6 list the output configuration of programmable reference generator (PRG).

Table 25-6. PRG Out Configuration

PRGOS[4:0]	Output Voltage of PRG
00000	$1V_{In}/32$
00001	$2V_{In}/32$
00010	$3V_{In}/32$
00011	$4V_{In}/32$
00100	$5V_{In}/32$
00101	$6V_{In}/32$
00110	$7V_{In}/32$
00111	$8V_{In}/32$
01000	$9V_{In}/32$
01001	$10V_{In}/32$
01010	$11V_{In}/32$
01011	$12V_{In}/32$
01100	$13V_{In}/32$
01101	$14V_{In}/32$
01110	$15V_{In}/32$

Table 25-6. PRG Out Configuration (continued)

PRGOS[4:0]	Output Voltage of PRG
01111	$16V_{In}/32$
10000	$17V_{In}/32$
10001	$18V_{In}/32$
10010	$19V_{In}/32$
10011	$20V_{In}/32$
10100	$21V_{In}/32$
10101	$22V_{In}/32$
10110	$23V_{In}/32$
10111	$24V_{In}/32$
11000	$25V_{In}/32$
11001	$26V_{In}/32$
11010	$27V_{In}/32$
11011	$28V_{In}/32$
11100	$29V_{In}/32$
11101	$30V_{In}/32$
11110	$31V_{In}/32$
11111	V_{In}

25.3.4 PRACMP Control Register 2 (PRACMPxC2)

	7	6	5	4	3	2	1	0
R	0	ACIPE6	ACIPE5	ACIPE4	ACIPE3	ACIPE2	ACIPE1	ACIPE0
W								
Reset	0	0	0	0	0	0	0	0

Unimplemented or Reserved

Figure 25-5. PRACMPx Control Register 2 (PRACMPxC2)

Table 25-7. PRACMPxC2 Field Descriptions

Field	Description
6:0 ACIPE6:ACIPE0	ACMP Input Pin Enable— This 7-bit register controls if the corresponding PRACMP external pin can be driven an analog input. 0 The corresponding external analog input is not allowed 1 The corresponding external analog input is allowed

25.4 Functional Description

The PRACMP module is functionally composed of two parts: programmable reference generator (PRG) and analog comparator (ACMP).

The programmable reference generator (PRG) includes a 32-level DAC (digital to analog convertor) and relevant control logic. PRG can select one of two reference inputs, V_{in1}(external Vdd) or V_{in2}(internal regulated Vdd), as the DAC input V_{in} by setting PRGINS bit of PRACMPxC1. After the DAC is enabled, it converts the data set in PRGOS[4:0] bits of PRACMPxC1 to a stepped analog output which is fed into ACMP as an internal reference input. This stepped analog ouput is also mapped out of the module. The output voltage range is from $V_{in}/32$ to $V_{in.}$.The step size is $V_{in}/32$.

The ACMP can achieve the analog comparison between positive input and negative input, and then give out a digital output and relevant interrupt. Both the positive and negative input of ACMP can be selected from the eight common inputs: seven external reference inputs and one internal reference input from the PRG output. The positive input of ACMP is selected by ACPSEL[2:0] bits of PRACMPxC0 and the negative input is selected by ACNSEL[2:0] bits of PRACMPxC0. Any pair of the eight inputs can be compared by configuring the PRACMPxC0 with the appropriate value.

After the ACMP is enabled by setting ACEN in PRACMPxCS, the comparison result appears as a digital output. Whenever a valid edge defined in ACINTS[1:0] occurs, the ACMPF bit in PRACMPxCS register is asserted. If ACIEN is set, a PRACMP CPU interrupt occurs. The valid edge is defined by ACINTS[1:0].When ACINTS[1:0] = 00, both the rising edge and falling edge on the ACMP output are valid. When ACINTS[1:0] = 01, only the falling edge on ACMP output is valid. When ACINTS[1:0] = 10, only rising edge on ACMP output is valid. ACINTS[1:0] = 11 is reserved.

The ACMP output is synchronized by the bus clock to generate ACMPO bit in PRACMPxCS so that the CPU can read the comparison. In stop3 mode if the output of ACMP is changed, ACMPO can’t be updated in time. The output can be synchronized and the ACMPO bit can be updated upon the waking up of the CPU because of the availability of the bus clock. The ACMPO changes following the comparison result, so it can serve as a tracking flag that continuously indicates the voltage delta on the inputs.

If a reference input external to the chip is selected as an input of ACMP, the corresponding ACIPE bit of PRACMPxC2 should be set to enable the input from pad interface. If the output of the ACMP needs to be put onto the external pin, the ACOPE bit of PRACMPxCS must be set to enable the ACMP pin function of pad logic.

25.5 Setup and Operation of PRACMP

The two parts of PRACMP (PRG and ACMP) can be set up and operated independently. But if the PRG works as an input of the ACMP, the PRG must be configured before the ACMP is enabled.

Because the input-switching can cause problems on the ACMP inputs, the user should complete the input selection before enableing the ACMP and should not change the input selection setting when the ACMP is enabled to avoid unexpected output. Similarly, because the programmable reference generator (PRG) experiences a setup delay after the PRGOS[4:0] is changed, the user should complete the setting of PRGOS[4:0] before PRG is enabled.

25.6 Resets

During a reset the PRACMP is configured in the default mode. Both ACMP and PRG are disabled.

25.7 Interrupts

If the bus clock is available when a valid edge defined in ACINTS[1:0] occurs, the ACMPF bit in PRACMPxCS register is asserted. If ACIEN is set, a PRACMP interrupt event occurs. The ACMPF bit remains asserted until the PRACMP interrupt is cleared by software. When in stop3 mode, a valid edge on ACMP output generates an asynchronous interrupt which can wake the MCU up from stop3. The interrupt can be cleared by writing a 0 to the ACMPF bit.

Chapter 26
Version 1 ColdFire Debug (CF1_DEBUG)

26.1 Introduction

This chapter describes the capabilities defined by the Version 1 ColdFire debug architecture. The Version 1 ColdFire core supports BDM functionality using the HCS08's single-pin interface. The traditional 3-pin full-duplex ColdFire BDM serial communication protocol based on 17-bit data packets is replaced with the HCS08 debug protocol where all communication is based on an 8-bit data packet using a single package pin (BKGD).

An on-chip trace buffer allows a stream of compressed processor execution status packets to be recorded for subsequent retrieval to provide program (and partial data) trace capabilities.

The following sections in this chapter provide details on the BKGD pin, the background debug serial interface controller (BDC), a standard 6-pin BDM connector, the BDM command set as well as real-time debug and trace capabilities. The V1 definition supports revision B+ (DEBUG_B+) of the ColdFire debug architecture.

A simplified block diagram of the V1 core including the processor and debug module is shown in Figure 26-1.

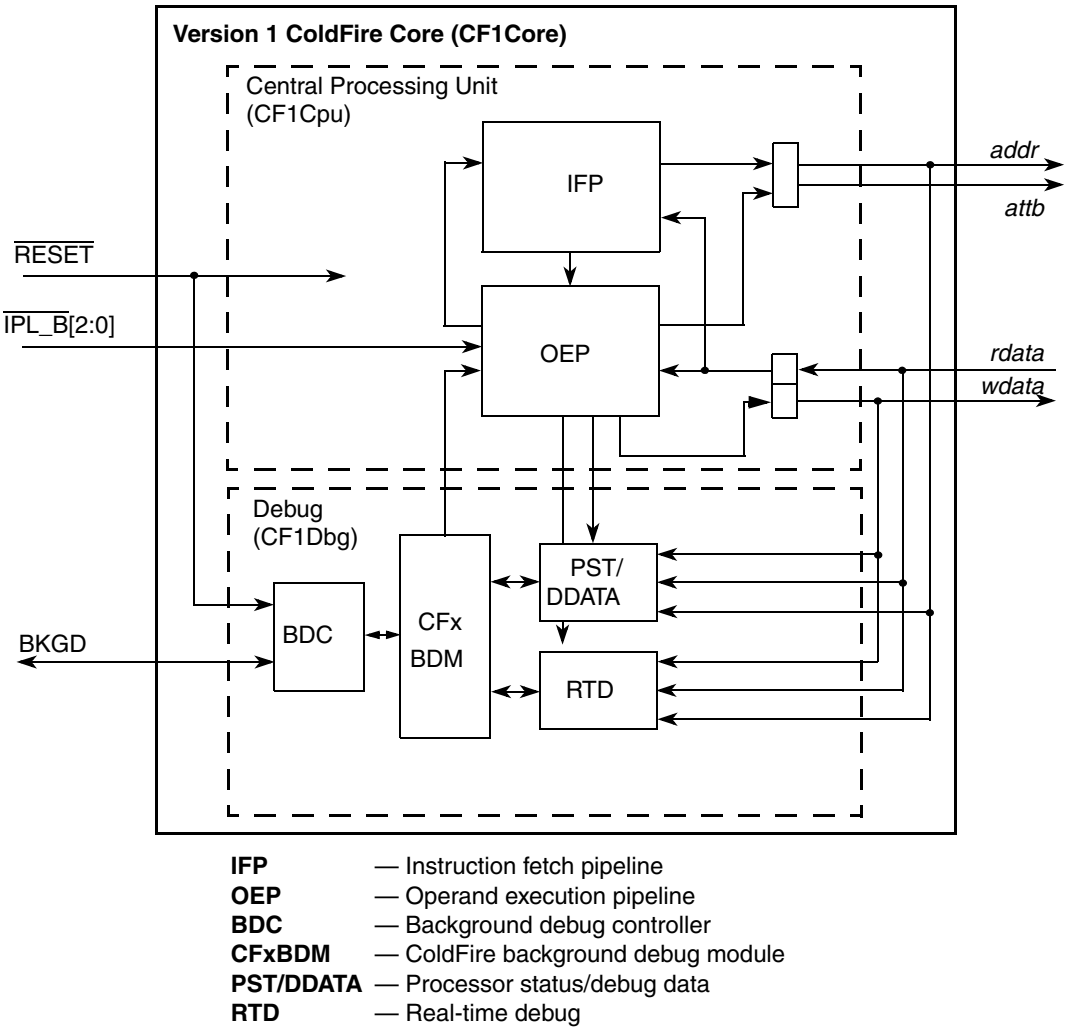


Figure 26-1. Simplified Version 1 ColdFire Core Block Diagram

26.1.1 Overview

Debug support is divided into three areas:

- Background debug mode (BDM)—Provides low-level debugging in the ColdFire processor core. In BDM, the processor core is halted and a variety of commands can be sent to the processor to access memory, registers, and peripherals. The external emulator uses a one-pin serial communication protocol. See Section 26.4.1, “Background Debug Mode (BDM)”.
- Real-time debug support—Use of the full BDM command set requires the processor to be halted, which many real-time embedded applications cannot support. The core includes a variety of internal breakpoint registers which can be configured to trigger and generate a special interrupt. The resulting debug interrupt lets real-time systems execute a unique service routine that can quickly save the contents of key registers and variables and return the system to normal operation. The external development system can then access the saved data, because the hardware supports

concurrent operation of the processor and BDM-initiated memory commands. In addition, the option is provided to allow interrupts to occur. See Section 26.4.2, “Real-Time Debug Support”.

- Program trace support—The ability to determine the dynamic execution path through an application is fundamental for debugging. The V1 solution implements a trace buffer that records processor execution status and data, which can be subsequently accessed by the external emulator system to provide program (and optional partial data) trace information. See Section 26.4.3, “Trace Support”.

There are two fields in debug registers which provide revision information: the hardware revision level in CSR and the 1-pin debug hardware revision level in CSR2. Table 26-1 summarizes the various debug revisions.

Table 26-1. Debug Revision Summary

Revision	CSR[HRL]	CSR2[D1HRL]	Enhancements
A	0000	N/A	Initial ColdFire debug definition
B	0001	N/A	BDM command execution does not affect hardware breakpoint logic Added BDM address attribute register (BAAR) $\overline{\text{BKPT}}$ configurable interrupt (CSR[BKD]) Level 1 and level 2 triggers on OR condition, in addition to AND SYNC_PC command to display the processor’s current PC
B+	1001	N/A	Added 3 PC breakpoint registers PBR1–3
CF1_B+	1001	0001	Converted to HCS08 1-pin BDM serial interface Added PST compression and on-chip PST/DDATA buffer for program trace

26.1.2 Features

The Version 1 ColdFire debug definition supports the following features:

- Classic ColdFire DEBUG_B+ functionality mapped into the single-pin BDM interface
- Real time debug support, with 6 hardware breakpoints (4 PC, 1 address pair and 1 data) that can be configured into a 1- or 2-level trigger with a programmable response (processor halt or interrupt)
- Capture of compressed processor status and debug data into on-chip trace buffer provides program (and optional slave bus data) trace capabilities
- On-chip trace buffer provides programmable start/stop recording conditions plus support for obtrusive or PC-profiling modes
- Debug resources are accessible via single-pin BDM interface or the privileged WDEBUG instruction from the core

26.1.3 Modes of Operations

V1 ColdFire devices typically implement a number of modes of operation, including run, wait, and stop modes. Additionally, the operation of the core’s debug module is highly dependent on a number of chip configurations which determine its operating state.

When operating in secure mode, as defined by a 2-bit field in the flash memory examined at reset, BDM access to debug resources is extremely restricted. It is possible to tell that the device has been secured, and

to clear security, which involves mass erasing the on-chip flash memory. No other debug access is allowed. Secure mode can be used in conjunction with each of the wait and stop low-power modes.

If the BDM interface is not enabled, access to the debug resources is limited in the same manner as a secure device.

If the device is not secure and the BDM interface is enabled (XCSR[ENBDM] is set), the device is operating in debug mode and additional resources are available via the BDM interface. In this mode, the status of the processor (running, stopped, or halted) determines which BDM commands may be used.

Debug mode functions are managed through the background debug controller (BDC) in the Version 1 ColdFire core. The BDC provides the means for analyzing MCU operation during software development.

BDM commands can be classified into three types as shown in Table 26-2.

Table 26-2. BDM Command Types

Command Type	Flash Secure?	BDM?	Core Status	Command Set
Always-available	Secure or Unsecure	Enabled or Disabled	—	• Read/write access to XCSR[31–24], CSR2[31–24], CSR3[31–24]
Non-intrusive	Unsecure	Enabled	Run, Halt	• Memory access • Memory access with status • Debug register access • BACKGROUND
Active background	Unsecure	Enabled	Halt	• Read or write CPU registers (also available in stop mode) • Single-step the application • Exit halt mode to return to the application program (GO)

For more information on these three BDM command classifications, see Section 26.4.1.5, “BDM Command Set Summary.”

The core’s halt mode is entered in a number of ways:

- The BKGD pin is low during POR
- The BKGD pin is low immediately after a BDM-initiated force reset (see CSR2[BDFR] in Section 26.3.3, “Configuration/Status Register 2 (CSR2),” for details)
- A background debug force reset occurs (CSR2[BDFR] is set) and CSR2[BFHBR] is set
- A computer operating properly reset occurs and CSR2[COPHR] is set
- An illegal operand reset occurs and CSR2[IOPHR] is set
- An illegal address reset occurs and CSR2[IADHR] is set
- A BACKGROUND command is received through the BKGD pin. If necessary, this wakes the device from STOP/WAIT modes.
- A properly-enabled (XCSR[ENBDM] is set) HALT instruction is executed
- Encountering a BDM breakpoint and the trigger response is programmed to generate a halt
- Reaching a PSTB trace buffer full condition when operating in an obtrusive recording mode (CSR2[PSTBRM] is set to 01 or 11)

While in halt mode, the core waits for serial background commands rather than executing instructions from the application program.

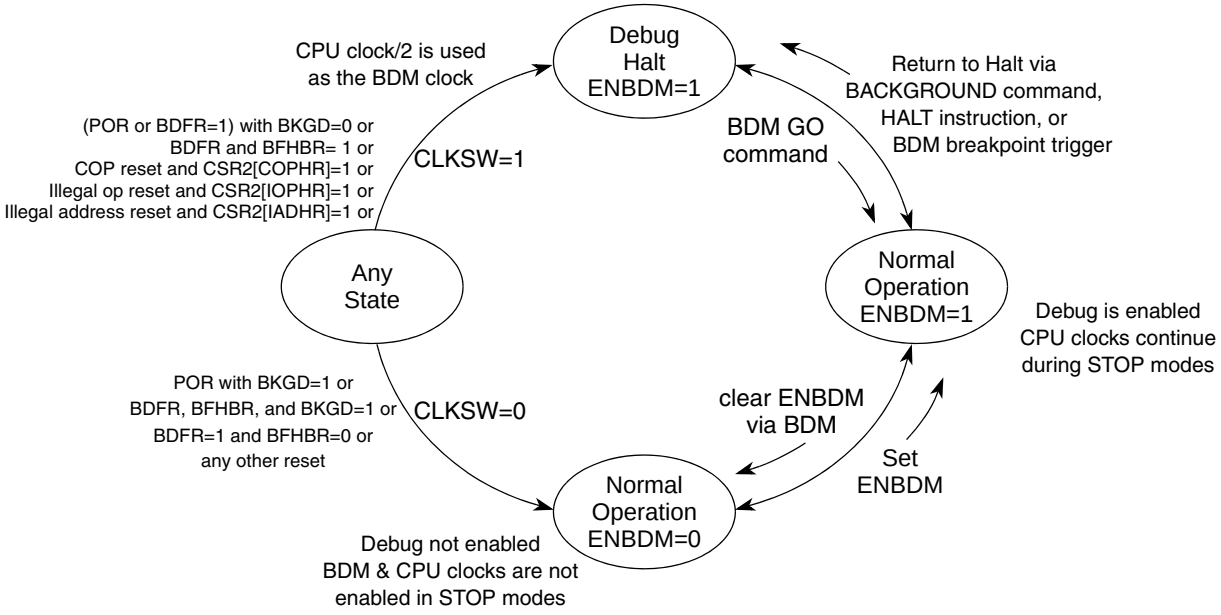


Figure 26-2. Debug Modes State Transition Diagram

Figure 26-2 contains a simplified view of the V1 ColdFire debug mode states. The XCSR[CLKSW] bit controls the BDC clock source. When CLKSW is set, the BDC serial clock frequency is half the CPU clock. When CLKSW is cleared, the BDC serial clock is supplied from an alternate clock source.

The ENBDM bit determines if the device can be placed in halt mode, if the core and BDC serial clocks continue to run in STOP modes, and if the regulator can be placed into standby mode. Again, if booting to halt mode, XCSR[ENBDM, CLKSW] are automatically set.

If ENBDM is cleared, the ColdFire core treats the HALT instruction as an illegal instruction and generates a reset (if CPUCR[IRD] is cleared) or an exception (if CPUCR[IRD] is set) if execution is attempted.

If XCSR[ENBDM] is set, the device can be restarted from STOP/WAIT via the BDM interface.

26.2 External Signal Descriptions

Table 26-3 describes the debug module's 1-pin external signal (BKGD). A standard 6-pin debug connector is shown in Section 26.4.4, "Freescale-Recommended BDM Pinout".

Table 26-3. Debug Module Signals

Signal	Description
Background Debug (BKGD)	Single-wire background debug interface pin. The primary function of this pin is for bidirectional serial communication of background debug mode commands and data. During reset, this pin selects between starting in active background (halt) mode or starting the application program. This pin also requests a timed sync response pulse to allow a host development tool to determine the correct clock frequency for background debug serial communications.

26.3 Memory Map/Register Definition

In addition to the BDM commands that provide access to the processor's registers and the memory subsystem, the debug module contains a number of registers. Most of these registers (all except the PST/DDATA trace buffer) are also accessible (write-only) from the processor's supervisor programming model by executing the WDEBUG instruction. Thus, the breakpoint hardware in the debug module can be read (certain registers) or written by the external development system using the serial debug interface or written by the operating system running on the processor core. Software is responsible for guaranteeing that accesses to these resources are serialized and logically consistent. The hardware provides a locking mechanism in the CSR to allow the external development system to disable any attempted writes by the processor to the breakpoint registers (setting CSR[IPW]). BDM commands must not be issued during the processor's execution of the WDEBUG instruction to configure debug module registers or the resulting behavior is undefined.

These registers, shown in Table 26-4, are treated as 32-bit quantities regardless of the number of implemented bits and unimplemented bits are reserved and must be cleared. These registers are also accessed through the BDM port by the commands, WRITE_DREG and READ_DREG, described in Section 26.4.1.5, "BDM Command Set Summary." These commands contain a 5-bit field, DRc, that specifies the register, as shown in Table 26-4.

Table 26-4. Debug Module Memory Map

DRc	Register Name	Width (bits)	Access	Reset Value	Section/ Page
0x00	Configuration/status register (CSR)	32	R/W (BDM), W (CPU)	0x0090_0000	26.3.1/26-7
0x01	Extended Configuration/Status Register (XCSR)	32	R/W[1] (BDM), W (CPU)	0x0000_0000	26.3.2/26-10
0x02	Configuration/Status Register 2 (CSR2)	32	R/W[1] (BDM), W (CPU)	See Section	26.3.3/26-13
0x03	Configuration/Status Register 3 (CSR3)	32[2]	R/W[1] (BDM), W (CPU)	0x0000_0000	26.3.4/26-16
0x05	BDM address attribute register (BAAR)	32[2]	W	0x0000_0005	26.3.5/26-17
0x06	Address attribute trigger register (AATR)	32[2]	W	0x0000_0005	26.3.6/26-18
0x07	Trigger definition register (TDR)	32	W	0x0000_0000	26.3.7/26-19
0x08	PC breakpoint register 0 (PBR0)	32	W	Undefined, Unaffected	26.3.8/26-22
0x09	PC breakpoint mask register (PBMR)	32	W	Undefined, Unaffected	26.3.8/26-22
0x0C	Address breakpoint high register (ABHR)	32	W	Undefined, Unaffected	26.3.9/26-24
0x0D	Address breakpoint low register (ABLR)	32	W	0x0000_0000	26.3.9/26-24
0x0E	Data breakpoint register (DBR)	32	W	0x0000_0000	26.3.10/26-25
0x0F	Data breakpoint mask register (DBMR)	32	W	0x0000_0000	26.3.10/26-25

Table 26-4. Debug Module Memory Map (continued)

DRc	Register Name	Width (bits)	Access	Reset Value	Section/ Page
0x18	PC breakpoint register 1 (PBR1)	32	W	PBR1[0] = 0	26.3.8/26-22
0x1A	PC breakpoint register 2 (PBR2)	32	W	PBR2[0] = 0	26.3.8/26-22
0x1B	PC breakpoint register 3 (PBR3)	32	W	PBR3[0] = 0	26.3.8/26-22
—	PST Trace Buffer *n* (PSTB*n*); *n* = 0–11 (0xB)	32	R (BDM)[3]	Undefined, Unaffected	26.4.1.5.12/26-46

1 The most significant bytes of the XCSR, CSR2, and CSR3 registers support special control functions and are writeable via BDM using the WRITE_XCSR_BYTE, WRITE_CSR2_BYTE, and WRITE_CSR3_BYTE commands. They can be read from BDM using the READ_XCSR_BYTE, READ_CSR2_BYTE, and READ_CSR3_BYTE commands. These 3 registers, along with the CSR, can also be referenced as 32-bit quantities using the BDM READ_DREG and WRITE_DREG commands, but the WRITE_DREG command only writes bits 23–0 of these three registers.

2 Each debug register is accessed as a 32-bit value; undefined fields are reserved and must be cleared.

3 The contents of the PST trace buffer is only read from BDM (32 bits per access) using READ_PSTB commands.

NOTE

Debug control registers can be written by the external development system or the CPU through the WDEBUG instruction. These control registers are write-only from the programming model and they can be written through the BDM port using the WRITE_DREG command. In addition, the four configuration/status registers (CSR, XCSR, CSR2, CSR3) can be read through the BDM port using the READ_DREG command.

The ColdFire debug architecture supports a number of hardware breakpoint registers that can be configured into single- or double-level triggers based on the PC or operand address ranges with an optional inclusion of specific data values. The triggers can be configured to halt the processor or generate a debug interrupt exception. Additionally, these same breakpoint registers can be used to specify start/stop conditions for recording in the PST trace buffer.

The core includes four PC breakpoint triggers and a set of operand address breakpoint triggers with two independent address registers (to allow specification of a range) and an optional data breakpoint with masking capabilities. Core breakpoint triggers are accessible through the serial BDM interface or written through the supervisor programming model using the WDEBUG instruction.

26.3.1 Configuration/Status Register (CSR)

CSR defines the debug configuration for the processor and memory subsystem and contains status information from the breakpoint logic. CSR is accessible from the programming model using the WDEBUG instruction and through the BDM port using the READ_DREG and WRITE_DREG commands.

DRc[4:0]: 0x00 (CSR)

Access: Supervisor write-only
BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BSTAT				FOF	TRG	HALT	BKPT	HRL				0	BKD	0	IPW
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TRC	0	DDC		UHE	BTB		0	NPL	IPI	SSM	0	0	FID	DDH
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-3. Configuration/Status Register (CSR)

Table 26-5. CSR Field Descriptions

Field	Description
31–28 BSTAT	Breakpoint status. Provides read-only status (from the BDM port only) information concerning hardware breakpoints. BSTAT is cleared by a TDR write, by a CSR read when a level-2 breakpoint is triggered, or a level-1 breakpoint is triggered and the level-2 breakpoint is disabled. The PSTB value that follows the PSTB entry of 0x1B is 0x20 + (2 × BSTAT). 0000 No breakpoints enabled 0001 Waiting for level-1 breakpoint 0010 Level-1 breakpoint triggered 0101 Waiting for level-2 breakpoint 0110 Level-2 breakpoint triggered
27 FOF	Fault-on-fault. Indicates a catastrophic halt occurred and forced entry into BDM. FOF is cleared by reset or when CSR is read (from the BDM port only).
26 TRG	Hardware breakpoint trigger. Indicates a hardware breakpoint halted the processor core and forced entry into BDM. Reset, the debug GO command, or reading CSR (from the BDM port only) clears TRG.
25 HALT	Processor halt. Indicates the processor executed a HALT and forced entry into BDM. Reset, the debug GO command, or reading CSR (from the BDM port only) clears HALT.
24 BKPT	Breakpoint assert. Indicates when either: • The $\overline{\text{BKPT}}$ input was asserted, • BDM BACKGROUND command received, or • The PSTB halt on full condition, CSR2[PSTBH], sets. This forces the processor into a BDM halt. Reset, the debug GO command, or reading CSR (from the BDM port only) clears BKPT.
23–20 HRL	Hardware revision level. Indicates, from the BDM port only, the level of debug module functionality. An emulator can use this information to identify the level of functionality supported. 0000 Revision A 0001 Revision B 0010 Revision C 0011 Revision D 1001 Revision B+ (The value used for this device) 1011 Revision D+
19	Reserved, must be cleared.

Table 26-5. CSR Field Descriptions (continued)

Field	Description
18 BKD	Breakpoint disable. Disables the BACKGROUND command functionality, and allows the execution of the BACKGROUND command to generate a debug interrupt. 0 Normal operation 1 The receipt of a BDM BACKGROUND command signals a debug interrupt to the ColdFire core. The processor makes this interrupt request pending until the next sample point occurs, when the exception is initiated. In the ColdFire architecture, the interrupt sample point occurs once per instruction. There is no support for nesting debug interrupts.
17	Reserved, must be cleared.
16 IPW	Inhibit processor writes. Inhibits processor-initiated writes to the debug module's programming model registers. IPW can be modified only by commands from the BDM interface.
15	Reserved, must be cleared.
14 TRC	Force emulation mode on trace exception. 0 Processor enters supervisor mode. 1 Processor enters emulator mode when a trace exception occurs.
13	Reserved, must be cleared.
12–11 DDC	Debug data control. Controls peripheral bus operand data capture for DDATA, which displays the number of bytes defined by the operand reference size (a marker) before the actual data; byte displays 8 bits, word displays 16 bits, and long displays 32 bits (one nibble at a time across multiple PSTCLK clock cycles). See Table 26-26. A non-zero value enables partial data trace capabilities. 00 No operand data is displayed. 01 Capture all write data. 10 Capture all read data. 11 Capture all read and write data.
10 UHE	User halt enable. Selects the CPU privilege level required to execute the HALT instruction. The core must be operating with XCSR[ENBDM] set to execute any HALT instruction, else the instruction is treated as an illegal opcode. 0 HALT is a supervisor-only instruction. 1 HALT is a supervisor/user instruction.
9–8 BTB	Branch target bytes. Defines the number of bytes of branch target address DDATA displays. See Section 26.4.3.1, "Begin Execution of Taken Branch (PST = 0x05)." 00 No target address capture 01 Lower 2 bytes of the target address 1*x* Lower 3 bytes of the target address
7	Reserved, must be cleared.
6 NPL	Non-pipelined mode. Determines if the core operates in pipelined mode. 0 Pipelined mode 1 Non-pipelined mode. The processor effectively executes one instruction at a time with no overlap. This typically adds five cycles to the execution time of each instruction. Given an average execution latency of ~2 cycles per instruction, throughput in non-pipeline mode would be ~7 cycles per instruction, approximately 25% - 33% of pipelined performance. Regardless of the NPL state, a triggered PC breakpoint is always reported before the triggering instruction executes. In normal pipeline operation, the occurrence of an address and/or data breakpoint trigger is imprecise. In non-pipeline mode, these triggers are always reported before the next instruction begins execution and trigger reporting can be considered precise.

Table 26-5. CSR Field Descriptions (continued)

Field	Description
5 IPI	Ignore pending interrupts when in single-step mode. 0 Core services any pending interrupt requests signalled while in single-step mode. 1 Core ignores any pending interrupt requests signalled while in single-step mode.
4 SSM	Single-step mode enable. 0 Normal mode. 1 Single-step mode. The processor halts after execution of each instruction. While halted, any BDM command can be executed. On receipt of the GO command, the processor executes the next instruction and halts again. This process continues until SSM is cleared.
3–2	Reserved, must be cleared.
1 FID	Force *ipg_debug*. The core generates this output to the device, signaling it is in debug mode. 0 Do not force the assertion of *ipg_debug* 1 Force the assertion of *ipg_debug*
0 DDH	Disable *ipg_debug* due to a halt condition. The core generates an output to the other modules in the device, signaling it is in debug mode. By default, this output signal is asserted when the core halts. 0 Assert *ipg_debug* if the core is halted 1 Negate *ipg_debug* due to the core being halted

26.3.2 Extended Configuration/Status Register (XCSR)

The 32-bit XCSR is partitioned into two sections: the upper byte contains status and command bits always accessible to the BDM interface, even if debug mode is disabled. This status byte is also known as XCSR_SB. The lower 24 bits contain fields related to the generation of automatic SYNC_PC commands, which can be used to periodically capture and display the current program counter (PC) in the PST trace buffer (if properly configured).

There are multiple ways to reference the XCSR. They are summarized in Table 26-6.

Table 26-6. XCSR Reference Summary

Method	Reference Details
READ_XCSR_BYTE	Reads XCSR[31–24] from the BDM interface. Available in all modes.
WRITE_XCSR_BYTE	Writes XCSR[31–24] from the BDM interface. Available in all modes.
READ_DREG	Reads XCSR[31–0] from the BDM interface. Classified as a non-intrusive BDM command.
WRITE_DREG	Writes XCSR[23–0] from the BDM interface. Classified as a non-intrusive BDM command.
WDEBUG instruction	Writes XCSR[23–0] during the core's execution of WDEBUG instruction. This instruction is a privileged supervisor-mode instruction.

DRc: 0x01 (XCSR)

Access: Supervisor write-only
BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CPU HALT	CPU STOP	CSTAT			CLK SW	SEC	EN BDM	0	0	0	0	0	0	0	0
W			ESEQC				ERASE									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	APCSC		APC ENB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-4. Extended Configuration/Status Register (XCSR)

Table 26-7. XCSR Field Descriptions

Field	Description
31 CPUHALT	Indicates that the CPU is in the halt state. The CPU state may be running, stopped, or halted, which is determined by the CPUHALT and CPUSTOP bits as shown below. **XCSR [CPUHALT]** / **XCSR [CPUSTOP]** / **CPU State** 0 / 0 / Running 0 / 1 / Stopped 1 / 0 / Halted
30 CPUSTOP	Indicates that the CPU is in the stop state. The CPU state may be running, stopped, or halted, which is determined by the CPUHALT and CPUSTOP bits as shown in the CPUHALT bit description.

Table 26-7. XCSR Field Descriptions (continued)

Field	Description
29–27 CSTAT (R) ESEQC (W)	During reads, indicates the BDM command status. 000 Command done, no errors 001 Command done, data invalid 01*x* Command done, illegal 1*xx* Command busy, overrun If an overrun is detected (CSTAT = 1*xx*), the following sequence is suggested to clear the source of the error: 1. Issue a SYNC command to reset the BDC channel. 2. The host issues a BDM NOP command. 3. The host checks the channel status using a READ_XCSR_BYTE command. 4. If XCSR[CSTAT] = 000 then status is okay; proceed else Halt the CPU with a BDM BACKGROUND command Repeat steps 1,2,3 If XCSR[CSTAT] ≠ 000, then reset device During writes, the ESEQC field is used for the erase sequence control during flash programming. ERASE must also be set for this bit to have an effect. 000 User mass erase Else Reserved **Note:** See the Memory chapter for a detailed description of the algorithm for clearing security.
26 CLKSW	Select source for serial BDC communication clock. 0 Alternate, asynchronous BDC clock, typically 10 MHz 1 Synchronous bus clock (CPU clock divided by 2) The initial state of the XCSR[CLKSW] bit is loaded by the hardware in response to certain reset events and the state of the BKGD pin as described in Figure 26-2.
25 SEC (R) ERASE (W)	The read value of this bit typically defines the status of the flash security field. 0 Flash security is disabled 1 Flash security is enabled In addition, the SEC bit is context-sensitive during reads. After a mass-erase sequence has been initiated by BDM, it acts as a flash busy flag. When the erase operation is complete and the bit is cleared, it returns to reflect the status of the chip security. 0 Flash is not busy performing a BDM mass-erase sequence 1 Flash is busy performing a BDM mass-erase sequence During writes, this bit qualifies XCSR[ESEQC] for the write modes shown in the ESEQC field description. 0 Do not perform a mass-erase of the flash. 1 Perform a mass-erase of the flash, using the sequence specified in the XCSR[ESEQC] field.
24 ENBDM	Enable BDM. 0 BDM mode is disabled 1 Active background mode is enabled (assuming the flash is not secure)
23–3	Reserved for future use by the debug module, must be cleared.

Table 26-7. XCSR Field Descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>2–1
APCSC</td><td>Automatic PC synchronization control. Determines the periodic interval of PC address captures, if XCSR[APCENB] is set. When the selected interval is reached, a SYNC_PC command is sent to the ColdFire CPU. For more information on the SYNC_PC operation, see the APCENB description.

The chosen frequency depends on CSR2[APCDIV16] as shown in the equation and table below:
$$\text{PC address capture period} = \frac{2^{(\text{APCSC}+1)} \times 1024}{16^{\text{APCDIV16}}}$$ Eqn. 26-1
<table>
<tr><th>XCSR [APCENB]</th><th>CSR2 [APCDIV16]</th><th>XCSR [APCSC]</th><th>SYNC_PC Interval</th></tr>
<tr><td>1</td><td>0</td><td>00</td><td>2048 cycles</td></tr>
<tr><td>1</td><td>0</td><td>01</td><td>4096 cycles</td></tr>
<tr><td>1</td><td>0</td><td>10</td><td>8192 cycles</td></tr>
<tr><td>1</td><td>0</td><td>11</td><td>16384 cycles</td></tr>
<tr><td>1</td><td>1</td><td>00</td><td>128 cycles</td></tr>
<tr><td>1</td><td>1</td><td>01</td><td>256 cycles</td></tr>
<tr><td>1</td><td>1</td><td>10</td><td>512 cycles</td></tr>
<tr><td>1</td><td>1</td><td>11</td><td>1024 cycles</td></tr>
</table>
</td></tr>
<tr><td>0
APCENB</td><td>Automatic PC synchronization enable. Enables the periodic output of the PC which can be used for PST/DDATA trace synchronization and code profiling.
As described in XCSR[APCSC], when the enabled periodic timer expires, a SYNC_PC command is sent to the ColdFire CPU which generates a forced instruction fetch of the next instruction. The PST/DDATA module captures the target address as defined by CSR[9] (two bytes if CSR[9] is cleared, three bytes if CSR[9] is set). This produces a PST sequence of the PST marker indicating a 2- or 3-byte address, followed by the captured instruction address.
0 Automatic PC synchronization disabled
1 Automatic PC synchronization enabled</td></tr>
</table>

26.3.3 Configuration/Status Register 2 (CSR2)

The 32-bit CSR2 is partitioned into two sections. The upper byte contains status and configuration bits always accessible to the BDM interface, even if debug mode is disabled. The lower 24 bits contain fields related to the configuration of the PST trace buffer (PSTB).

There are multiple ways to reference CSR2. They are summarized in Table 26-8.

Table 26-8. CSR2 Reference Summary

Method	Reference Details
READ_CSR2_BYTE	Reads CSR2[31—24] from the BDM interface. Available in all modes.
WRITE_CSR2_BYTE	Writes CSR2[31—24] from the BDM interface. Available in all modes.
READ_DREG	Reads CSR2[31—0] from the BDM interface. Classified as a non-intrusive BDM command.

Table 26-8. CSR2 Reference Summary (continued)

Method	Reference Details
WRITE_DREG	Writes CSR2[23–0] from the BDM interface. Classified as a non-intrusive BDM command.
WDEBUG Instruction	Writes CSR2[23–0] during the core's execution of WDEBUG instruction. This instruction is a privileged supervisor-mode instruction.

DRc: 0x02 (CSR2)

Access: Supervisor read-only
BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PSTBP	0	COP HR	IOP HR	IAD HR	0	BFHBR	0	PSTBH	PSTBST		0	D1HRL			
W								BDFR								
Power-on Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Other Reset	0	0	u	u	u	0	u	0	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PSTBWA								0	APC DIV16	0	PSTBRM		PSTBSS		
W									PSTBR							
Reset	Unaffected and Undefined								0	0	0	0	0	0	0	0

Figure 26-5. Configuration/Status Register 2 (CSR2)

Table 26-9. CSR2 Field Descriptions

Field	Description
31 PSTBP	PST buffer stop. Signals if a PST buffer stop condition has been reached. 0 A PST trace buffer stop condition has not been reached 1 A PST trace buffer stop condition has been reached
30	Reserved, must be cleared.
29 COPHR	Computer operating properly halt after reset. Determines operation of the device after a COP reset. This bit is cleared after a power-on reset and is unaffected by any other reset. 0 After a computer operating properly reset, the device immediately enters normal operation mode. 1 A computer operating properly reset immediately halts the device (as if the BKGD pin was held low after a power-on reset). **Note:** This bit may only be changed if XCSR[ENBDM] is set and the flash is unsecure.
28 IOPHR	Illegal operation halt after reset. Determines operation of the device after an illegal operation reset. This bit is cleared after a power-on reset and is unaffected by any other reset. 0 After the device has an illegal operation reset, the device immediately enters normal operation mode. 1 An illegal operation reset immediately halts the device (as if the BKGD pin was held low after a power-on reset). **Note:** This bit may only be changed if XCSR[ENBDM] is set and the flash is unsecure.
27 IADHR	Illegal address halt after reset. Determines operation of the device after an illegal address reset. This bit is cleared after a power-on reset and is unaffected by any other reset. 0 After the device has an illegal address reset, the device immediately enters normal operation mode. 1 An illegal address reset immediately halts the device (as if the BKGD pin was held low after a power-on reset). **Note:** This bit may only be changed if XCSR[ENBDM] is set and the flash is unsecure.

Table 26-9. CSR2 Field Descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>26</td><td>Reserved, must be cleared.</td></tr>
<tr><td>25
BFHBR</td><td>BDM force halt on BDM reset. Determines operation of the device after a BDM reset. This bit is cleared after a power-on reset and is unaffected by any other reset.
0 The device enters normal operation mode following a BDM reset.
1 The device enters in halt mode following a BDM reset, as if the BKGD pin was held low after a power-on-reset or standard BDM-initiated reset.
Note: This bit can only change state if XCSR[ENBDM] = 1 and the flash is unsecure.</td></tr>
<tr><td>24
BDFR</td><td>Background debug force reset. Forces a BDM reset to the device. This bit always reads as 0 after the reset has been initiated.
0 No reset initiated.
1 Force a BDM reset.</td></tr>
<tr><td>23
PSTBH</td><td>PST trace buffer halt. Indicates if the processor is halted due to the PST trace buffer being full when recording in a obtrusive mode.
0 PST trace buffer not full
1 CPU halted due to PST trace buffer being full in obtrusive mode</td></tr>
<tr><td>22–21
PSTBST</td><td>PST trace buffer state. Indicates the current state of the PST trace buffer recording.
00 PSTB disabled
01 PSTB enabled and waiting for the start condition
10 PSTB enabled, recording and waiting for the stop condition
11 PSTB enabled, completed recording after the stop condition was reached</td></tr>
<tr><td>20</td><td>Reserved, must be cleared.</td></tr>
<tr><td>19–16
D1HRL</td><td>Debug 1-pin hardware revision level. Indicates the hardware revision level of the 1-pin debug module implemented in the ColdFire core. For this device, this field is 0x1.</td></tr>
<tr><td>15–8
PSTBWA</td><td>PST trace buffer write address. Indicates the current write address of the PST trace buffer. The most significant bit of this field is sticky; if set, it remains set until a PST/DDATA reset event occurs. As the ColdFire core inserts PST and DDATA packets into the trace buffer, this field is incremented. The value of the write address defines the next location in the PST trace buffer to be loaded. In other words, the contents of PSTB[PSTBWA-1] is the last valid entry in the trace buffer.
The msb of this field can be used to determine if the entire PST trace buffer has been loaded with valid data.
<table>
<tr><th>PSTBWA[7]</th><th>PSTB Valid Data Locations
(Oldest to Newest)</th></tr>
<tr><td>0</td><td>0, 1, ... PSTBWA-1</td></tr>
<tr><td>1</td><td>PSTBWA, PSTBWA+1,..., 0, 1, PSTBWA-1</td></tr>
</table>
The PSTBWA is unaffected when a buffer stop condition has been reached, the buffer is disabled, or a system reset occurs. This allows the contents of the PST trace buffer to be retrieved after these events to assist in debug.
Note: Since this device contains a 64-entry trace buffer, PSTBWA[6] is always zero.</td></tr>
<tr><td>7
PSTBR</td><td>PST trace buffer reset. Generates a reset of the PST trace buffer logic, which clears PSTBWA and PSTBST. The same resources are reset when a disabled trace buffer becomes enabled and upon the receipt of a BDM GO command when operating in obtrusive trace mode. These reset events also clear any accumulation of PSTs. This bit always reads as a zero.
0 Do not force a PST trace buffer reset
1 Force a PST trace buffer reset</td></tr>
</table>

Table 26-9. CSR2 Field Descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>6
APCDIV16</td><td>Automatic PC synchronization divide cycle counts by 16. This bit divides the cycle counts for automatic SYNC_PC command insertion by 16. See the APCSC and APCENB field descriptions.</td></tr>
<tr><td>5</td><td>Reserved, must be cleared.</td></tr>
<tr><td>4–3
PSTBRM</td><td>PST trace buffer recording mode. Defines the trace buffer recording mode. The start and stop recording conditions are defined by the PSTBSS field.
00 Non-obstrusive, normal recording mode
01 Obtrusive, normal recording
10 Non-obtrusive, PC profile recording. Automatic PC synchronization must be enabled (see XCSR[APCSC, APCENB], CSR2[APCDIV16], and CSR[BTB]).
11 Obtrusive, PC profile recording. Automatic PC synchronization must be enabled (see XCSR[APCSC, APCENB], CSR2[APCDIV16], and CSR[BTB]).
The terms obtrusive and non-obtrusive are defined as:
• Non-obtrusive—The core is not halted. The PST trace buffer is overwritten unless a PSTB start/stop combination results in less than or equal to 64 PSTB captures.
• Obtrusive—The core is halted when the PSTB trace buffer reaches its full level (full before overwriting). The PSTB trace buffer contents are available by the BDM PSTB_READ commands. The PSTB trace buffer write address resets and the CPU resumes upon a BDM GO command.</td></tr>
<tr><td>2–0
PSTBSS</td><td>PST trace buffer start/stop definition. Specifies the start and stop conditions for PST trace buffer recording. In certain cases, the start and stop conditions are defined by the breakpoint registers. The remaining breakpoint registers are available for trigger configurations.
<table>
<tr><th>PSTBSS</th><th>Start Condition</th><th>Stop Condition</th></tr>
<tr><td>000</td><td colspan="2">Trace buffer disabled, no recording</td></tr>
<tr><td>001</td><td colspan="2">Unconditional recording</td></tr>
<tr><td>010</td><td rowspan="2">ABxR{& DBR/DBMR}</td><td>PBR0/PBMR</td></tr>
<tr><td>011</td><td>PBR1</td></tr>
<tr><td>100</td><td rowspan="2">PBR0/PBMR</td><td>ABxR{& DBR/DBMR}</td></tr>
<tr><td>101</td><td>PBR1</td></tr>
<tr><td>110</td><td rowspan="2">PBR1</td><td>ABxR{& DBR/DBMR}</td></tr>
<tr><td>111</td><td>PBR0/PBMR</td></tr>
</table></td></tr>
</table>

26.3.4 Configuration/Status Register 3 (CSR3)

CSR3 contains the BDM flash clock divider (BFCDIV) value in a format similar to HCS08 devices.

There are multiple ways to reference CSR3. They are summarized in Table 26-10.

Table 26-10. CSR3 Reference Summary

Method	Reference Details
READ_CSR3_BYTE	Reads CSR3[31–24] from the BDM interface. Available in all modes.
WRITE_CSR3_BYTE	Writes CSR3[31–24] from the BDM interface. Available in all modes.

Table 26-10. CSR3 Reference Summary (continued)

Method	Reference Details
READ_DREG	Reads CSR3[31—0] from the BDM interface. Classified as a non-intrusive BDM command.
WRITE_DREG	Writes CSR3[23—0] from the BDM interface. Classified as a non-intrusive BDM command.
WDEBUG Instruction	No operation during the core's execution of a WDEBUG instruction

DRc: 0x03 (CSR3)

Access: Supervisor write-only
BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	BFC DIV8	BFCDIV						0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 26-6. Configuration/Status Register 3 (CSR3)

Table 26-11. CSR3 Field Descriptions

Field	Description
31	Reserved, must be cleared.
30 BFCDIV8	BDM flash clock divide by 8. 0 Input to the flash clock divider is the bus clock 1 Input to the flash clock divider is the bus clock divided by 8
29–24 BFCDIV	BDM flash clock divider. The BFCDIV8 and BFCDIV fields specify the frequency of the internal flash clock when performing a mass erase operation initiated by setting XCSR[ERASE]. These fields must be loaded with the appropriate values prior to the setting of XCSR[ERASE] to initiate a mass erase operation in the flash memory. This field divides the bus clock (or the bus clock divided by 8 if BFCDIV8 is set) by the value defined by the BFCDIV plus one. The resulting frequency of the internal flash clock must fall within the range of 150–200 kHz for proper flash operations. Program/erase timing pulses are one cycle of this internal flash clock, which corresponds to a range of 5–6.7 μs. The automated programming logic uses an integer number of these pulses to complete an erase or program operation. if BFCDIV8 = 0, then $f_{FCLK} = f_{Bus} \div (BFCDIV + 1)$ if BFCDIV8 = 1, then $f_{FCLK} = f_{Bus} \div (8 \times (BFCDIV + 1)$ where f_{FCLK} is the frequency of the flash clock and f_{Bus} is the frequency of the bus clock.
23–0	Reserved for future use by the debug module, must be cleared.

26.3.5 BDM Address Attribute Register (BAAR)

BAAR defines the address space for memory-referencing BDM commands. BAAR[R, SZ] are loaded directly from the BDM command, while the lower five bits can be programmed from the external

development system. BAAR is loaded any time AATR is written and is initialized to a value of 0x05, setting supervisor data as the default address space. The upper 24 bits of this register are reserved for future use and any attempted write of these bits is ignored.

DRc: 0x05 (BAAR) Access: Supervisor write-only, BDM write-only

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

Figure 26-7. BDM Address Attribute Register (BAAR)

Table 26-12. BAAR Field Descriptions

Field	Description
31–8	Reserved for future use by the debug module, must be cleared.
7 R	Read/Write. 0 Write 1 Read
6–5 SZ	Size. 00 Longword 01 Byte 10 Word 11 Reserved
4–3 TT	Transfer type. See the TT definition in the AATR description, Section 26.3.6, “Address Attribute Trigger Register (AATR)”.
2–0 TM	Transfer modifier. See the TM definition in the AATR description, Section 26.3.6, “Address Attribute Trigger Register (AATR)”.

26.3.6 Address Attribute Trigger Register (AATR)

AATR defines address attributes and a mask to be matched in the trigger. The register value is compared with address attribute signals from the processor’s high-speed local bus, as defined by the setting of the trigger definition register (TDR). AATR is accessible in supervisor mode as debug control register 0x06 using the WDEBUG instruction and through the BDM port using the WRITE_DREG command.

DRc: 0x06 (AATR) Access: Supervisor write-only, BDM write-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RM	SZM		TTM		TMM			R	SZ			TT		TM	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Figure 26-8. Address Attribute Trigger Register (AATR)

Table 26-13. AATR Field Descriptions

Field	Description
31–16	Reserved, must be cleared.
15 RM	Read/write mask. Masks the R bit in address comparisons.
14–13 SZM	Size mask. Masks the corresponding SZ bit in address comparisons.
12–11 TTM	Transfer type mask. Masks the corresponding TT bit in address comparisons.
10–8 TMM	Transfer modifier mask. Masks the corresponding TM bit in address comparisons.
7 R	Read/write. R is compared with the R/$\overline{W}$ signal of the processor's local bus.
6–5 SZ	Size. Compared to the processor's local bus size signals. 00 Longword 01 Byte 10 Word 11 Reserved
4–3 TT	Transfer type. Compared with the local bus transfer type signals. These bits also define the TT encoding for BDM memory commands. 00 Normal processor access Else Reserved
2–0 TM	Transfer modifier. Compared with the local bus transfer modifier signals, which give supplemental information for each transfer type. These bits also define the TM encoding for BDM memory commands (for backward compatibility). 000 Reserved 001 User-mode data access 010 User-mode code access 011 Reserved 100 Reserved 101 Supervisor-mode data access 110 Supervisor-mode code access 111 Reserved

26.3.7 Trigger Definition Register (TDR)

TDR configures the operation of the hardware breakpoint logic that corresponds with the ABHR/ABLR/AATR, PBR/PBR1/PBR2/PBR3/PBMR, and DBR/DBMR registers within the debug module. TDR controls the actions taken under the defined conditions. Breakpoint logic may be configured as one- or two-level trigger. TDR[31–16] defines the second-level trigger, and TDR[15–0] defines the first-level trigger.

NOTE

The debug module has no hardware interlocks. To prevent spurious breakpoint triggers while the breakpoint registers are being loaded, disable TDR (clear TDR[L2EBL,L1EBL]) before defining triggers.

A write to TDR clears the CSR trigger status bits, CSR[BSTAT]. TDR is accessible in supervisor mode as debug control register 0x07 using the WDEBUG instruction and through the BDM port using the WRITE_DREG command.

DRc: 0x07 (TDR)

Access: Supervisor write-only
BDM write-only

Second Level Trigger

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	TRC		L2EBL	L2ED							L2DI	L2EA			L2EPC	L2PCI
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

First Level Trigger

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	L2T	L1T	L1EBL	L1ED							L1DI	L1EA			L1EPC	L1PCI
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-9. Trigger Definition Register (TDR)

Table 26-14. TDR Field Descriptions

Field	Description
31–30 TRC	Trigger response control. Determines how the processor responds to a completed trigger condition. The trigger response is displayed on PST. 00 Display on PST only 01 Processor halt 10 Debug interrupt 11 Reserved
29 L2EBL	Enable level 2 breakpoint. Global enable for the breakpoint trigger. 0 Disables all level 2 breakpoints 1 Enables all level 2 breakpoint triggers
28–22 L2ED	Enable level 2 data breakpoint. Setting an L2ED bit enables the corresponding data breakpoint condition based on the size and placement on the processor's local data bus. Clearing all ED bits disables data breakpoints.

TDR Bit	Description
28	Data longword. Entire processor's local data bus.
27	Lower data word.
26	Upper data word.
25	Lower lower data byte. Low-order byte of the low-order word.
24	Lower middle data byte. High-order byte of the low-order word.
23	Upper middle data byte. Low-order byte of the high-order word.
22	Upper upper data byte. High-order byte of the high-order word.

Table 26-14. TDR Field Descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>21
L2DI</td><td>Level 2 data breakpoint invert. Inverts the logical sense of all the data breakpoint comparators. This can develop a trigger based on the occurrence of a data value other than the DBR contents.
0 No inversion
1 Invert data breakpoint comparators.</td></tr>
<tr><td>20–18
L2EA</td><td>Enable level 2 address breakpoint. Setting an L2EA bit enables the corresponding address breakpoint. Clearing all three bits disables the breakpoint.
<table>
<tr><th>TDR Bit</th><th>Description</th></tr>
<tr><td>20</td><td>Address breakpoint inverted. Breakpoint is based outside the range between ABLR and ABHR.</td></tr>
<tr><td>19</td><td>Address breakpoint range. The breakpoint is based on the inclusive range defined by ABLR and ABHR.</td></tr>
<tr><td>18</td><td>Address breakpoint low. The breakpoint is based on the address in the ABLR.</td></tr>
</table></td></tr>
<tr><td>17
L2EPC</td><td>Enable level 2 PC breakpoint.
0 Disable PC breakpoint
1 Enable PC breakpoint</td></tr>
<tr><td>16
L2PCI</td><td>Level 2 PC breakpoint invert.
0 The PC breakpoint is defined within the region defined by PBRn and PBMR.
1 The PC breakpoint is defined outside the region defined by PBRn and PBMR.</td></tr>
<tr><td>15
L2T</td><td>Level 2 trigger. Determines the logic operation for the trigger between the PC_condition and the (Address_range and Data) condition where the inclusion of a Data_condition is optional. The ColdFire debug architecture supports the creation of single or double-level triggers.
0 Level 2 trigger = PC_condition && (Address_range && Data_condition)
1 Level 2 trigger = PC_condition || (Address_range && Data_condition)</td></tr>
<tr><td>14
L1T</td><td>Level 1 trigger. Determines the logic operation for the trigger between the PC_condition and the (Address_range and Data) condition where the inclusion of a Data_condition is optional. The ColdFire debug architecture supports the creation of single or double-level triggers.
0 Level 1 trigger = PC_condition && (Address_range && Data_condition)
1 Level 1 trigger = PC_condition || (Address_range && Data_condition)</td></tr>
<tr><td>13
L1EBL</td><td>Enable level 1 breakpoint. Global enable for the breakpoint trigger.
0 Disables all level 1 breakpoints
1 Enables all level 1 breakpoint triggers</td></tr>
</table>

Table 26-14. TDR Field Descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>12–6
L1ED</td><td>Enable level 1 data breakpoint. Setting an L1ED bit enables the corresponding data breakpoint condition based on the size and placement on the processor's local data bus. Clearing all L1ED bits disables data breakpoints.
<table>
<tr><th>TDR Bit</th><th>Description</th></tr>
<tr><td>12</td><td>Data longword. Entire processor's local data bus.</td></tr>
<tr><td>11</td><td>Lower data word.</td></tr>
<tr><td>10</td><td>Upper data word.</td></tr>
<tr><td>9</td><td>Lower lower data byte. Low-order byte of the low-order word.</td></tr>
<tr><td>8</td><td>Lower middle data byte. High-order byte of the low-order word.</td></tr>
<tr><td>7</td><td>Upper middle data byte. Low-order byte of the high-order word.</td></tr>
<tr><td>6</td><td>Upper upper data byte. High-order byte of the high-order word.</td></tr>
</table></td></tr>
<tr><td>5
L1DI</td><td>Level 1 data breakpoint invert. Inverts the logical sense of all the data breakpoint comparators. This can develop a trigger based on the occurrence of a data value other than the DBR contents.
0 No inversion
1 Invert data breakpoint comparators.</td></tr>
<tr><td>4–2
L1EA</td><td>Enable level 1 address breakpoint. Setting an L1EA bit enables the corresponding address breakpoint. Clearing all three bits disables the address breakpoint.
<table>
<tr><th>TDR Bit</th><th>Description</th></tr>
<tr><td>4</td><td>Enable address breakpoint inverted. Breakpoint is based outside the range between ABLR and ABHR.</td></tr>
<tr><td>3</td><td>Enable address breakpoint range. The breakpoint is based on the inclusive range defined by ABLR and ABHR.</td></tr>
<tr><td>2</td><td>Enable address breakpoint low. The breakpoint is based on the address in the ABLR.</td></tr>
</table></td></tr>
<tr><td>1
L1EPC</td><td>Enable level 1 PC breakpoint.
0 Disable PC breakpoint
1 Enable PC breakpoint</td></tr>
<tr><td>0
L1PCI</td><td>Level 1 PC breakpoint invert.
0 The PC breakpoint is defined within the region defined by PBRn and PBMR.
1 The PC breakpoint is defined outside the region defined by PBRn and PBMR.</td></tr>
</table>

26.3.8 Program Counter Breakpoint/Mask Registers (PBR0–3, PBMR)

The PBR*n* registers define instruction addresses for use as part of the trigger. These registers' contents are compared with the processor's program counter register when the appropriate valid bit is set (for PBR1–3) and TDR is configured appropriately. PBR0 bits are masked by setting corresponding PBMR bits (PBMR has no effect on PBR1–3). Results are compared with the processor's program counter register, as defined in TDR. The PC breakpoint registers, PBR1–3, have no masking associated with them, but do include a

valid bit. These registers' contents are compared with the processor's program counter register when TDR is configured appropriately.

The PC breakpoint registers are accessible in supervisor mode using the WDEBUG instruction and through the BDM port using the WRITE_DREG command using values shown in Section 26.4.1.4, "BDM Command Set Descriptions".

NOTE

Version 1 ColdFire core devices implement a 24-bit, 16 MB address map. When programming these registers with a 32-bit address, the upper byte must be zero-filled.

DRc: 0x08 (PBR0) Access: Supervisor write-only, BDM write-only

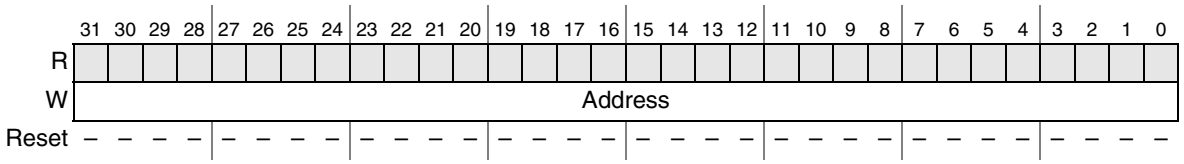


Figure 26-10. Program Counter Breakpoint Register 0 (PBR0)

Table 26-15. PBR0 Field Descriptions

Field	Description
31–0 Address	PC breakpoint address. The address to be compared with the PC as a breakpoint trigger. Because all instruction sizes are multiples of 2 bytes, bit 0 of the address should always be zero.

DRc: 0x18 (PBR1)
0x1A (PBR2)
0x1C (PBR3) Access: Supervisor write-only, BDM write-only

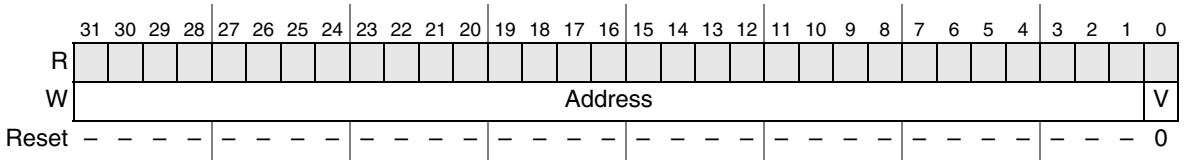


Figure 26-11. Program Counter Breakpoint Register *n* (PBR*n*, *n = 1,2,3*)

Table 26-16. PBR*n* Field Descriptions

Field	Description
31–1 Address	PC breakpoint address. The 31-bit address to be compared with the PC as a breakpoint trigger.
0 V	Valid bit. This bit must be set for the PC breakpoint to occur at the address specified in the Address field. 0 PBR is disabled. 1 PBR is enabled.

Figure 26-12 shows PBMR. PBMR is accessible in supervisor mode using the WDEBUG instruction and via the BDM port using the WRITE_DREG command. PBMR only masks PBR0.

DRc: 0x09 (PBMR) Access: Supervisor write-only
BDM write-only

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

Figure 26-12. Program Counter Breakpoint Mask Register (PBMR)

Table 26-17. PBMR Field Descriptions

Field	Description
31–0 Mask	PC breakpoint mask. If using PBR0, this register must be initialized since it is undefined after reset. 0 The corresponding PBR0 bit is compared to the appropriate PC bit. 1 The corresponding PBR0 bit is ignored.

26.3.9 Address Breakpoint Registers (ABLR, ABHR)

The ABLR and ABHR define regions in the processor's data address space that can be used as part of the trigger. These register values are compared with the address for each transfer on the processor's high-speed local bus. The trigger definition register (TDR) identifies the trigger as one of three cases:

- Identical to the value in ABLR
- Inside the range bound by ABLR and ABHR inclusive
- Outside that same range

The address breakpoint registers are accessible in supervisor mode using the WDEBUG instruction and through the BDM port using the WRITE_DREG command using values shown in Section 26.4.1.4, "BDM Command Set Descriptions."

NOTE

Version 1 ColdFire core devices implement a 24-bit, 16 MB address map. When programming these registers with a 32-bit address, the upper byte must be zero-filled.

DRc: 0x0C (ABHR)
0x0D (ABLR)
Access: Supervisor write-only
BDM write-only

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

Figure 26-13. Address Breakpoint Registers (ABLR, ABHR)

Table 26-18. ABLR Field Description

Field	Description
31–0 Address	Low address. Holds the 32-bit address marking the lower bound of the address breakpoint range. Breakpoints for specific addresses are programmed into ABLR.

Table 26-19. ABHR Field Description

Field	Description
31–0 Address	High address. Holds the 32-bit address marking the upper bound of the address breakpoint range.

26.3.10 Data Breakpoint and Mask Registers (DBR, DBMR)

DBR specifies data patterns used as part of the trigger into debug mode. DBR bits are masked by setting corresponding DBMR bits, as defined in TDR.

DBR and DBMR are accessible in supervisor mode using the WDEBUG instruction and through the BDM port using the WRITE_DREG commands.

DRc[4:0]: 0x0E (DBR)
0x0F (DBMR)

Access: Supervisor write-only
BDM write-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																																
W	Data (DBR); Mask (DBMR)																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-14. Data Breakpoint & Mask Registers (DBR, DBMR)

Table 26-20. DBR Field Descriptions

Field	Description
31–0 Data	Data breakpoint value. Contains the value to be compared with the data value from the processor's local bus as a breakpoint trigger.

Table 26-21. DBMR Field Descriptions

Field	Description
31–0 Mask	Data breakpoint mask. The 32-bit mask for the data breakpoint trigger. 0 The corresponding DBR bit is compared to the appropriate bit of the processor's local data bus 1 The corresponding DBR bit is ignored

The DBR supports aligned and misaligned references. Table 26-22 shows the relationships between processor address, access size, and location within the 32-bit data bus.

Table 26-22. Access Size and Operand Data Location

Address[1–0]	Access Size	Operand Location
00	Byte	D[31–24]
01	Byte	D[23–16]
10	Byte	D[15–8]
11	Byte	D[7–0]
0*x*	Word	D[31–16]
1*x*	Word	D[15–0]
xx	Longword	D[31–0]

26.3.10.1 Resulting Set of Possible Trigger Combinations

The resulting set of possible breakpoint trigger combinations consists of the following options where || denotes logical OR, && denotes logical AND, and {} denotes an optional additional trigger term:

One-level triggers of the form:

```
if      (PC_breakpoint)
if      (PC_breakpoint || Address_breakpoint{&& Data_breakpoint})
if      (Address_breakpoint {&& Data_breakpoint})
```

Two-level triggers of the form:

```
if      (PC_breakpoint)
        then if (Address_breakpoint{&& Data_breakpoint})

if      (Address_breakpoint {&& Data_breakpoint})
        then if (PC_breakpoint)
```

In these examples, PC_breakpoint is the logical summation of the PBR0/PBMR, PBR1, PBR2, and PBR3 breakpoint registers; Address_breakpoint is a function of ABHR, ABLR, and AATR; Data_breakpoint is a function of DBR and DBMR. In all cases, the data breakpoints can be included with an address breakpoint to further qualify a trigger event as an option.

The breakpoint registers can also be used to define the start and stop recording conditions for the PST trace buffer. For information on this functionality, see Section 26.3.3, “Configuration/Status Register 2 (CSR2)”.

26.3.11 PST Buffer (PSTB)

The PST trace buffer contains 64 six-bit entries, packed consecutively into 12 longword locations. See Figure 26-15 for an illustration of how the buffer entries are packed.

The write pointer for the trace buffer is available as CSR2[PSTBWA]. Using this pointer, it is possible to determine the oldest-to-newest entries in the trace buffer.

Core register number (CRN)	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
0x10	TB #00		TB #01		TB #02		TB #03	TB #04	05[5:4]
0x11	TB #05[3:0]		TB #06		TB #07		TB #08	TB #09	TB #10[5:2]
0x12	10[1:0]	TB #11		TB #12		TB #13		TB #14	TB #15
0x13	TB #16		TB #17		TB #18		TB #19	TB #20	21[5:4]
0x14	TB #21[3:0]		TB #22		TB #23		TB #24	TB #25	TB #26[5:2]
0x15	26[1:0]	TB #27		TB #28		TB #29		TB #30	TB #31
0x16	TB #32		TB #33		TB #34		TB #35	TB #36	37[5:4]
0x17	TB #37[3:0]		TB #38		TB #39		TB #40	TB #41	TB #42[5:2]
0x18	42[1:0]	TB #43		TB #44		TB #45		TB #46	TB #47
0x19	TB #48		TB #49		TB #50		TB #51	TB #52	53[5:4]
0x1A	TB #53[3:0]		TB #54		TB #55		TB #56	TB #57	TB #58[5:2]
0x1B	58[1:0]	TB #59		TB #60		TB #61		TB #62	TB #63

Figure 26-15. PST Trace Buffer Entries and Locations

26.4 Functional Description

26.4.1 Background Debug Mode (BDM)

This section provides details on the background debug serial interface controller (BDC) and the BDM command set.

The BDC provides a single-wire debug interface to the target MCU. As shown in the Version 1 ColdFire core block diagram of Figure 26-1, the BDC module interfaces between the single-pin (BKGD) interface and the remaining debug modules, including the ColdFire background debug logic, the real-time debug hardware, and the PST/DDATA trace logic. This interface provides a convenient means for programming the on-chip flash and other non-volatile memories. The BDC is the primary debug interface for development and allows non-intrusive access to memory data and traditional debug features such as run/halt control, read/write of core registers, breakpoints, and single instruction step.

Features of the background debug controller (BDC) include:

- Single dedicated pin for mode selection and background communications
- Special BDC registers not located in system memory map
- SYNC command to determine target communications rate
- Non-intrusive commands for memory access
- Active background (halt) mode commands for core register access
- GO command to resume execution
- BACKGROUND command to halt core or wake CPU from low-power modes
- Oscillator runs in stop mode, if BDM enabled

Based on these features, BDM is useful for the following reasons:

- In-circuit emulation is not needed, so physical and electrical characteristics of the system are not affected.
- BDM is always available for debugging the system and provides a communication link for upgrading firmware in existing systems.
- Provides high-speed memory downloading, especially useful for flash programming
- Provides absolute control of the processor, and thus the system. This feature allows quick hardware debugging with the same tool set used for firmware development.

26.4.1.1 CPU Halt

Although certain BDM operations can occur in parallel with CPU operations, unrestricted BDM operation requires the CPU to be halted. The sources that can cause the CPU to halt are listed below in order of priority. Recall that the default configuration of the Version 1 ColdFire core (CF1Core) defines the occurrence of certain exception types to automatically generate a system reset. Some of these fault types include illegal instructions, privilege errors, address errors, and bus error terminations, with the response under control of the processor's CPUCR[ARD, IRD] bits.

Table 26-23. CPU Halt Sources

<table>
<tr><th>Halt Source</th><th>Halt Timing</th><th colspan="3">Description</th></tr>
<tr><td rowspan="3">Fault-on-fault</td><td rowspan="3">Immediate</td><td colspan="3">Refers to the occurrence of any fault while exception processing. For example, a bus error is signaled during exception stack frame writes or while fetching the first instruction in the exception service routine.</td></tr>
<tr><td>CPUCR[ARD] = 1</td><td colspan="2">Immediately enters halt.</td></tr>
<tr><td>CPUCR[ARD] = 0</td><td colspan="2">Reset event is initiated.</td></tr>
<tr><td>Hardware breakpoint trigger</td><td>Pending</td><td colspan="3">Halt is made pending in the processor. The processor samples for pending halt and interrupt conditions once per instruction. When a pending condition is asserted, the processor halts execution at the next sample point.</td></tr>
<tr><td rowspan="6">HALT instruction</td><td rowspan="6">Immediate</td><td rowspan="2">BDM disabled</td><td>CPUCR[IRD] = 0</td><td>A reset is initiated since attempted execution of an illegal instruction</td></tr>
<tr><td>CPUCR[IRD] = 1</td><td>An illegal instruction exception is generated.</td></tr>
<tr><td>BDM enabled, supervisor mode</td><td colspan="2">Processor immediately halts execution at the next instruction sample point. The processor can be restarted by a BDM GO command. Execution continues at the instruction after HALT.</td></tr>
<tr><td rowspan="3">BDM enabled, user mode</td><td>CSR[UHE] = 0
CPUCR[IRD] = 0</td><td>A reset event is initiated, because a privileged instruction was attempted in user mode.</td></tr>
<tr><td>CSR[UHE] = 0
CPUCR[IRD] = 1</td><td>A privilege violation exception is generated.</td></tr>
<tr><td>CSR[UHE] = 1</td><td>Processor immediately halts execution at the next instruction sample point. The processor can be restarted by a BDM GO command. Execution continues at the instruction after HALT.</td></tr>
</table>

Table 26-23. CPU Halt Sources (continued)

Halt Source	Halt Timing	Description		
BACKGROUND command	Pending	BDM disabled or flash secure	Illegal command response and BACKGROUND command is ignored.	
		BDM enabled and flash unsecure	Processor is running	Halt is made pending in the processor. The processor samples for pending halt and interrupt conditions once per instruction. When a pending condition is asserted, the processor halts execution at the next sample point.
			Processor is stopped	Processing of the BACKGROUND command is treated in a special manner. The processor exits the stopped mode and enters the halted state, at which point all BDM commands may be exercised. When restarted, the processor continues by executing the next sequential instruction (the instruction following STOP).
PSTB full condition	Pending	PSTB	PSTB obtrusive recording mode pends halt in the processor if the trace buffer reaches its full threshold (full is defined as before the buffer is overwritten). When a pending condition is asserted, the processor halts at the next sample point.	
BKGD held low for ≥2 bus clocks after reset negated for POR or BDM reset	Immediate	Flash unsecure	Enters debug mode with XCSR[ENBDM, CLKSW] set. The full set of BDM commands is available and debug can proceed. If the core is reset into a debug halt condition, the processor's response to the GO command depends on the BDM command(s) performed while it was halted. Specifically, if the PC register was loaded, the GO command causes the processor to exit halted state and pass control to the instruction address in the PC, bypassing normal reset exception processing. If the PC was not loaded, the GO command causes the processor to exit halted state and continue reset exception processing.	
		Flash secure	Enters debug mode with XCSR[ENBDM, CLKSW] set. The allowable commands are limited to the always-available group. A GO command to start the processor is not allowed. The only recovery actions in this mode are: • Issue a BDM reset setting CSR2[BDFR] with CSR2[BDHBR] cleared and the BKGD pin held high to reset into normal operating mode • Erase the flash to unsecure the memory and then proceed with debug • Power cycle the device with the BKGD pin held high to reset into the normal operating mode	

The processor's run/stop/halt status is always accessible in XCSR[CPUHALT,CPUSTOP]. Additionally, CSR[27–24] indicate the halt source, showing the highest priority source for multiple halt conditions. This field is cleared by a read of the CSR. A processor halt due to the PSTB full condition as indicated by CSR2[PSTH] is also reflected in CSR[BKPT]. The debug GO command clears CSR[26–24] and CSR2[PSTBH].

26.4.1.2 Background Debug Serial Interface Controller (BDC)

BDC serial communications use a custom serial protocol first introduced on the M68HC12 Family of microcontrollers and later used in the M68HCS08 family. This protocol assumes that the host knows the

communication clock rate determined by the target BDC clock rate. The BDC clock rate may be the system bus clock frequency or an alternate frequency source depending on the state of XCSR[CLKSW]. All communication is initiated and controlled by the host which drives a high-to-low edge to signal the beginning of each bit time. Commands and data are sent most significant bit (msb) first. For a detailed description of the communications protocol, refer to Section 26.4.1.3, “BDM Communication Details”.

If a host is attempting to communicate with a target MCU that has an unknown BDC clock rate, a SYNC command may be sent to the target MCU to request a timed synchronization response signal from which the host can determine the correct communication speed. After establishing communications, the host can read XCSR and write the clock switch (CLKSW) control bit to change the source of the BDC clock for further serial communications if necessary.

BKGD is a pseudo-open-drain pin and there is an on-chip pullup so no external pullup resistor is required. Unlike typical open-drain pins, the external RC time constant on this pin, which is influenced by external capacitance, plays almost no role in signal rise time. The custom protocol provides for brief, actively driven speed-up pulses to force rapid rise times on this pin without risking harmful drive level conflicts. Refer to Section 26.4.1.3, “BDM Communication Details,” for more details.

When no debugger pod is connected to the standard 6-pin BDM interface connector (Section 26.4.4, “Freescale-Recommended BDM Pinout”), the internal pullup on BKGD chooses normal operating mode. When a development system is connected, it can pull BKGD and $\overline{\text{RESET}}$ low, release $\overline{\text{RESET}}$ to select active background (halt) mode rather than normal operating mode, and then release BKGD. It is not necessary to reset the target MCU to communicate with it through the background debug interface. There is also a mechanism to generate a reset event in response to setting CSR2[BDFR].

26.4.1.3 BDM Communication Details

The BDC serial interface requires the external host controller to generate a falling edge on the BKGD pin to indicate the start of each bit time. The external controller provides this falling edge whether data is transmitted or received.

BKGD is a pseudo-open-drain pin that can be driven by an external controller or by the MCU. Data is transferred msb first at 16 BDC clock cycles per bit (nominal speed). The interface times-out if 512 BDC clock cycles occur between falling edges from the host. If a time-out occurs, the status of any command in progress must be determined before new commands can be sent from the host. To check the status of the command, follow the steps detailed in the bit description of XCSR[CSTAT] in Table 26-7.

The custom serial protocol requires the debug pod to know the target BDC communication clock speed. The clock switch (CLKSW) control bit in the XCSR[31–24] register allows you to select the BDC clock source. The BDC clock source can be the bus clock or the alternate BDC clock source. When the MCU is reset in normal user mode, CLKSW is cleared and that selects the alternate clock source. This clock source is a fixed frequency independent of the bus frequency so it does change if the user modifies clock generator settings. This is the preferred clock source for general debugging.

When the MCU is reset in active background (halt) mode, CLKSW is set which selects the bus clock as the source of the BDC clock. This CLKSW setting is most commonly used during flash memory programming because the bus clock can usually be configured to operate at the highest allowed bus frequency to ensure the fastest possible flash programming times. Because the host system is in control of

changes to clock generator settings, it knows when a different BDC communication speed should be used. The host programmer also knows that no unexpected change in bus frequency could occur to disrupt BDC communications.

Normally, setting CLKSW should not be used for general debugging because there is no way to ensure the application program does not change the clock generator settings. This is especially true in the case of application programs that are not yet fully debugged.

After any reset (or at any other time), the host system can issue a SYNC command to determine the speed of the BDC clock. CLKSW may be written using the serial WRITE_XCSR_BYTE command through the BDC interface. CLKSW is located in the special XCSR byte register in the BDC module and it is not accessible in the normal memory map of the ColdFire core. This means that no program running on the processor can modify this register (intentionally or unintentionally).

The BKGD pin can receive a high- or low-level or transmit a high- or low-level. The following diagrams show timing for each of these cases. Interface timing is synchronous to clocks in the target BDC, but asynchronous to the external host. The internal BDC clock signal is shown for reference in counting cycles.

Figure 26-16 shows an external host transmitting a logic 1 or 0 to the BKGD pin of a target MCU. The host is asynchronous to the target so there is a 0–1 cycle delay from the host-generated falling edge to where the target perceives the beginning of the bit time. Ten target BDC clock cycles later, the target senses the bit level on the BKGD pin. Typically, the host actively drives the pseudo-open-drain BKGD pin during host-to-target transmissions to speed up rising edges. Because the target does not drive the BKGD pin during the host-to-target transmission period, there is no need to treat the line as an open-drain signal during this period.

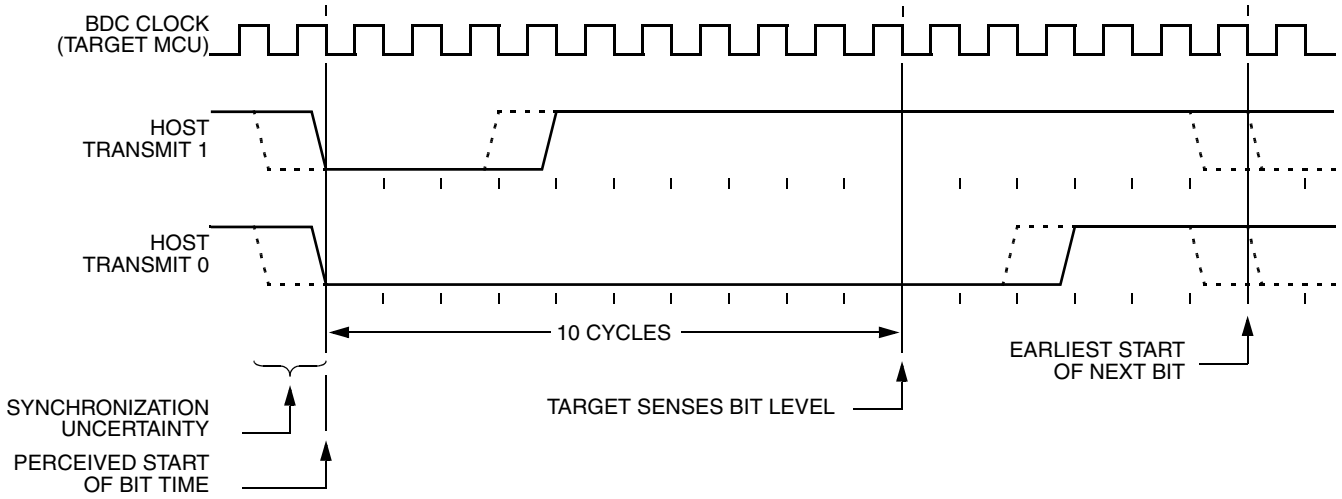


Figure 26-16. BDC Host-to-Target Serial Bit Timing

Figure 26-17 shows the host receiving a logic 1 from the target MCU. Because the host is asynchronous to the target MCU, there is a 0–1 cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target MCU. The host holds the BKGD pin low long enough for the target to recognize it (at least two target BDC cycles). The host must release the low drive before the target

MCU drives a brief active-high speedup pulse seven cycles after the perceived start of the bit time. The host should sample the bit level about 10 cycles after it started the bit time.

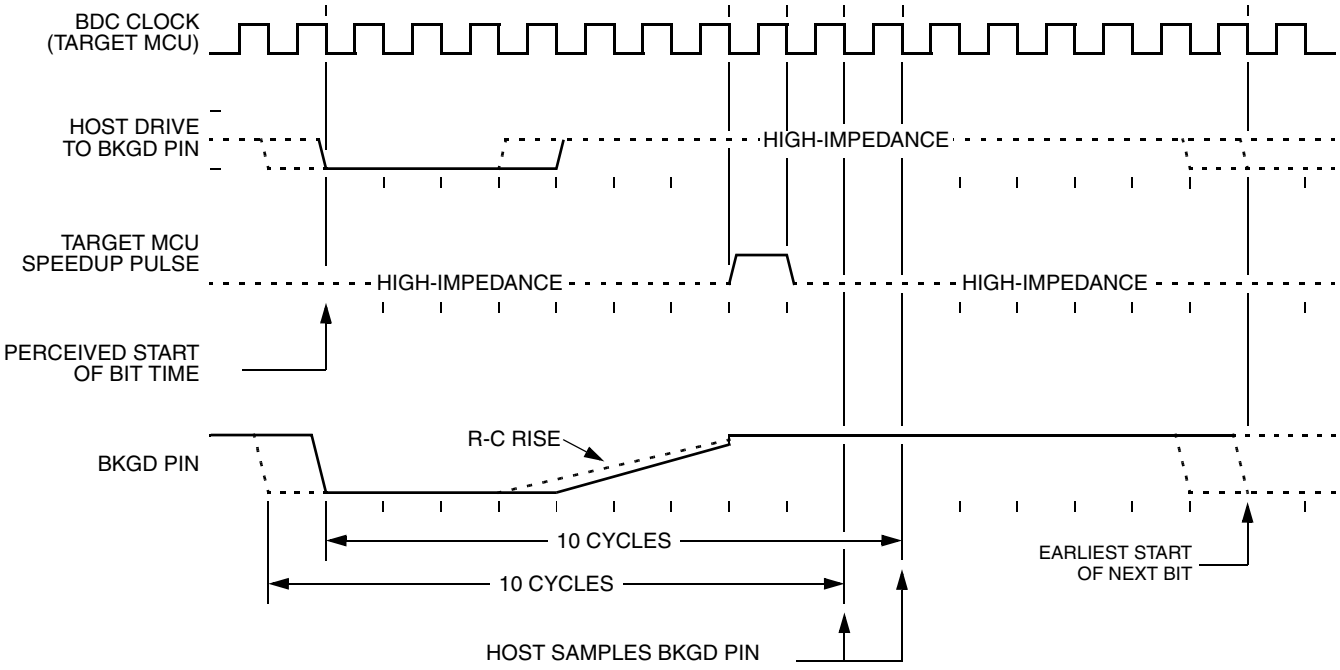


Figure 26-17. BDC Target-to-Host Serial Bit Timing (Logic 1)

Figure 26-18 shows the host receiving a logic 0 from the target MCU. Because the host is asynchronous to the target MCU, there is a 0–1 cycle delay from the host-generated falling edge on BKGD to the start of the bit time as perceived by the target MCU. The host initiates the bit time, but the target MCU finishes it. Because the target wants the host to receive a logic 0, it drives the BKGD pin low for 13 BDC clock cycles, then briefly drives it high to speed up the rising edge. The host samples the bit level about 10 cycles after starting the bit time.

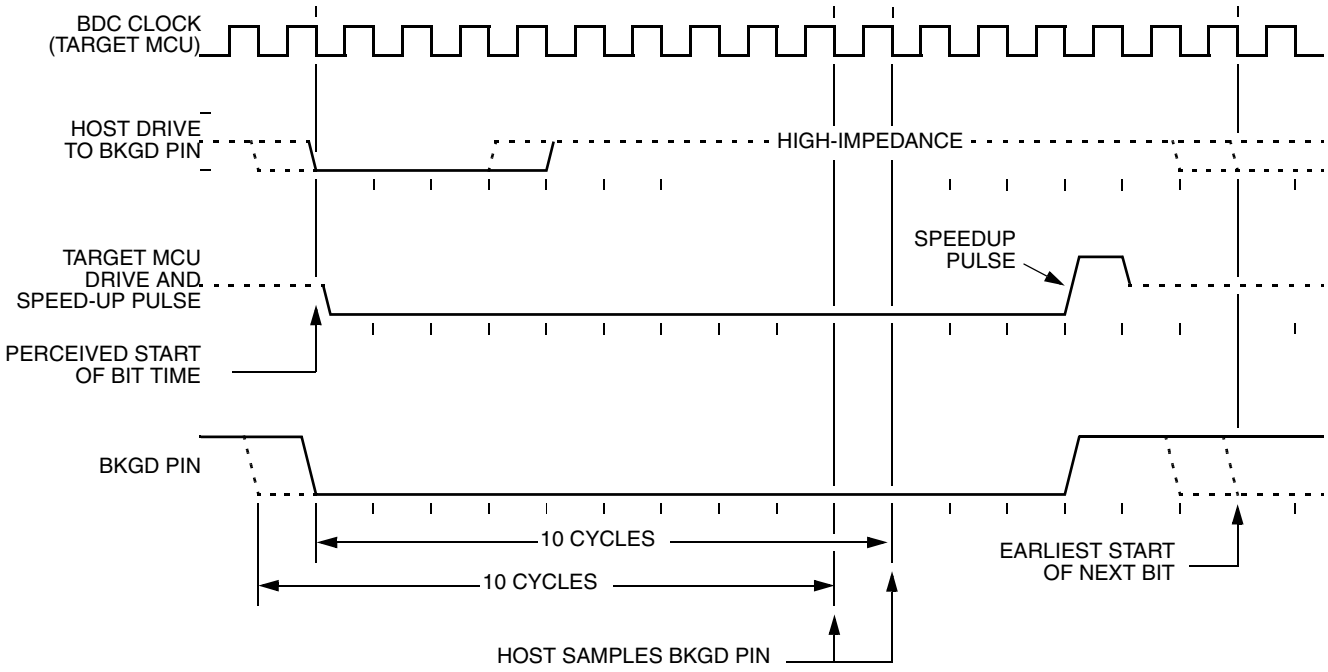


Figure 26-18. BDM Target-to-Host Serial Bit Timing (Logic 0)

26.4.1.4 BDM Command Set Descriptions

This section presents detailed descriptions of the BDM commands.

The V1 BDM command set is based on transmission of one or more 8-bit data packets per operation. Each operation begins with a host-to-target transmission of an 8-bit command code packet. The command code definition broadly maps the operations into four formats as shown in Figure 26-19.

Miscellaneous Commands

	7	6	5	4	3	2	1	0
W	0	0	R/$\overline{W}$	0	MSCMD			
R/W	Optional Command Extension Byte (Data)							

Memory Commands

	7	6	5	4	3	2	1	0
W	0	0	R/$\overline{W}$	1	SZ		MCMD	
W if addr, R/W if data	Command Extension Bytes (Address, Data)							

Core Register Commands

	7	6	5	4	3	2	1	0
W	CRG		R/$\overline{W}$	CRN				
R/W	Command Extension Bytes (Data)							

PST Trace Buffer Read Commands

	7	6	5	4	3	2	1	0
W	0	1	0	CRN				
R	Trace Buffer Data[31–24], see Figure 26-15							
R	Trace Buffer Data[23–16], see Figure 26-15							
R	Trace Buffer Data[15–08], see Figure 26-15							
R	Trace Buffer Data[07–00], see Figure 26-15							

Figure 26-19. BDM Command Encodings

Table 26-24. BDM Command Field Descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>5
R/$\overline{W}$</td><td>Read/$\overline{Write}$.
0 Command is performing a write operation.
1 Command is performing a read operation.</td></tr>
<tr><td>3–0
MSCMD</td><td>Miscellaneous command. Defines the miscellaneous command to be performed.
0000 No operation
0001 Display the CPU's program counter (PC) plus optional capture in the PST trace buffer
0010 Enable the BDM acknowledge communication mode
0011 Disable the BDM acknowledge communication mode
0100 Force a CPU halt (background)
1000 Resume CPU execution (go)
1101 Read/write of the debug XCSR most significant byte
1110 Read/write of the debug CSR2 most significant byte
1111 Read/write of the debug CSR3 most significant byte</td></tr>
<tr><td>3–2
SZ</td><td>Memory operand size. Defines the size of the memory reference.
00 8-bit byte
01 16-bit word
10 32-bit long</td></tr>
<tr><td>1–0
MCMD</td><td>Memory command. Defines the type of the memory reference to be performed.
00 Simple write if R/$\overline{W}$ = 0; simple read if R/$\overline{W}$ = 1
01 Write + status if R/$\overline{W}$ = 0; read + status if R/$\overline{W}$ = 1
10 Fill if R/$\overline{W}$ = 0; dump if R/$\overline{W}$ = 1
11 Fill + status if R/$\overline{W}$ = 0; dump + status if R/$\overline{W}$ = 1</td></tr>
<tr><td>7–6
CRG</td><td>Core register group. Defines the core register group to be referenced.
01 CPU's general-purpose registers (An, Dn) or PST trace buffer
10 Debug's control registers
11 CPU's control registers (PC, SR, VBR, CPUCR,...)</td></tr>
<tr><td>4–0
CRN</td><td>Core register number. Defines the specific core register (its number) to be referenced. All other CRN values are reserved.
<table>
<tr><th>CRG</th><th>CRN</th><th>Register</th></tr>
<tr><td rowspan="3">01</td><td>0x00–0x07</td><td>D0–7</td></tr>
<tr><td>0x08–0x0F</td><td>A0–7</td></tr>
<tr><td>0x10–0x1B</td><td>PST Buffer 0–11</td></tr>
<tr><td>10</td><td colspan="2">DRc[4:0] as described in Table 26-4</td></tr>
<tr><td rowspan="8">11</td><td>0x00</td><td>OTHER_A7</td></tr>
<tr><td>0x01</td><td>VBR</td></tr>
<tr><td>0x02</td><td>CPUCR</td></tr>
<tr><td>0x04</td><td>MACSR</td></tr>
<tr><td>0x05</td><td>MASK</td></tr>
<tr><td>0x06</td><td>ACC</td></tr>
<tr><td>0x0E</td><td>SR</td></tr>
<tr><td>0x0F</td><td>PC</td></tr>
</table>
</td></tr>
</table>

26.4.1.5 BDM Command Set Summary

Table 26-25 summarizes the BDM command set. Subsequent paragraphs contain detailed descriptions of each command. The nomenclature below is used in Table 26-25 to describe the structure of the BDM commands.

Commands begin with an 8-bit hexadecimal command code in the host-to-target direction (most significant bit first)

```
/       =       separates parts of the command
d       =       delay 32 target BDC clock cycles
ad24    =       24-bit memory address in the host-to-target direction
rd8     =       8 bits of read data in the target-to-host direction
rd16    =       16 bits of read data in the target-to-host direction
rd32    =       32 bits of read data in the target-to-host direction
rd.sz   =       read data, size defined by sz, in the target-to-host direction
wd8     =       8 bits of write data in the host-to-target direction
wd16    =       16 bits of write data in the host-to-target direction
wd32    =       32 bits of write data in the host-to-target direction
wd.sz   =       write data, size defined by sz, in the host-to-target direction
ss      =       the contents of XCSR[31:24] in the target-to-host direction (STATUS)
sz      =       memory operand size (0b00 = byte, 0b01 = word, 0b10 = long)
crn     =       core register number
WS      =       command suffix signaling the operation is with status
```

Table 26-25. BDM Command Summary

Command Mnemonic	Command Classification	ACK if Enb?[1]	Command Structure	Description
SYNC	Always Available	N/A	N/A[2]	Request a timed reference pulse to determine the target BDC communication speed
ACK_DISABLE	Always Available	No	0x03/d	Disable the communication handshake. This command does not issue an ACK pulse.
ACK_ENABLE	Always Available	Yes	0x02/d	Enable the communication handshake. Issues an ACK pulse after the command is executed.
BACKGROUND	Non-Intrusive	Yes	0x04/d	Halt the CPU if ENBDM is set. Otherwise, ignore as illegal command.
DUMP_MEM.sz	Non-Intrusive	Yes	(0x32+4 x sz)/d/rd.sz	Dump (read) memory based on operand size (sz). Used with READ_MEM to dump large blocks of memory. An initial READ_MEM is executed to set up the starting address of the block and to retrieve the first result. Subsequent DUMP_MEM commands retrieve sequential operands.

Table 26-25. BDM Command Summary (continued)

Command Mnemonic	Command Classification	ACK if Enb?[1]	Command Structure	Description
DUMP_MEM.sz_WS	Non-Intrusive	No	(0x33+4 x sz)/d/ss/rd.sz	Dump (read) memory based on operand size (sz) and report status. Used with READ_MEM{_WS} to dump large blocks of memory. An initial READ_MEM{_WS} is executed to set up the starting address of the block and to retrieve the first result. Subsequent DUMP_MEM{_WS} commands retrieve sequential operands.
FILL_MEM.sz	Non-Intrusive	Yes	(0x12+4 x sz)/wd.sz/d	Fill (write) memory based on operand size (sz). Used with WRITE_MEM to fill large blocks of memory. An initial WRITE_MEM is executed to set up the starting address of the block and to write the first operand. Subsequent FILL_MEM commands write sequential operands.
FILL_MEM.sz_WS	Non-Intrusive	No	(0x13+4 x sz)/wd.sz/d/ss	Fill (write) memory based on operand size (sz) and report status. Used with WRITE_MEM{_WS} to fill large blocks of memory. An initial WRITE_MEM{_WS} is executed to set up the starting address of the block and to write the first operand. Subsequent FILL_MEM{_WS} commands write sequential operands.
GO	Non-Intrusive	Yes	0x08/d	Resume the CPU's execution[3]
NOP	Non-Intrusive	Yes	0x00/d	No operation
READ_CREG	Active Background	Yes	(0xE0+CRN)/d/rd32	Read one of the CPU's control registers
READ_DREG	Non-Intrusive	Yes	(0xA0+CRN)/d/rd32	Read one of the debug module's control registers
READ_MEM.sz	Non-Intrusive	Yes	(0x30+4 x sz)/ad24/d/rd.sz	Read the appropriately-sized (sz) memory value from the location specified by the 24-bit address
READ_MEM.sz_WS	Non-Intrusive	No	(0x31+4 x sz)/ad24/d/ss/rd.sz	Read the appropriately-sized (sz) memory value from the location specified by the 24-bit address and report status
READ_PSTB	Non-Intrusive	Yes	(0x40+CRN)/d/rd32	Read the requested longword location from the PST trace buffer
READ_Rn	Active Background	Yes	(0x60+CRN)/d/rd32	Read the requested general-purpose register (An, Dn) from the CPU
READ_XCSR_BYTE	Always Available	No	0x2D/rd8	Read the most significant byte of the debug module's XCSR
READ_CSR2_BYTE	Always Available	No	0x2E/rd8	Read the most significant byte of the debug module's CSR2
READ_CSR3_BYTE	Always Available	No	0x2F/rd8	Read the most significant byte of the debug module's CSR3

Table 26-25. BDM Command Summary (continued)

Command Mnemonic	Command Classification	ACK if Enb?[1]	Command Structure	Description
SYNC_PC	Non-Intrusive	Yes	0x01/d	Display the CPU's current PC and capture it in the PST trace buffer
WRITE_CREG	Active Background	Yes	(0xC0+CRN)/wd32/d	Write one of the CPU's control registers
WRITE_DREG	Non-Intrusive	Yes	(0x80+CRN)/wd32/d	Write one of the debug module's control registers
WRITE_MEM.sz	Non-Intrusive	Yes	(0x10+4 x sz)/ad24/wd.sz/d	Write the appropriately-sized (sz) memory value to the location specified by the 24-bit address
WRITE_MEM.sz_WS	Non-Intrusive	No	(0x11+4 x sz)/ad24/wd.sz/d/ss	Write the appropriately-sized (sz) memory value to the location specified by the 24-bit address and report status
WRITE_Rn	Active Background	Yes	(0x40+CRN)/wd32/d	Write the requested general-purpose register (An, Dn) of the CPU
WRITE_XCSR_BYTE	Always Available	No	0x0D/wd8	Write the most significant byte of the debug module's XCSR
WRITE_CSR2_BYTE	Always Available	No	0x0E/wd8	Write the most significant byte of the debug module's CSR2
WRITE_CSR3_BYTE	Always Available	No	0x0F/wd8	Write the most significant byte of the debug module's CSR3

1 This column identifies if the command generates an ACK pulse if operating with acknowledge mode enabled. See Section 26.4.1.5.3, "ACK_ENABLE," for addition information.

2 The SYNC command is a special operation which does not have a command code.

3 If a GO command is received while the processor is not halted, it performs no operation.

26.4.1.5.1 SYNC

The SYNC command is unlike other BDC commands because the host does not necessarily know the correct speed to use for serial communications until after it has analyzed the response to the SYNC command.

To issue a SYNC command, the host:

1. Drives the BKGD pin low for at least 128 cycles of the slowest possible BDC clock (bus clock or device-specific alternate clock source).
2. Drives BKGD high for a brief speed-up pulse to get a fast rise time. (This speedup pulse is typically one cycle of the host clock which is as fast as the maximum target BDC clock.)
3. Removes all drive to the BKGD pin so it reverts to high impedance.
4. Listens to the BKGD pin for the sync response pulse.

Upon detecting the sync request from the host (which is a much longer low time than would ever occur during normal BDC communications), the target:

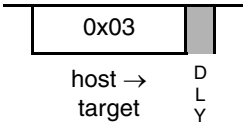
1. Waits for BKGD to return to a logic high.

2. Delays 16 cycles to allow the host to stop driving the high speed-up pulse.
3. Drives BKGD low for 128 BDC clock cycles.
4. Drives a 1-cycle high speed-up pulse to force a fast rise time on BKGD.
5. Removes all drive to the BKGD pin so it reverts to high impedance.

The host measures the low time of this 128-cycle sync response pulse and determines the correct speed for subsequent BDC communications. Typically, the host can determine the correct communication speed within a few percent of the actual target speed and the serial protocol can easily tolerate this speed error.

26.4.1.5.2 ACK_DISABLE

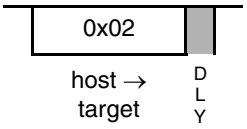
Disable host/target handshake protocol **Always Available**



Disables the serial communication handshake protocol. The subsequent commands, issued after the ACK_DISABLE command, do not execute the hardware handshake protocol. This command is not followed by an ACK pulse.

26.4.1.5.3 ACK_ENABLE

Enable host/target handshake protocol **Always Available**

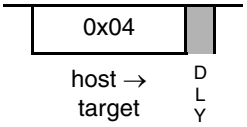


Enables the hardware handshake protocol in the serial communication. The hardware handshake is implemented by an acknowledge (ACK) pulse issued by the target MCU in response to a host command. The ACK_ENABLE command is interpreted and executed in the BDC logic without the need to interface with the CPU. However, an acknowledge (ACK) pulse is issued by the target device after this command is executed. This feature can be used by the host to evaluate if the target supports the hardware handshake protocol. If the target supports the hardware handshake protocol, subsequent commands are enabled to execute the hardware handshake protocol, otherwise this command is ignored by the target.

For additional information about the hardware handshake protocol, refer to Section 26.4.1.6, “Serial Interface Hardware Handshake Protocol,” and Section 26.4.1.7, “Hardware Handshake Abort Procedure.”

26.4.1.5.4 BACKGROUND

Enter active background mode (if enabled) **Non-intrusive**



Provided XCSR[ENBDM] is set (BDM enabled), the BACKGROUND command causes the target MCU to enter active background (halt) mode as soon as the current CPU instruction finishes. If ENBDM is cleared (its default value), the BACKGROUND command is ignored.

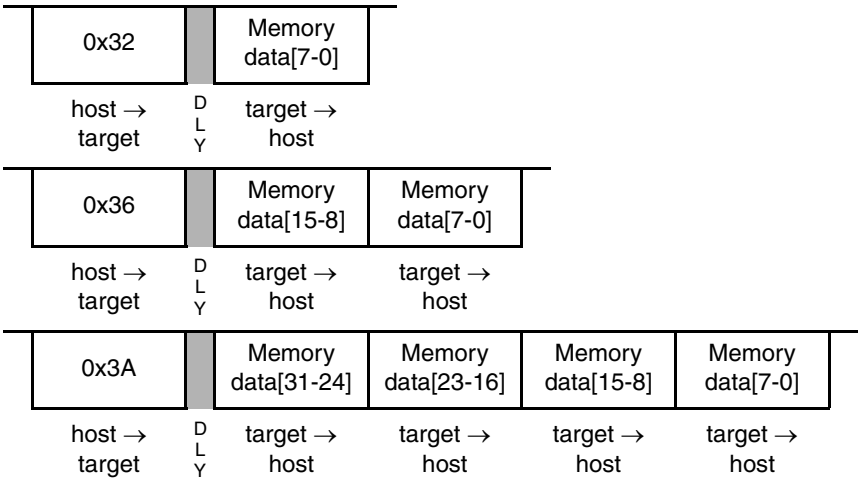
A delay of 32 BDC clock cycles is required after the BACKGROUND command to allow the target MCU to finish its current CPU instruction and enter active background mode before a new BDC command can be accepted.

After the target MCU is reset into a normal operating mode, the host debugger would send a WRITE_XCSR_BYTE command to set ENBDM before attempting to send the BACKGROUND command the first time. Normally, the development host would set ENBDM once at the beginning of a debug session or after a target system reset, and then leave the ENBDM bit set during debugging operations. During debugging, the host would use GO commands to move from active background mode to normal user program execution and would use BACKGROUND commands or breakpoints to return to active background mode.

26.4.1.5.5 DUMP_MEM.sz, DUMP_MEM.sz_WS

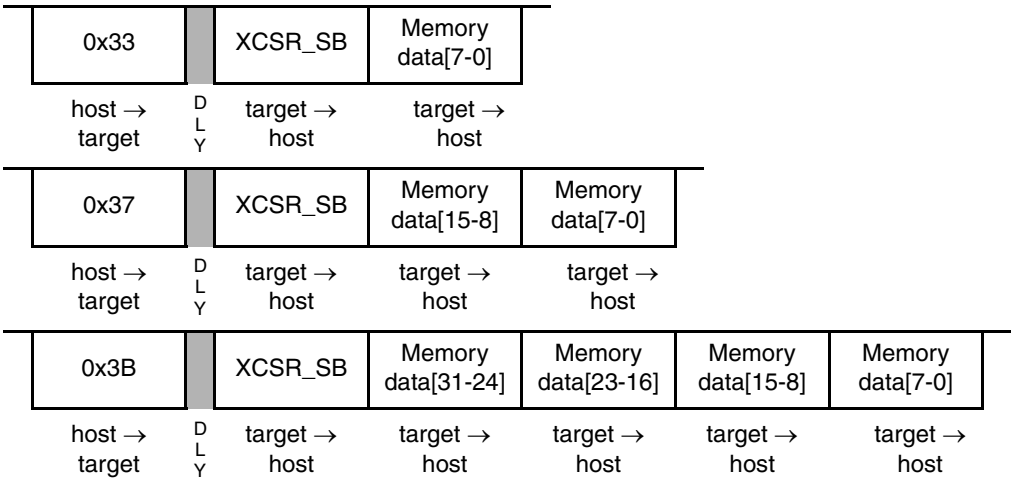
DUMP_MEM.sz

Read memory specified by debug address register, then increment address **Non-intrusive**



DUMP_MEM.sz_WS

Read memory specified by debug address register with status, then increment address **Non-intrusive**



DUMP_MEM{_WS} is used with the READ_MEM{_WS} command to access large blocks of memory. An initial READ_MEM{_WS} is executed to set-up the starting address of the block and to retrieve the first result. If an initial READ_MEM{_WS} is not executed before the first DUMP_MEM{_WS}, an illegal command response is returned. The DUMP_MEM{_WS} command retrieves subsequent operands. The initial address is incremented by the operand size (1, 2, or 4) and saved in a temporary register. Subsequent DUMP_MEM{_WS} commands use this address, perform the memory read, increment it by the current operand size, and store the updated address in the temporary register. If the with-status option is specified, the core status byte (XCSR_SB) contained in XCSR[31–24] is returned before the read data. XCSR_SB reflects the state after the memory read was performed.

NOTE

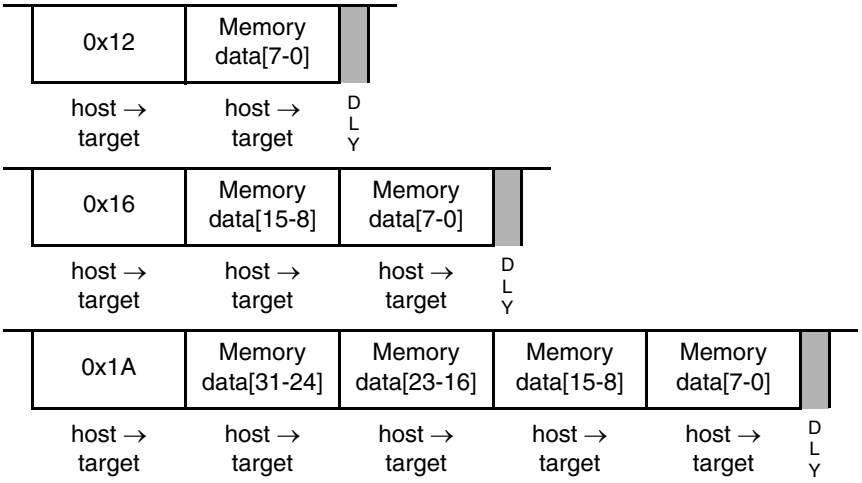
DUMP_MEM{_WS} does not check for a valid address; it is a valid command only when preceded by NOP, READ_MEM{_WS}, or another DUMP_MEM{_WS} command. Otherwise, an illegal command response is returned. NOP can be used for inter-command padding without corrupting the address pointer.

The size field (sz) is examined each time a DUMP_MEM{_WS} command is processed, allowing the operand size to be dynamically altered. The examples show the DUMP_MEM.B{_WS}, DUMP_MEM.W{_WS} and DUMP_MEM.L{_WS} commands.

26.4.1.5.6 FILL_MEM.sz, FILL_MEM.sz_WS

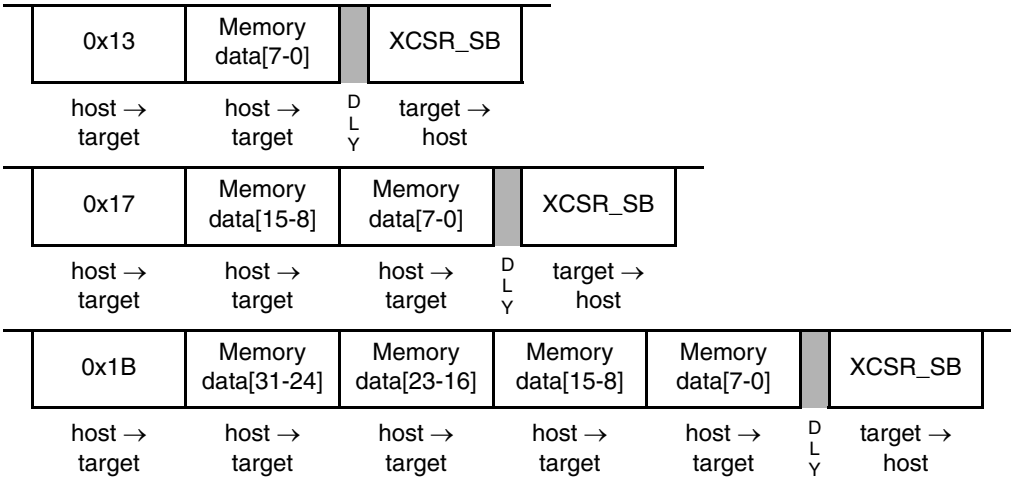
FILL_MEM.sz

Write memory specified by debug address register, then increment address — **Non-intrusive**



FILL_MEM.sz_WS

Write memory specified by debug address register with status, then increment address — **Non-intrusive**



FILL_MEM{_WS} is used with the WRITE_MEM{_WS} command to access large blocks of memory. An initial WRITE_MEM{_WS} is executed to set up the starting address of the block and write the first datum. If an initial WRITE_MEM{_WS} is not executed before the first FILL_MEM{_WS}, an illegal command response is returned. The FILL_MEM{_WS} command stores subsequent operands. The initial address is incremented by the operand size (1, 2, or 4) and saved in a temporary register. Subsequent WRITE_MEM{_WS} commands use this address, perform the memory write, increment it by the current operand size, and store the updated address in the temporary register. If the with-status option is specified,

the core status byte (XCSR_SB) contained in XCSR[31–24] is returned after the write data. XCSR_SB reflects the state after the memory write was performed.

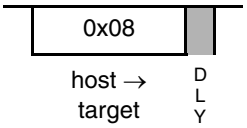
NOTE

FILL_MEM{_WS} does not check for a valid address; it is a valid command only when preceded by NOP, WRITE_MEM{_WS}, or another FILL_MEM{_WS} command. Otherwise, an illegal command response is returned. NOP can be used for intercommand padding without corrupting the address pointer.

The size field (sz) is examined each time a FILL_MEM{_WS} command is processed, allowing the operand size to be dynamically altered. The examples show the FILL_MEM.B{_WS}, FILL_MEM.W{_WS} and FILL_MEM.L{_WS} commands.

26.4.1.5.7 GO

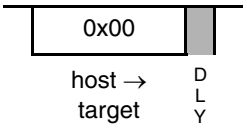
Go **Non-intrusive**



This command is used to exit active background (halt) mode and begin (or resume) execution of the application's instructions. The CPU's pipeline is flushed and refilled before normal instruction execution resumes. Prefetching begins at the current address in the PC and at the current privilege level. If any register (such as the PC or SR) is altered by a BDM command while the processor is halted, the updated value is used when prefetching resumes. If a GO command is issued and the CPU is not halted, the command is ignored.

26.4.1.5.8 NOP

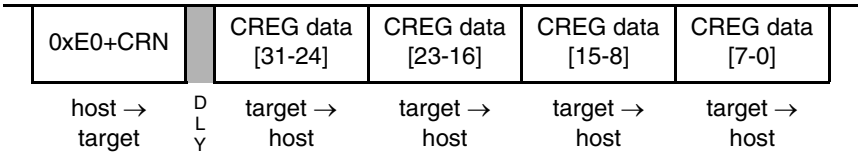
No operation **Non-intrusive**



NOP performs no operation and may be used as a null command where required.

26.4.1.5.9 READ_CREG

Read CPU control register **Active Background**

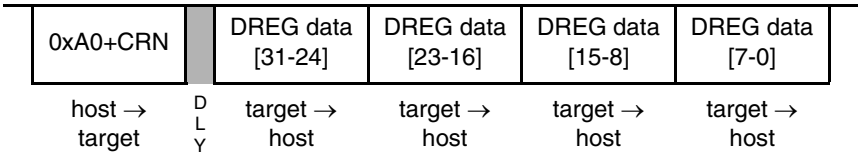


If the processor is halted, this command reads the selected control register and returns the 32-bit result. This register grouping includes the PC, SR, CPUCR, MACSR, MASK, ACC, VBR, and OTHER_A7. Accesses to processor control registers are always 32-bits wide, regardless of implemented register width. The register is addressed through the core register number (CRN). See Table 26-24 for the CRN details when CRG is 11.

If the processor is not halted, this command is rejected as an illegal operation and no operation is performed.

26.4.1.5.10 READ_DREG

Read debug control register **Non-intrusive**

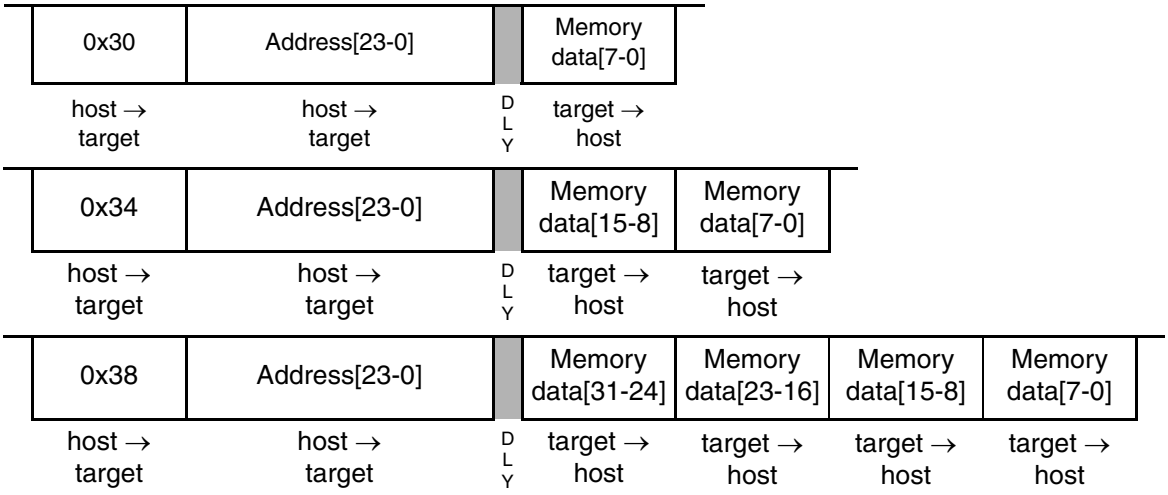


This command reads the selected debug control register and returns the 32-bit result. This register grouping includes the CSR, XCSR, CSR2, and CSR3. Accesses to debug control registers are always 32-bits wide, regardless of implemented register width. The register is addressed through the core register number (CRN). See Table 26-4 for CRN details.

26.4.1.5.11 READ_MEM.sz, READ_MEM.sz_WS

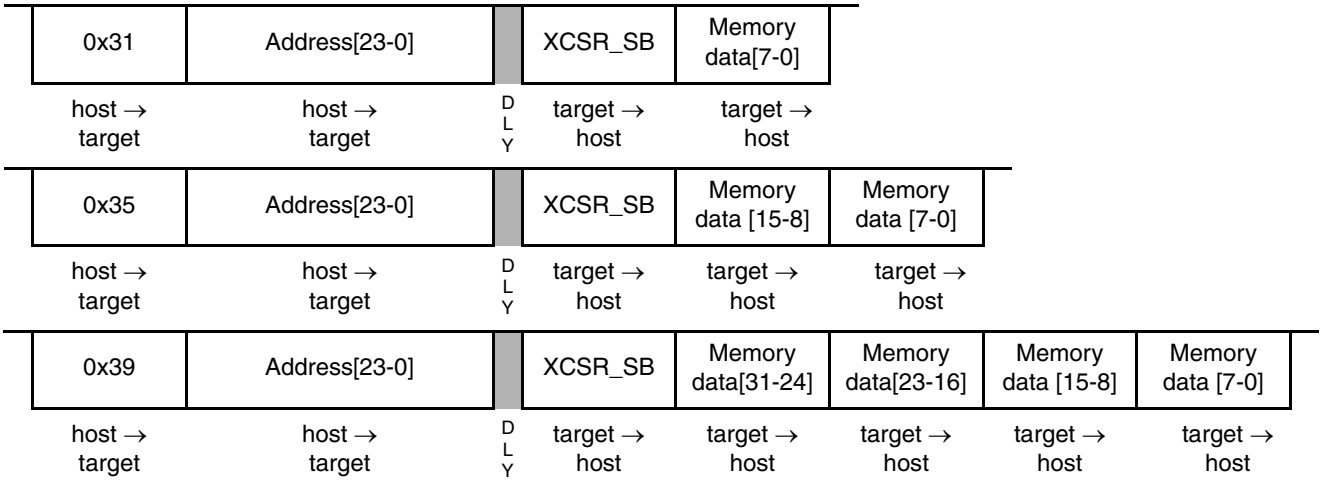
READ_MEM.sz

Read memory at the specified address **Non-intrusive**



READ_MEM.sz_WS

Read memory at the specified address with status **Non-intrusive**

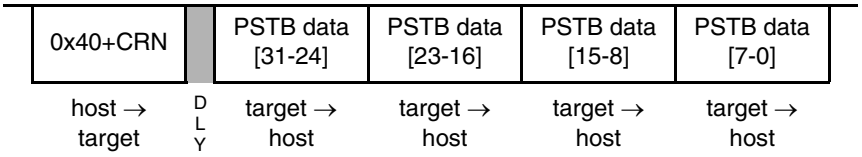


Read data at the specified memory address. The reference address is transmitted as three 8-bit packets (msb to lsb) immediately after the command packet. The access attributes are defined by BAAR[TT,TM]. The hardware forces low-order address bits to zeros for word and longword accesses to ensure these accesses are on 0-modulo-size alignments. If the with-status option is specified, the core status byte (XCSR_SB) contained in XCSR[31–24] is returned before the read data. XCSR_SB reflects the state after the memory read was performed.

The examples show the READ_MEM.B{_WS}, READ_MEM.W{_WS} and READ_MEM.L{_WS} commands.

26.4.1.5.12 READ_PSTB

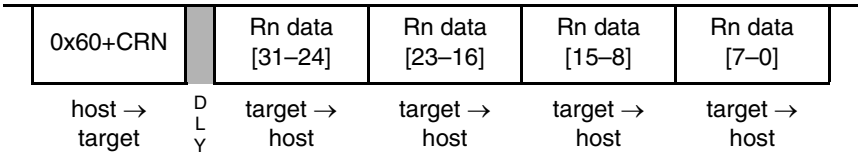
Read PST trace buffer at the specified address **Non-intrusive**



Read 32 bits of captured PST/DDATA values from the trace buffer at the specified address. The PST trace buffer contains 64 six-bit entries, packed consecutively into 12 longword locations. See Figure 26-15 for an illustration of how the buffer entries are packed.

26.4.1.5.13 READ_Rn

Read general-purpose CPU register **Active Background**

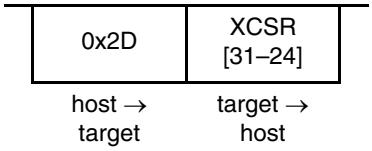


If the processor is halted, this command reads the selected CPU general-purpose register (An, Dn) and returns the 32-bit result. See Table 26-24 for the CRN details when CRG is 01.

If the processor is not halted, this command is rejected as an illegal operation and no operation is performed.

26.4.1.5.14 READ_XCSR_BYTE

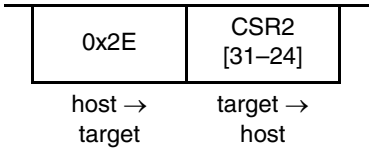
Read XCSR Status Byte **Always Available**



Read the special status byte of XCSR (XCSR[31–24]). This command can be executed in any mode.

26.4.1.5.15 READ_CSR2_BYTE

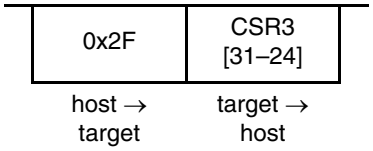
Read CSR2 Status Byte **Always Available**



Read the most significant byte of CSR2 (CSR2[31–24]). This command can be executed in any mode.

26.4.1.5.16 READ_CSR3_BYTE

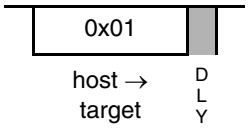
Read CSR3 Status Byte **Always Available**



Read the most significant byte of the CSR3 (CSR3[31–24]). This command can be executed in any mode.

26.4.1.5.17 SYNC_PC

Synchronize PC to PST/DDATA Signals **Non-intrusive**



Capture the processor's current PC (program counter) and display it on the PST/DDATA signals. After the debug module receives the command, it sends a signal to the ColdFire core that the current PC must be displayed. The core responds by forcing an instruction fetch to the next PC with the address being captured by the DDATA logic. The DDATA logic captures a 2- or 3-byte instruction address, based on CSR[9]. If CSR[9] is cleared, then a 2-byte address is captured, else a 3-byte address is captured. The specific sequence of PST and DDATA values is defined as:

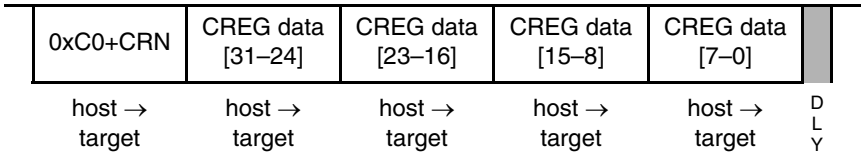
1. Debug signals a SYNC_PC command is pending.
2. CPU completes the current instruction.
3. CPU forces an instruction fetch to the next PC, generating a PST = 0x5 value indicating a taken branch. DDATA captures the instruction address corresponding to the PC. DDATA generates a PST marker signalling a 2- or 3-byte address as defined by CSR[9] (CSR[9] = 0, 2-byte; CSR[9] = 1, 3-byte) and displays the captured PC address.

This command can be used to provide a PC synchronization point between the core's execution and the application code in the PST trace buffer. It can also be used to dynamically access the PC for performance

monitoring as the execution of this command is considerably less obtrusive to the real-time operation of an application than a BACKGROUND/read-PC/GO command sequence.

26.4.1.5.18 WRITE_CREG

Write CPU control register **Active Background**

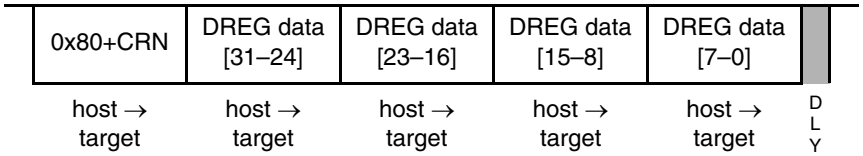


If the processor is halted, this command writes the 32-bit operand to the selected control register. This register grouping includes the PC, SR, CPUCR, MACSR, MASK, ACC, VBR, and OTHER_A7. Accesses to processor control registers are always 32-bits wide, regardless of implemented register width. The register is addressed through the core register number (CRN). See Table 26-24 for the CRN details when CRG is 11.

If the processor is not halted, this command is rejected as an illegal operation and no operation is performed.

26.4.1.5.19 WRITE_DREG

Write debug control register **Non-intrusive**



This command writes the 32-bit operand to the selected debug control register. This grouping includes all the debug control registers ({X}CSR*n*, BAAR, AATR, TDR, PBR*n*, PBMR, AB*x*R, DBR, DBMR). Accesses to debug control registers are always 32-bits wide, regardless of implemented register width. The register is addressed through the core register number (CRN). See Table 26-4 for CRN details.

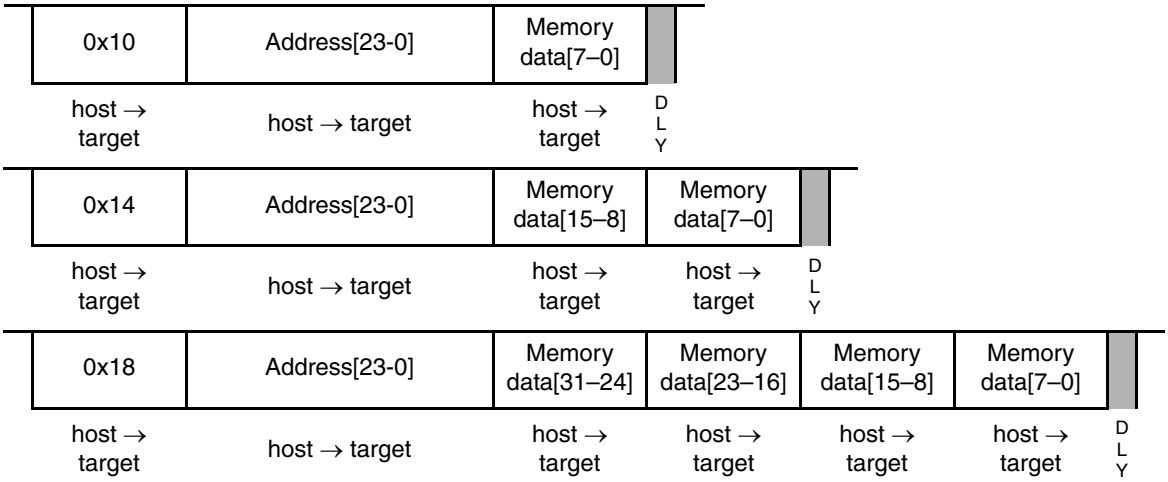
NOTE

When writing XCSR, CSR2, or CSR3, WRITE_DREG only writes bits 23–0. The upper byte of these debug registers is only written with the special WRITE_XCSR_BYTE, WRITE_CSR2_BYTE, and WRITE_CSR3_BYTE commands.

26.4.1.5.20 WRITE_MEM.sz, WRITE_MEM.sz_WS

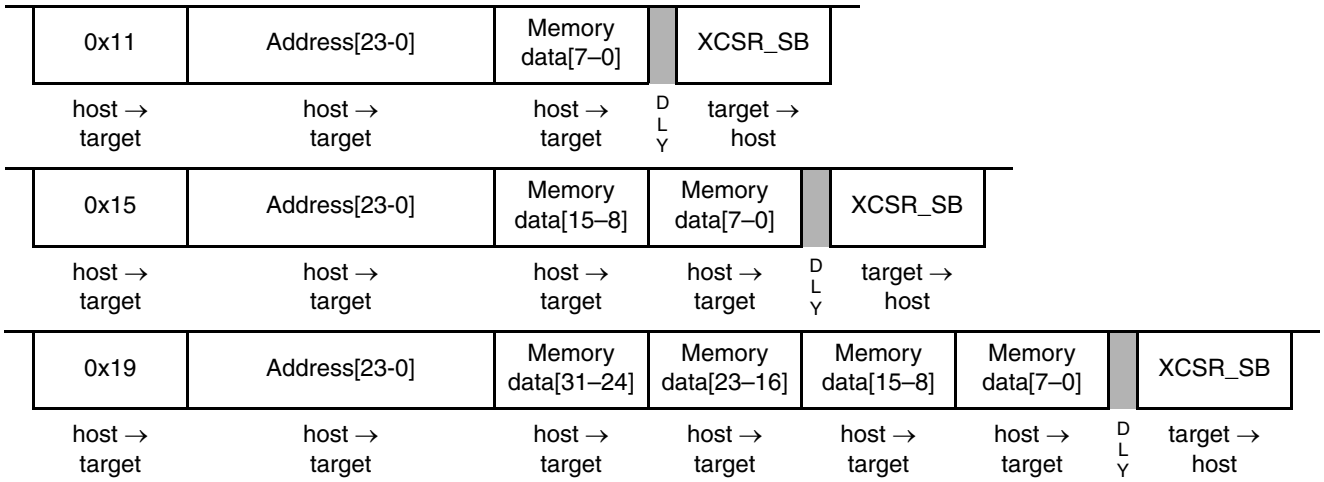
WRITE_MEM.sz

Write memory at the specified address **Non-intrusive**



WRITE_MEM.sz_WS

Write memory at the specified address with status **Non-intrusive**

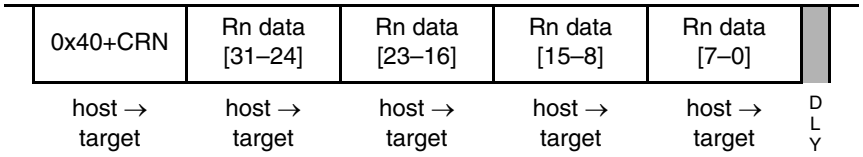


Write data at the specified memory address. The reference address is transmitted as three 8-bit packets (msb to lsb) immediately after the command packet. The access attributes are defined by BAAR[TT,TM]. The hardware forces low-order address bits to zeros for word and longword accesses to ensure these accesses are on 0-modulo-size alignments. If the with-status option is specified, the core status byte (XCSR_SB) contained in XCSR[31–24] is returned after the read data. XCSR_SB reflects the state after the memory write was performed.

The examples show the WRITE_MEM.B{_WS}, WRITE_MEM.W{_WS}, and WRITE_MEM.L{_WS} commands.

26.4.1.5.21 WRITE_Rn

Write general-purpose CPU register **Active Background**

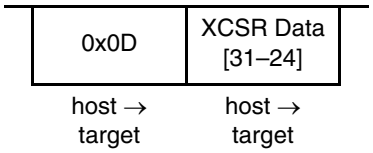


If the processor is halted, this command writes the 32-bit operand to the selected CPU general-purpose register (An, Dn). See Table 26-24 for the CRN details when CRG is 01.

If the processor is not halted, this command is rejected as an illegal operation and no operation is performed.

26.4.1.5.22 WRITE_XCSR_BYTE

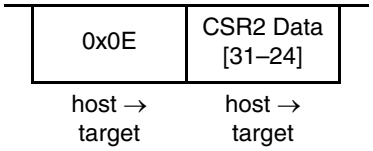
Write XCSR Status Byte **Always Available**



Write the special status byte of XCSR (XCSR[31–24]). This command can be executed in any mode.

26.4.1.5.23 WRITE_CSR2_BYTE

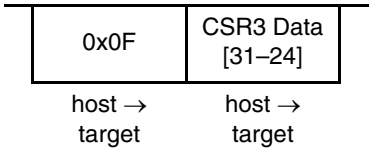
Write CSR2 Status Byte **Always Available**



Write the most significant byte of CSR2 (CSR2[31–24]). This command can be executed in any mode.

26.4.1.5.24 WRITE_CSR3_BYTE

Write CSR3 Status Byte **Always Available**



Write the most significant byte of CSR3 (CSR3[31–24]). This command can be executed in any mode.

26.4.1.5.25 BDM Accesses of the MAC Registers

The presence of rounding logic in the output datapath of the MAC requires special care for BDM-initiated reads and writes of its programming model. In particular, any result rounding modes must be disabled during the read/write process so the exact bit-wise MAC register contents are accessed.

For example, a BDM read of the accumulator (ACC) must be preceded by two commands accessing the MAC status register, as shown in the following sequence:

```
BdmReadACC (
        rcreg    macsr;            // read current macsr contents and save
        wcreg    #0,macsr;         // disable all rounding modes
        rcreg    ACC;              // read the desired accumulator
        wcreg    #saved_data,macsr;// restore the original macsr
)
```

Likewise, to write an accumulator register, the following BDM sequence is needed:

```
BdmWriteACC (
        rcreg    macsr;            // read current macsr contents and save
        wcreg    #0,macsr;         // disable all rounding modes
        wcreg    #data,ACC;        // write the desired accumulator
        wcreg    #saved_data,macsr;// restore the original macsr
)
```

For more information on saving and restoring the complete MAC programming model, see Section 9.3.1.2, “Saving and Restoring the MAC Programming Model.”

26.4.1.6 Serial Interface Hardware Handshake Protocol

BDC commands that require CPU execution are ultimately treated at the core clock rate. Because the BDC clock source can be asynchronous relative to the bus frequency when CLKSW is cleared, it is necessary to provide a handshake protocol so the host can determine when an issued command is executed by the CPU. This section describes this protocol.

The hardware handshake protocol signals to the host controller when an issued command was successfully executed by the target. This protocol is implemented by a low pulse (16 BDC clock cycles) followed by a brief speedup pulse on the BKGD pin, generated by the target MCU when a command, issued by the host, has been successfully executed. See Figure 26-20. This pulse is referred to as the ACK pulse. After the ACK pulse is finished, the host can start the data-read portion of the command if the last-issued command was a read command, or start a new command if the last command was a write command or a control command (BACKGROUND, GO, NOP, SYNC_PC). The ACK pulse is not issued earlier than 32 BDC clock cycles after the BDC command was issued. The end of the BDC command is assumed to be the 16th BDC clock cycle of the last bit. This minimum delay assures enough time for the host to recognize the ACK pulse. There is no upper limit for the delay between the command and the related ACK pulse, because the command execution depends on the CPU bus frequency, which in some cases could be slow compared to the serial communication rate. This protocol allows great flexibility for pod designers, because it does not rely on any accurate time measurement or short response time to any event in the serial communication.

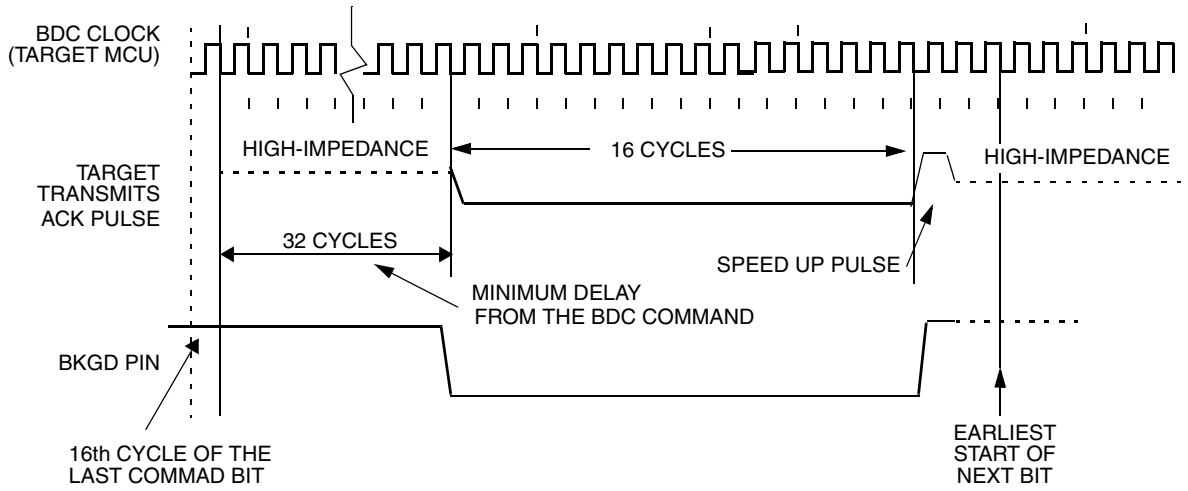


Figure 26-20. Target Acknowledge Pulse (ACK)

NOTE

If the ACK pulse was issued by the target, the host assumes the previous command was executed. If the CPU enters a stop mode prior to executing a non-intrusive command, the command is discarded and the ACK pulse is not issued. After entering a stop mode, the BDC command is no longer pending and the XCSR[CSTAT] value of 001 is kept until the next command is successfully executed.

Figure 26-21 shows the ACK handshake protocol in a command level timing diagram. A READ_MEM.B command is used as an example:

1. The 8-bit command code is sent by the host, followed by the address of the memory location to be read.
2. The target BDC decodes the command and sends it to the CPU.
3. Upon receiving the BDC command request, the CPU schedules a execution slot for the command.
4. The CPU temporarily stalls the instruction stream at the scheduled point, executes the READ_MEM.B command and then continues.

This process is referred to as cycle stealing. The READ_MEM.B appears as a single-cycle operation to the processor, even though the pipelined nature of the Operand Execution Pipeline requires multiple CPU clock cycles for it to actually complete. After that, the debug module tracks the execution of the READ_MEM.b command as the processor resumes the normal flow of the application program. After detecting the READ_MEM.B command is done, the BDC issues an ACK pulse to the host controller, indicating that the addressed byte is ready to be retrieved. After detecting the ACK pulse, the host initiates the data-read portion of the command.

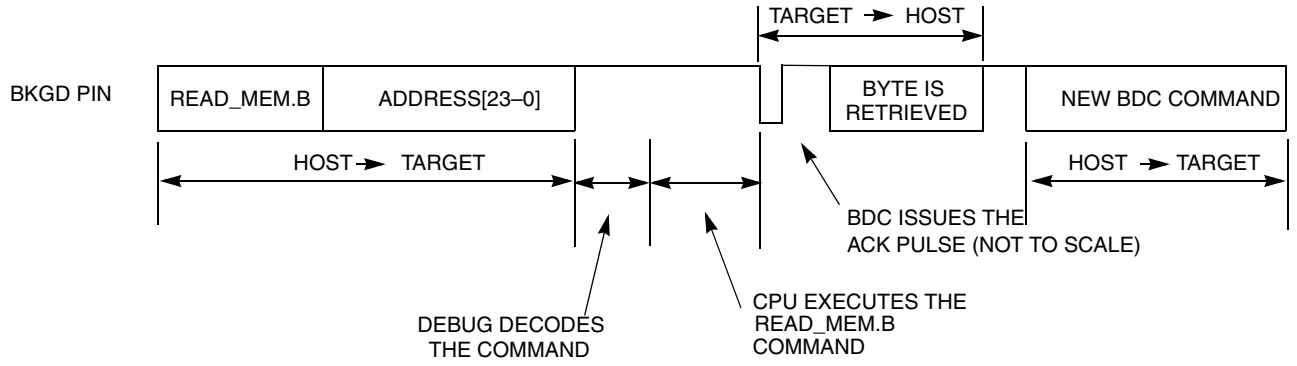


Figure 26-21. Handshake Protocol at Command Level

Unlike a normal bit transfer, where the host initiates the transmission by issuing a negative edge in the BKGD pin, the serial interface ACK handshake pulse is initiated by the target MCU. The hardware handshake protocol in Figure 26-21 specifies the timing when the BKGD pin is being driven, so the host should follow these timing relationships to avoid the risks of an electrical conflict at the BKGD pin.

The ACK handshake protocol does not support nested ACK pulses. If a BDC command is not acknowledged by an ACK pulse, the host first needs to abort the pending command before issuing a new BDC command. When the CPU enters a stop mode at about the same time the host issues a command that requires CPU execution, the target discards the incoming command. Therefore, the command is not acknowledged by the target, meaning that the ACK pulse is not issued in this case. After a certain time, the host could decide to abort the ACK protocol to allow a new command. Therefore, the protocol provides a mechanism where a command (a pending ACK) could be aborted. Unlike a regular BDC command, the ACK pulse does not provide a timeout. In the case of a STOP instruction where the ACK is prevented from being issued, it would remain pending indefinitely if not aborted. See the handshake abort procedure described in Section 26.4.1.7, “Hardware Handshake Abort Procedure.”

26.4.1.7 Hardware Handshake Abort Procedure

The abort procedure is based on the SYNC command. To abort a command that has not responded with an ACK pulse, the host controller generates a sync request (by driving BKGD low for at least 128 serial clock cycles and then driving it high for one serial clock cycle as a speedup pulse). By detecting this long low pulse on the BKGD pin, the target executes the sync protocol (see Section 26.4.1.5.1, “SYNC”), and assumes that the pending command and therefore the related ACK pulse, are being aborted. Therefore, after the sync protocol completes, the host is free to issue new BDC commands.

Because the host knows the target BDC clock frequency, the SYNC command does not need to consider the lowest possible target frequency. In this case, the host could issue a SYNC close to the 128 serial clock cycles length, providing a small overhead on the pulse length to assure the sync pulse is not misinterpreted by the target.

It is important to notice that any issued BDC command that requires CPU execution is scheduled for execution by the pipeline based on the dynamic state of the machine, provided the processor does not enter any of the stop modes. If the host aborts a command by sending the sync pulse, it should then read XCSR[CSTAT] after the sync response is issued by the target, checking for CSTAT cleared, before

attempting to send any new command that requires CPU execution. This prevents the new command from being discarded at the debug/CPU interface, due to the pending command being executed by the CPU. Any new command should be issued only after XCSR[CSTAT] is cleared.

There are multiple reasons that could cause a command to take too long to execute, measured in terms of the serial communication rate: The BDC clock frequency could be much faster than the CPU clock frequency, or the CPU could be accessing a slow memory, which would cause pipeline stall cycles to occur. All commands referencing the CPU registers or memory require access to the processor's local bus to complete. If the processor is executing a tight loop contained within a single aligned longword, the processor may never successfully grant the internal bus to the debug command. For example:

```
        align   4
label1: nop
        bra.b   label1
or

        align   4
label2: bra.w   label2
```

These two examples of tight loops exhibit the BDM lockout behavior. If the loop spans across two longwords, there are no issues, so the recommended construct is:

```
        align   4
label3: bra.l   label3
```

The hardware handshake protocol is appropriate for these situations, but the host could also decide to use the software handshake protocol instead. In this case, if XCSR[CSTAT] is 001, there is a BDC command pending at the debug/CPU interface. The host controller should monitor XCSR[CSTAT] and wait until it is 000 to be able to issue a new command that requires CPU execution. However, if the XCSR[CSTAT] is 1*xx*, the host should assume the last command failed to execute. To recover from this condition, the following sequence is suggested:

1. Issue a SYNC command to reset the BDC communication channel.
2. The host issues a BDM NOP command.
3. The host reads the channel status using a READ_XCSR_BYTE command.
4. If XCSR[CSTAT] is 000
   then the status is okay; proceed
   else
      Halt the CPU using a BDM BACKGROUND command
      Repeat steps 1,2,3
      If XCSR[CSTAT] is 000, then proceed, else reset the device

Figure 26-22 shows a SYNC command aborting a READ_MEM.B. After the command is aborted, a new command could be issued by the host.

NOTE

Figure 26-22 signal timing is not drawn to scale.

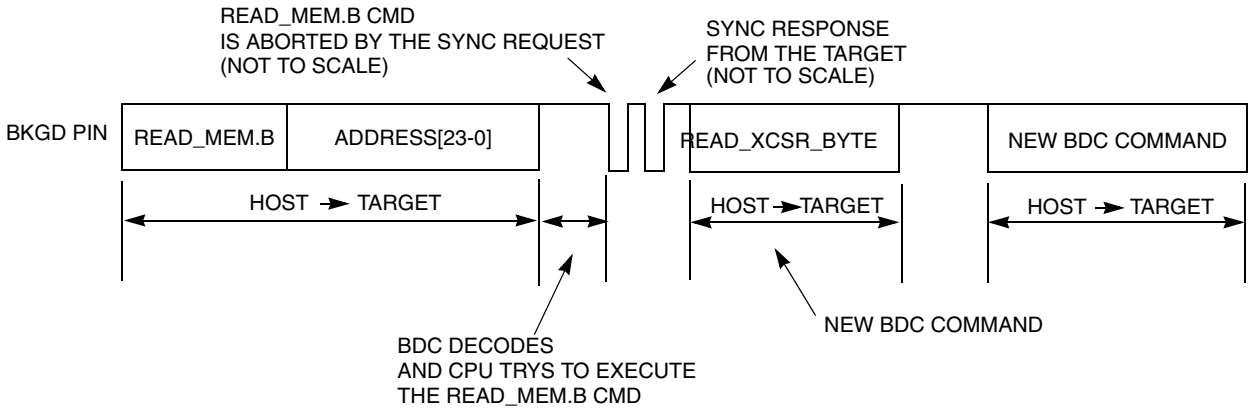


Figure 26-22. ACK Abort Procedure at the Command Level

Figure 26-23 a shows a conflict between the ACK pulse and the sync request pulse. This conflict could occur if a pod device is connected to the target BKGD pin and the target is already executing a BDC command. Consider that the target CPU is executing a pending BDC command at the exact moment the pod is being connected to the BKGD pin. In this case, an ACK pulse is issued at the same time as the SYNC command. In this case there is an electrical conflict between the ACK speedup pulse and the sync pulse. Because this is not a probable situation, the protocol does not prevent this conflict from happening.

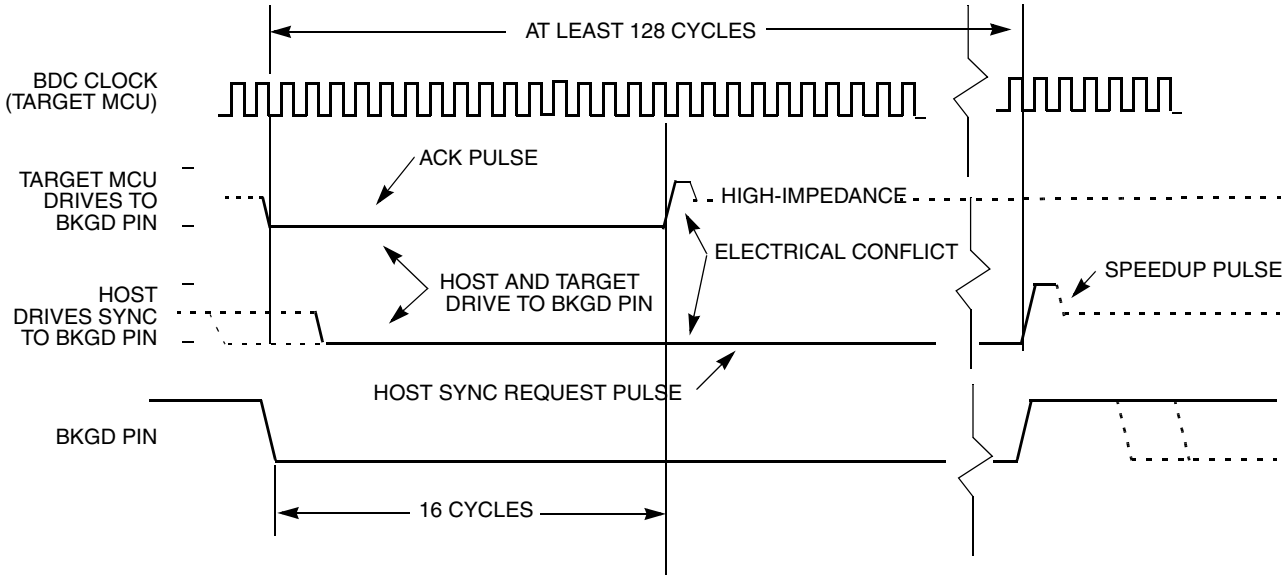


Figure 26-23. ACK Pulse and SYNC Request Conflict

The hardware handshake protocol is enabled by the ACK_ENABLE command and disabled by the ACK_DISABLE command. It also allows for pod devices to choose between the hardware handshake protocol or the software protocol that monitors the XCSR status byte. The ACK_ENABLE and ACK_DISABLE commands are:

- ACK_ENABLE — Enables the hardware handshake protocol. The target issues the ACK pulse when a CPU command is executed. The ACK_ENABLE command itself also has the ACK pulse as a response.

- ACK_DISABLE — Disables the ACK pulse protocol. In this case, the host should verify the state of XCSR[CSTAT] to evaluate if there are pending commands and to check if the CPU's operating state has changed to or from active background mode via XCSR[31–30].

The default state of the protocol, after reset, is hardware handshake protocol disabled.

The commands that do not require CPU execution, or that have the status register included in the retrieved bit stream, do not perform the hardware handshake protocol. Therefore, the target does not respond with an ACK pulse for those commands even if the hardware protocol is enabled. Conversely, only commands that require CPU execution and do not include the status byte perform the hardware handshake protocol. See the third column in Table 26-25 for the complete enumeration of this function.

An exception is the ACK_ENABLE command, which does not require CPU execution but responds with the ACK pulse. This feature can be used by the host to evaluate if the target supports the hardware handshake protocol. If an ACK pulse is issued in response to this command, the host knows that the target supports the hardware handshake protocol. If the target does not support the hardware handshake protocol the ACK pulse is not issued. In this case, the ACK_ENABLE command is ignored by the target, because it is not recognized as a valid command.

26.4.2 Real-Time Debug Support

The ColdFire family supports debugging real-time applications. For these types of embedded systems, the processor must continue to operate during debug. The foundation of this area of debug support is that while the processor cannot be halted to allow debugging, the system can generally tolerate the small intrusions with minimal effect on real-time operation.

NOTE

The details regarding real-time debug support will be supplied at a later time.

26.4.3 Trace Support

For the baseline V1 ColdFire core and its single debug signal, support for trace functionality is completely redefined. The V1 solution provides an on-chip PST/DDATA trace buffer (known as the PSTB) to record the stream of PST and DDATA values.

As a review, the classic ColdFire debug architecture supports real-time trace via the PST/DDATA output signals. For this functionality, the following apply:

- One (or more) PST value is generated for each executed instruction
- Branch target instruction address information is displayed on all non-PC-relative change-of-flow instructions, where the user selects a programmable number of bytes of target address
  - — Displayed information includes PST marker plus target instruction address as DDATA
  - — Captured address creates the appropriate number of DDATA entries, each with 4 bits of address
- Optional data trace capabilities are provided for accesses mapped to the slave peripheral bus
  - — Displayed information includes PST marker plus captured operand value as DDATA
  - — Captured operand creates the appropriate number of DDATA entries, each with 4 bits of data

The resulting PST/DDATA output stream, with the application program memory image, provides an instruction-by-instruction dynamic trace of the execution path.

Even with the application of a PST trace buffer, problems associated with the PST bandwidth and associated fill rate of the buffer remain. Given that there is one (or more) PST entry per instruction, the PSTB would fill rapidly without some type of data compression.

Consider the following example to illustrate the PST compression algorithm. Most sequential instructions generate a single PST = 1 value. Without compression, the execution of ten sequential instructions generates a stream of ten PST = 1 values. With PST compression, the reporting of any PST = 1 value is delayed so that consecutive PST = 1 values can be accumulated. When a PST ≠ 1 value is reported, the maximum accumulation count is reached, or a debug data value is captured, a single accumulated PST value is generated. Returning to the example with compression enabled, the execution of ten sequential instructions generates a single PST value indicating ten sequential instructions have been executed.

This technique has proven to be effective at significantly reducing the average PST entries per instruction and PST entries per machine cycle. The application of this compression technique makes the application of a useful PST trace buffer for the V1 ColdFire core realizable. The resulting 5-bit PST definitions are shown in Table 26-26.

Table 26-26. CF1 Debug Processor Status Encodings

PST[4:0]	Definition
0x00	Continue execution. Many instructions execute in one processor cycle. If an instruction requires more processor clock cycles, subsequent clock cycles are indicated by driving PST with this encoding.
0x01	Begin execution of one instruction. For most instructions, this encoding signals the first processor clock cycle of an instruction's execution. Certain change-of-flow opcodes, plus the PULSE and WDDATA instructions, generate different encodings.
0x02	Reserved
0x03	Entry into user-mode. Signaled after execution of the instruction that caused the ColdFire processor to enter user mode.
0x04	Begin execution of PULSE and WDDATA instructions. PULSE defines triggers or markers for debug and/or performance analysis. WDDATA lets the core write any operand (byte, word, or longword) directly to the DDATA port, independent of debug module configuration. When WDDATA is executed, a value of 0x04 is signaled on the PST port, followed by the appropriate marker, and then the data transfer on the DDATA port. The number of captured data bytes depends on the WDDATA operand size.
0x05	Begin execution of taken branch or SYNC_PC BDM command. For some opcodes, a branch target address may be displayed on DDATA depending on the CSR settings. CSR also controls the number of address bytes displayed, indicated by the PST marker value preceding the DDATA nibble that begins the data output. This encoding also indicates that the SYNC_PC command has been processed.
0x06	Reserved
0x07	Begin execution of return from exception (RTE) instruction.
0x08–0x0B	Indicates the number of data bytes to be loaded into the PST trace buffer. The capturing of peripheral bus data references is controlled by CSR[DDC]. 0x08 Begin 1-byte data transfer on DDATA 0x09 Begin 2-byte data transfer on DDATA 0x0A Reserved 0x0B Begin 4-byte data transfer on DDATA

Table 26-26. CF1 Debug Processor Status Encodings (continued)

PST[4:0]	Definition
0x0C–0x0F	Indicates the number of address bytes to be loaded into the PST trace buffer. The capturing of branch target addresses is controlled by CSR[BTB]. 0x0C Reserved 0x0D Begin 2-byte address transfer on DDATA (Displayed address is shifted right 1: ADDR[16:1]) 0x0E Begin 3-byte address transfer on DDATA (Displayed address is shifted right 1: ADDR[23:1]) 0x0F Reserved
0x10–0x11	Reserved
0x12	Completed execution of 2 sequential instructions
0x13	Completed execution of 3 sequential instructions
0x14	Completed execution of 4 sequential instructions
0x15	Completed execution of 5 sequential instructions
0x16	Completed execution of 6 sequential instructions
0x17	Completed execution of 7 sequential instructions
0x18	Completed execution of 8 sequential instructions
0x19	Completed execution of 9 sequential instructions
0x1A	Completed execution of 10 sequential instructions
0x1B	This value signals there has been a change in the breakpoint trigger state machine. It appears as a single marker for each state change and is immediately followed by a DDATA value signaling the new breakpoint trigger state encoding. The DDATA breakpoint trigger state value is defined as (0x20 + 2 × CSR[BSTAT]): 0x20 No breakpoints enabled 0x22 Waiting for a level-1 breakpoint 0x24 Level-1 breakpoint triggered 0x2A Waiting for a level-2 breakpoint 0x2C Level-2 breakpoint triggered
0x1C	Exception processing. This value signals the processor has encountered an exception condition. Although this is a multi-cycle mode, there are only two PST = 0x1C values recorded before the mode value is suppressed.
0x1D	Emulator mode exception processing. This value signals the processor has encountered a debug interrupt or a properly-configured trace exception. Although this is a multi-cycle mode, there are only two PST = 0x1D values recorded before the mode value is suppressed.
0x1E	Processor is stopped. This value signals the processor has executed a STOP instruction. Although this is a multi-cycle mode because the ColdFire processor remains stopped until an interrupt or reset occurs, there are only two PST = 0x1E values recorded before the mode value is suppressed.
0x1F	Processor is halted. This value signals the processor has been halted. Although this is a multi-cycle mode because the ColdFire processor remains halted until a BDM go command is received or reset occurs, there are only two PST = 0x1F values recorded before the mode value is suppressed.

26.4.3.1 Begin Execution of Taken Branch (PST = 0x05)

The PST is 0x05 when a taken branch is executed. For some opcodes, a branch target address may be loaded into the trace buffer (PSTB) depending on the CSR settings. CSR also controls the number of address bytes loaded that is indicated by the PST marker value immediately preceding the DDATA entry in the PSTB that begins the address entries.

Multiple byte DDATA values are displayed in least-to-most-significant order. The processor captures only those target addresses associated with taken branches that use a variant addressing mode (RTE and RTS instructions, JMP and JSR instructions using address register indirect or indexed addressing modes, and all exception vectors).

The simplest example of a branch instruction using a variant address is the compiled code for a C language case statement. Typically, the evaluation of this statement uses the variable of an expression as an index into a table of offsets, where each offset points to a unique case within the structure. For such change-of-flow operations, the ColdFire processor loads the PSTB as follows:

1. Load PST=0x05 to identify that a taken branch is executed.
2. Optionally load the marker for the target address capture. Encodings 0x0D or 0x0E identify the number of bytes loaded into the PSTB.
3. The new target address is optionally available in the PSTB. The number of bytes of the target address loaded is configurable (2 or 3 bytes, where the encoding is 0x0D and 0x0E, respectively).

Another example of a variant branch instruction is a JMP (A0) instruction. Figure 26-24 shows the PSTB entries that indicate a JMP (A0) execution, assuming CSR[BTB] was programmed to display the lower two bytes of an address.

PST/DDATA Values	Description
0x05	Taken Branch
0x0D	2-byte Address Marker
{10, Address[4:1]}	Address >> 1
{10, Address[8:5]}	
{10, Address[12:9]}	
{10, Address[16:13]}	

Figure 26-24. Example JMP Instruction Output in PSTB

The PST of 0x05 indicates a taken branch and the marker value 0x0D indicates a 2-byte address. Therefore, the following entries display the lower two bytes of address register A0, right-shifted by 1, in least-to-most-significant nibble order. The next PST entry after the JMP instruction completes depends on the target instruction. See Section 26.4.3.2, “PST Trace Buffer (PSTB) Entry Format,” for entry descriptions explaining the 2-bit prefix before each address nibble.

26.4.3.2 PST Trace Buffer (PSTB) Entry Format

As PST and DDATA values are captured and loaded in the trace buffer, each entry is six bits in size therefore, the type of the entry can easily be determined when post-processing the PSTB. See Figure 26-25.

	5	4	3	2	1	0
PSTB[PST]	0	PST[4:0]				
Data PSTB[DDATA]	1	$\overline{R}$/W	Data[3:0]			
Address PSTB[DDATA]	1	0	Address[3:0][1]			
Reset:	—	—	—	—	—	—

[1] Depending on which nibble is displayed (as determined by CSR[9:8]), Address[3:0] sequentially (least-to-most-significant nibble order) displays four bits of the real CPU address [16:1] or [24:1].

Figure 26-25. V1 PST/DDATA Trace Buffer Entry Format

26.4.3.3 PST/DDATA Example

In this section, an example showing the behavior of the PST/DDATA functionality is detailed. Consider the following interrupt service routine that counts the interrupt, negates the IRQ, performs a software IACK, and then exits. This example is presented here because it exercises a considerable set of the PST/DDATA capabilities.

```
                      _isr:
01074: 46fc 2700        mov.w   &0x2700,%sr        # disable interrupts
01078: 2f08             mov.l   %a0,-(%sp)         # save a0
0107a: 2f00             mov.l   %d0,-(%sp)         # save d0
0107c: 302f 0008        mov.w   (8,%sp),%d0        # load format/vector word
01080: e488             lsr.l   &2,%d0             # align vector number
01082: 0280 0000 00ff   andi.l  &0xff,%d0          # isolate vector number
01088: 207c 0080 1400   mov.l   &int_count,%a0     # base of interrupt counters

                     _isr_entry1:
0108e: 52b0 0c00        addq.l  &1,(0,%a0,%d0.l*4) # count the interrupt
01092: 11c0 a021        mov.b   %d0,IGCR0+1.w      # negate the irq
01096: 1038 a020        mov.b   IGCR0.w,%d0        # force the write to complete
0109a: 4e71             nop                        # synchronize the pipelines
0109c: 71b8 ffe0        mvz.b   SWIACK.w,%d0       # software iack: pending irq?
010a0: 0c80 0000 0041   cmpi.l  %d0,&0x41          # level 7 or none pending?
010a6: 6f08             ble.b   _isr_exit          # yes, then exit
010a8: 52b9 0080 145c   addq.l  &1,swiack_count    # increment the swiack count
010ae: 60de             bra.b   _isr_entry1        # continue at entry1

                     _isr_exit:
010b0: 201f             mov.l   (%sp)+,%d0         # restore d0
010b2: 205f             mov.l   (%sp)+,%a0         # restore a0
010b4: 4e73             rte                        # exit
```

This ISR executes mostly as straight-line code: there is a single conditional branch @ PC = 0x10A6, which is taken in this example. The following description includes the PST and DDATA values generated as this

code snippet executes. In this example, the CSR setting enables only the display of 2-byte branch addresses. Operand data captures are not enabled. The sequence begins with an interrupt exception:

```
interrupt exception occurs @ pc = 5432 while in user mode
                                                # pst   = 1c, 1c, 05, 0d
                                                # ddata = 2a, 23, 28, 20
                                                #         trg_addr = 083a << 1
                                                #         trg_addr = 1074
                     _isr:
01074: 46fc 2700        mov.w   &0x2700,%sr     # pst   = 01
01078: 2f08             mov.l   %a0,-(%sp)      # pst   = 01
0107a: 2f00             mov.l   %d0,-(%sp)      # pst   = 01
0107c: 302f 0008        mov.w   (8,%sp),%d0     # pst   = 01
01080: e488             lsr.l   &2,%d0          # pst   = 01
01082: 0280 0000 00ff   andi.l  &0xff,%d0       # pst   = 01
01088: 207c 0080 1400   mov.l   &int_count,%a0  # pst   = 01
0108e: 52b0 0c00        addq.l  &1,(0,%a0,%d0.l*4) # pst   = 01
01092: 11c0 a021        mov.b   %d0,IGCR0+1.w   # pst   = 01, 08
                                                # ddata = 30, 30
                                                #         wdata.b = 0x00
01096: 1038 a020        mov.b   IGCR0.w,%d0     # pst   = 01, 08
                                                # ddata = 28, 21
                                                #         rdata.b = 0x18
0109a: 4e71             nop                     # pst   = 01
0109c: 71b8 ffe0        mvz.b   SWIACK.w,%d0    # pst   = 01, 08
                                                # ddata = 20, 20
                                                #         rdata.b = 0x00
010a0: 0c80 0000 0041   cmpi.l  %d0,&0x41       # pst   = 01
010a6: 6f08             ble.b   _isr_exit       # pst   = 05 (taken branch)
010b0: 201f             mov.l   (%sp)+,%d0      # pst   = 01
010b2: 205f             mov.l   (%sp)+,%a0      # pst   = 01
010b4: 4e73             rte                     # pst   = 07, 03, 05, 0d
                                                # ddata = 29, 21, 2a, 22
                                                #         trg_addr = 2a19 << 1
                                                #         trg_addr = 5432
```

As the PSTs are compressed, the resulting stream of 6-bit hexadecimal entries is loaded into consecutive locations in the PST trace buffer:

```
PSTB[*]= 1c, 1c, 05, 0d,   // interrupt exception
         2a, 23, 28, 20,   // branch target addr = 1074
         1a,               // 10 sequential insts
         13,               // 3 sequential insts
         05, 12,           // taken_branch + 2 sequential
         07, 03, 05, 0d,   // rte, entry into user mode
         29, 21, 2a, 22    // branch target addr = 5432
```

Architectural studies on the compression algorithm determined an appropriate size for the PST trace buffer. Using a suite of ten MCU benchmarks, a 64-entry PSTB was found to capture an average window of time of 520 processor cycles with program trace using 2-byte addresses enabled.

26.4.3.4 Processor Status, Debug Data Definition

This section specifies the ColdFire processor and debug module's generation of the processor status (PST) and debug data (DDATA) output on an instruction basis. In general, the PST/DDATA output for an instruction is defined as follows:

PST = 0x01, {PST = 0x0[89B], DDATA = operand}

where the {...} definition is optional operand information defined by the setting of the CSR, and [...] indicates the presence of one value from the list.

The CSR provides capabilities to display operands based on reference type (read, write, or both). A PST value {0x08, 0x09, or 0x0B} identifies the size and presence of valid data to follow in the PST trace buffer (PSTB) {1, 2, or 4 bytes, respectively}. Additionally, CSR[DDC] specifies whether operand data capture is enabled and what size. Also, for certain change-of-flow instructions, CSR[BTB] provides the capability to display the target instruction address in the PSTB (2 or 3 bytes) using a PST value of 0x0D or 0x0E, respectively.

26.4.3.4.1 User Instruction Set

Table 26-27 shows the PST/DDATA specification for user-mode instructions. Rn represents any {Dn, An} register. In this definition, the y suffix generally denotes the source, and x denotes the destination operand. For a given instruction, the optional operand data is displayed only for those effective addresses referencing memory. The DD nomenclature refers to the DDATA outputs.

Table 26-27. PST/DDATA Specification for User-Mode Instructions

Instruction	Operand Syntax	PST/DDATA
add.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
add.l	Dy,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
adda.l	<ea>y,Ax	PST = 0x01, {PST = 0x0B, DD = source operand}
addi.l	#<data>,Dx	PST = 0x01
addq.l	#<data>,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
addx.l	Dy,Dx	PST = 0x01
and.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
and.l	Dy,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
andi.l	#<data>,Dx	PST = 0x01
asl.l	{Dy,#<data>},Dx	PST = 0x01
asr.l	{Dy,#<data>},Dx	PST = 0x01
bcc.{b,w,l}		if taken, then PST = 0x05, else PST = 0x01
bchg.{b,l}	#<data>,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
bchg.{b,l}	Dy,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
bclr.{b,l}	#<data>,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}

Table 26-27. PST/DDATA Specification for User-Mode Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
bclr.{b,l}	Dy,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
bitrev.l	Dx	PST = 0x01
bra.{b,w,l}		PST = 0x05
bset.{b,l}	#<data>,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
bset.{b,l}	Dy,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
bsr.{b,w,l}		PST = 0x05, {PST = 0x0B, DD = destination operand}
btst.{b,l}	#<data>,<ea>x	PST = 0x01, {PST = 0x08, DD = source operand}
btst.{b,l}	Dy,<ea>x	PST = 0x01, {PST = 0x08, DD = source operand}
byterev.l	Dx	PST = 0x01
clr.b	<ea>x	PST = 0x01, {PST = 0x08, DD = destination operand}
clr.l	<ea>x	PST = 0x01, {PST = 0x0B, DD = destination operand}
clr.w	<ea>x	PST = 0x01, {PST = 0x09, DD = destination operand}
cmp.b	<ea>y,Dx	PST = 0x01, {0x08, source operand}
cmp.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
cmp.w	<ea>y,Dx	PST = 0x01, {0x09, source operand}
cmpa.l	<ea>y,Ax	PST = 0x01, {PST = 0x0B, DD = source operand}
cmpa.w	<ea>y,Ax	PST = 0x01, {0x09, source operand}
cmpi.b	#<data>,Dx	PST = 0x01
cmpi.l	#<data>,Dx	PST = 0x01
cmpi.w	#<data>,Dx	PST = 0x01
eor.l	Dy,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
eori.l	#<data>,Dx	PST = 0x01
ext.l	Dx	PST = 0x01
ext.w	Dx	PST = 0x01
extb.l	Dx	PST = 0x01
illegal		PST = 0x01[1]
jmp	<ea>y	PST = 0x05, {PST = 0x0[DE], DD = target address}[2]
jsr	<ea>y	PST = 0x05, {PST = 0x0[DE], DD = target address}, {PST = 0x0B, DD = destination operand}[2]
lea.l	<ea>y,Ax	PST = 0x01
link.w	Ay,#<displacement>	PST = 0x01, {PST = 0x0B, DD = destination operand}
lsl.l	{Dy,#<data>},Dx	PST = 0x01
lsr.l	{Dy,#<data>},Dx	PST = 0x01

Table 26-27. PST/DDATA Specification for User-Mode Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
mov3q.l	#<data>,<ea>x	PST = 0x01, {PST = 0x0B,DD = destination operand}
move.b	<ea>y,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
move.l	<ea>y,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
move.w	<ea>y,<ea>x	PST = 0x01, {PST = 0x09, DD = source}, {PST = 0x09, DD = destination}
move.w	CCR,Dx	PST = 0x01
move.w	{Dy,#<data>},CCR	PST = 0x01
movea.l	<ea>y,Ax	PST = 0x01, {PST = 0x0B, DD = source}
movea.w	<ea>y,Ax	PST = 0x01, {PST = 0x09, DD = source}
movem.l	#list,<ea>x	PST = 0x01, {PST = 0x0B, DD = destination},...
movem.l	<ea>y,#list	PST = 0x01, {PST = 0x0B, DD = source},...
moveq.l	#<data>,Dx	PST = 0x01
muls.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
muls.w	<ea>y,Dx	PST = 0x01, {PST = 0x09, DD = source operand}
mulu.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
mulu.w	<ea>y,Dx	PST = 0x01, {PST = 0x09, DD = source operand}
mvs.b	<ea>y,Dx	PST = 0x01, {0x08, source operand}
mvs.w	<ea>y,Dx	PST = 0x01, {0x09, source operand}
mvz.b	<ea>y,Dx	PST = 0x01, {0x08, source operand}
mvz.w	<ea>y,Dx	PST = 0x01, {0x09, source operand}
neg.l	Dx	PST = 0x01
negx.l	Dx	PST = 0x01
nop		PST = 0x01
not.l	Dx	PST = 0x01
or.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
or.l	Dy,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
ori.l	#<data>,Dx	PST = 0x01
pea.l	<ea>y	PST = 0x01, {PST = 0x0B, DD = destination operand}
pulse		PST = 0x04
rts		PST = 0x01, {PST = 0x0B, DD = source operand}, PST = 0x05, {PST = 0x0[DE], DD = target address}
sats.l	Dx	PST = 0x01
scc.b	Dx	PST = 0x01
sub.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}

Table 26-27. PST/DDATA Specification for User-Mode Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
sub.l	Dy,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
suba.l	<ea>y,Ax	PST = 0x01, {PST = 0x0B, DD = source operand}
subi.l	#<data>,Dx	PST = 0x01
subq.l	#<data>,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
subx.l	Dy,Dx	PST = 0x01
swap.w	Dx	PST = 0x01
tas.b	<ea>x	PST = 0x01, {0x08, source}, {0x08, destination}
tpf		PST = 0x01
tpf.l	#<data>	PST = 0x01
tpf.w	#<data>	PST = 0x01
trap	#<data>	PST = 0x01[1]
tst.b	<ea>x	PST = 0x01, {PST = 0x08, DD = source operand}
tst.l	<ea>y	PST = 0x01, {PST = 0x0B, DD = source operand}
tst.w	<ea>y	PST = 0x01, {PST = 0x09, DD = source operand}
unlk	Ax	PST = 0x01, {PST = 0x0B, DD = destination operand}
wddata.b	<ea>y	PST = 0x04, {PST = 0x08, DD = source operand}
wddata.l	<ea>y	PST = 0x04, {PST = 0x0B, DD = source operand}
wddata.w	<ea>y	PST = 0x04, {PST = 0x09, DD = source operand}

[1] During normal exception processing, the PSTB is loaded with two successive 0x1C entries indicating the exception processing state. The exception stack write operands, as well as the vector read and target address of the exception handler may also be displayed.

```
Exception Processing:
      PST = 0x1C, 0x1C,
     {PST = 0x0B,DD = destination},              // stack frame
     {PST = 0x0B,DD = destination},              // stack frame
     {PST = 0x0B,DD = source},                   // vector read
      PST = 0x05,{PST = 0x0[DE],DD = target}    // handler PC
```

A similar set of PST/DD values is generated in response to an emulator mode excetion. For these events (caused by a debug interrupt or properly-enabled trace exception), the initial PST values are 0x1D, 0x1D and the remaining sequence is equivalent to normal exception processing.

The PST/DDATA specification for the reset exception is shown below:

```
Exception Processing:
     PST = 0x1C, 0x1C,
     PST = 0x05,{PST = 0x0[DE],DD = target}   //  initial PC
```

The initial references at address 0 and 4 are never captured nor displayed because these accesses are treated as instruction fetches.

For all types of exception processing, the PST = 0x1C (or 0x1D) value is driven for two trace buffer entries.

[2] For JMP and JSR instructions, the optional target instruction address is displayed only for those effective address fields defining variant addressing modes. This includes the following <ea>x values: (An), (d16,An), (d8,An,Xi), (d8,PC,Xi).

Table 26-28 shows the PST/DDATA specification for the MAC instructions.

Table 26-28. PST/DDATA Values for Multiply-Accumulate Instructions

Instruction	Operand Syntax	PST/DDATA
mac.l	Ry,Rx	PST = 0x1
mac.l	Ry,Rx,<ea>y,Rw	PST = 0x1, {PST = 0xB, DD = source operand}
mac.w	Ry,Rx	PST = 0x1
mac.w	Ry,Rx,ea,Rw	PST = 0x1, {PST = 0xB, DD = source operand}
move.l	{Ry,#<data>},ACC	PST = 0x1
move.l	{Ry,#<data>},MACSR	PST = 0x1
move.l	{Ry,#<data>},MASK	PST = 0x1
move.l	ACC,Rx	PST = 0x1
move.l	MACSR,CCR	PST = 0x1
move.l	MACSR,Rx	PST = 0x1
move.l	MASK,Rx	PST = 0x1
msac.l	Ry,Rx	PST = 0x1
msac.l	Ry,Rx,<ea>y,Rw	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}
msac.w	Ry,Rx	PST = 0x1
msac.w	Ry,Rx,<ea>y,Rw	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}

26.4.3.4.2 Supervisor Instruction Set

The supervisor instruction set has complete access to the user mode instructions plus the opcodes shown below. The PST/DDATA specification for these opcodes is shown in Table 26-29.

Table 26-29. PST/DDATA Specification for Supervisor-Mode Instructions

Instruction	Operand Syntax	PST/DDATA
halt		PST = 0x1F, PST = 0x1F
move.l	Ay,USP	PST = 0x01
move.l	USP,Ax	PST = 0x01
move.w	SR,Dx	PST = 0x01
move.w	{Dy,#<data>},SR	PST = 0x01, {PST = 0x03}
movec.l	Ry,Rc	PST = 0x01

Table 26-29. PST/DDATA Specification for Supervisor-Mode Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
rte		PST = 0x07, {PST = 0x0B, DD = source operand}, {PST = 0x03}, {PST = 0x0B, DD = source operand}, PST = 0x05, {PST = 0x0[DE], DD = target address}
stldsr.w	#imm	PST = 0x01, {PST = 0x0B, DD = destination operand, PST = 0x03}
stop	#<data>	PST = 0x1E, PST = 0x1E
wdebug.l	<ea>y	PST = 0x01, {PST = 0x0B, DD = source, PST = 0x0B, DD = source}

The move-to-SR, STLDSR, and RTE instructions include an optional PST = 0x3 value, indicating an entry into user mode.

Similar to the exception processing mode, the stopped state (PST = 0x1E) and the halted state (PST = 0x1F) display this status for two entries when the ColdFire processor enters the given mode.

26.4.4 Freescale-Recommended BDM Pinout

Typically, a relatively simple interface pod is used to translate commands from a host computer into commands for the custom serial interface to the single-wire background debug system. Depending on the development tool vendor, this interface pod may use a standard RS-232 serial port, a parallel printer port, or some other type of communications such as a universal serial bus (USB) to communicate between the host PC and the pod. The pod typically connects to the target system with ground, the BKGD pin, $\overline{\text{RESET}}$, and sometimes V_{DD}. An open-drain connection to reset allows the host to force a target system reset, useful to regain control of a lost target system or to control startup of a target system before the on-chip nonvolatile memory has been programmed. Sometimes V_{DD} can be used to allow the pod to use power from the target system to avoid the need for a separate power supply. However, if the pod is powered separately, it can be connected to a running target system without forcing a target system reset or otherwise disturbing the running application program.

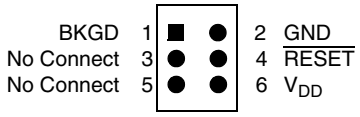


Figure 26-26. Recommended BDM Connector

Appendix A
Revision History

This appendix lists major changes between versions of the MCF51EM256 Series ColdFire Integrated Microcontroller Reference Manual.

A.1 Changes Between Rev. 4 and Rev. 5

Table A-1. MCF51EM256RM Rev. 4 to Rev. 5 Changes

Chapter	Description
Device Overview	Updated MCU block diagram, removed 32 KB option for flash1 and flash2; removed 4/2 KB options for RAM. Removed Simplified CF1Core and Low-Cost Platform Block Diagram and its table. Added a note to Table 1-7.
Rapid GPIO	Corrected the register name at 0x(FF)FF_FFF4. Removed V1 Corefire platform block diagram and related information.
PDB	Updated to use the latest PDB block guide.

A.2 Changes Between Rev. 5 and Rev. 6

Table A-2. MCF51EM256RM Rev. 5 to Rev. 6 Changes

Chapter	Description
Throughout	Corrected the PRACMP pin out to be consistent with those in the Chapter 25, "Programmable Reference Analog Comparator (PRACMP)." Changed the V_{DDAD} to V_{DDA} and V_{SSAD} to V_{SSA}.
Device Overview	Corrected the version of SPI1 module in the Table 1-4.
Memory	Corrected Bit 0 of ICSSC register, IRTC_CFG_DATA register, Bit 3 of SPR1BR, bits of SPI1S, Bit 6–7 of SPI2C2; bits of IRTC_TTSR_MY register, Bit 7 of ADCxSC1B register, Bit 5–7 of PDBSC, Bit 14–15 of PDBCHxCR, LCDBPENx to BPENx, INTCPL6P7 register, INTCPL6P6 register, PRACMP registers, ADCxCFG registers, ADCxCFG2 registers, ADCxSC2 registers, ADCxSC3 registers in the Table 3-3. Some of the registers in this table changed their names: PDBSCR changed to PDBSC, PDBCOUNT changed to PDBCNT, PDBIDELAY changed to PDBIDLY, PDBCHnDELA changed to PDBCHnDLYA, PDBCHnDELB changed to PDBCHnDLYB BPENx to LCDBPENx. Added SOPT2 register, SPICI register. Changed the name of the flash registers.
Modes of Operation	Corrected the wakeup pins used to exit from stop2 in Section 6.8.1, "Stop2 Mode."
Resets, Interrupts, and General System Control	Corrected the PDB interrupts in the Table 7-1. Added Section 7.7.7, "System Options 2 Register (SOPT2)." Corrected the Description in the Table 7-14. Changed the Reset to information of bit 2 in the Figure 7-11 and added a figure note to it. Corrected the Description of VREF in the Table 7-16.

Table A-2. MCF51EM256RM Rev. 5 to Rev. 6 Changes (continued)

Chapter	Description
Interrupt Controller	Changed the vector of number 67 to PDB_err in the Table 10-2. Correct the ENB bit status after reset to 1 in the Figure 10-4. Added an interrupt source of PDB_err in Level 7 Priority 1 of Table 10-12.
Voltage Reference	Updated the MODE description.

A.3 Changes Between Rev. 6 and Rev. 7

Table A-3. MCF51EM256RM Rev. 6 to Rev. 7 Changes

Chapter	Description
Throughout	Updated pinout information to PTB0/RGPIO8/KBI1P0/PRACMP2P0/RX2 and PTB2/RGPIO10/KBI1P2/PRACMP1P0/RX1. Changed all the robust RTC to IRTC.
Device Overview	Updated the version of IRTC and GPIO in the Table 1-4 Versions of On-Chip Modules.
Pins and Connections	Added a note to the Table 2-23 ADC Pinout Summary.
Memory	Updated the register information at address 0x(FF)FF_845D, 0x(FF)FF_849D, and 0x(FF)FF_84DD
Resets, Interrupts, and General System Control	Corrected the heading information to COP window Opens (SOPT1[COPW] = 1) in the Table 7-11 COP Configuration Details. Updated the reset information of bit 4 in the Figure 7-7 System Options 2 Register (SOPT2). Updated the description of PMC_LVD_TRIM in the Table 7-12 SOPT2 Bit Field Descriptions.
Independent Real Time Clock	Updated “any change of state on these pins will indicate a tamper” to “high level input on this pin will indicate a tamper” in the Section 17.4, “Overview.” Corrected the reset to information of Bit 0 in the Figure 17-13 IRTC Interrupt Enable Register (IRTC_IER). Updated “Any state change on these bits is considered as a tamper and an interrupt is generated to CPU.” to “High level input on this pin is considered as a tamper and an interrupt is generated if the tamper interrupt is enabled.”
Internal Clock Source (ICS)	Updated the title of Table 11-1 ICS Clock Check and Select Block Diagram.
Analog-to-Digital Converter (ADC16)	Updated Table 21-1 ADC Channel Assignments. Removed “Calibration is recommended to be done after any reset.” in the Section 21.5, “Functional Description.” Added “For best calibration results it is recommended to set hardware averaging to maximum (AVGE=1, AVGS=11 for average of 32), ADC clock frequency fADCK less than or equal to 4 MHz, VREFH=VDDAD and to calibrate at nominal voltage and temperature.” and removed “hardware average function,” in the Section 21.5.7, “Calibration Function.”

A.4 Changes Between Rev. 7 and Rev. 8

Table A-4. MCF51EM256RM Rev. 7 to Rev. 8 Changes

Chapter	Description
Device Overview	Updated the descriptions of SPI in the Table 1-3.
Pins and Connections	Updated Figure 2-3.
Memory	Updated the registers at Base+0x040C in the Table 3-15.
Programmable Reference Analog Comparator (PRACMP)	Updated Table 25-1.