MCF5213 ColdFire$^{®}$ Integrated Microcontroller Reference Manual

Additional Devices Supported:
MCF5211
MCF5212

MCF5213RM
Rev 1.1
07/2005

freescale™

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

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)
support@freescale.com

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 Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 26668334
support.asia@freescale.com

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

freescale™
semiconductor

MCF5213RM
Rev 1.1
07/2005

Contents

Chapter 1
Overview

Contents

Contents

Chapter 4
Hardware Multiply/Accumulate (MAC) Unit

Contents

Contents

Chapter 7
Power Management

Contents

Chapter 8
Chip Configuration Module (CCM)

Chapter 9
Reset Controller Module

Contents

Chapter 10
System Control Module (SCM)

Contents

Chapter 11
General Purpose I/O Module

Chapter 12
Interrupt Controller Module

Contents

Contents

Chapter 15
ColdFire Flash Module (CFM)

Contents

Chapter 16
EzPort

Chapter 17
Programmable Interrupt Timer Modules (PIT0–PIT1)

Contents

Chapter 18
General Purpose Timer Module (GPT)

Contents

Chapter 19
DMA Timers (DTIM0–DTIM3)

Chapter 20
Queued Serial Peripheral Interface (QSPI)

Contents

Chapter 21
UART Modules

Contents

Chapter 22
I^{2}C Interface

Contents

Chapter 23
Analog-to-Digital Converter (ADC)

Contents

MCF5213 Reference Manual, Rev. 1.1

Contents

Chapter 24
Pulse Width Modulation (PWM) Module

Contents

Chapter 25
FlexCAN

Contents

Chapter 26
Debug Module

Contents

Chapter 27
IEEE 1149.1 Test Access Port (JTAG)

Contents

Appendix A
Register Memory Map

About This Book

The primary objective of this reference manual is to define the functionality of the MCF5213 processor for use by software and hardware developers. In addition, this manual supports the MCF5211 and MCF5212. This book is written from the perspective of the MCF5213, and unless otherwise noted, the information also applies to the MCF5211 and MCF5212. The MCF5211 and MCF5212 have the same functionality as the MCF5213, and any differences in data regarding bus timing, signal behavior, and AC, DC, and thermal characteristics are in the data sheet.

The information in this book is subject to change without notice, as described in the disclaimers on the title page. As with any technical documentation, it is the reader's responsibility to be sure he is using the most recent version of the documentation.

To locate any published errata or updates for this document, refer to the world-wide web at http://www.freescale.com/coldfire.

Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products with the MCF5213. It is assumed that the reader understands operating systems, microprocessor system design, basic principles of software and hardware, and basic details of the ColdFire® architecture.

Organization

Following is a summary and brief description of the major sections of this manual:

- Chapter 1, "Overview," includes general descriptions of the modules and features on the device, focusing in particular on new features.
- Chapter 2, "Signal Descriptions," describes the device signals. It includes a listing of signals that characterizes each signal as an input or output, defines its state at reset, and identifies whether a pull-up resistor should be used.
- Chapter 3, "ColdFire Core," provides an overview of the microprocessor core. It describes the organization of the Version 2 (V2) ColdFire processor core and includes an overview of the programming model as they are implemented on the device.
- Chapter 4, "Hardware Multiply/Accumulate (MAC) Unit," describes the multiply/accumulate (MAC) unit, which executes integer multiply, multiply-accumulate, and miscellaneous register instructions. The MAC is integrated into the operand execution pipeline (OEP).
- Chapter 5, "Static RAM (SRAM)," covers general operations, configuration, and initialization of the on-chip static RAM (SRAM) implementation. It also provides information and examples of how to minimize power consumption when using the SRAM.
- Chapter 6, "Clock Module," describes the device's different clocking methods. It also describes clock module operation in low power modes.

- Chapter 7, "Power Management," describes the low power operation of the device and peripheral behavior in low power modes.
- Chapter 8, "Chip Configuration Module (CCM)," details the various operating configurations of the device. It provides a description of signals used by the CCM and a programming model.
- Chapter 9, "Reset Controller Module," describes the operation of the reset controller module, detailing the different types of reset that can occur.
- Chapter 10, "System Control Module (SCM)," describes the functionality of the SCM, which provides the programming model for peripheral access control, the software core watchdog timer (CWT), and the generic access error information.
- Chapter 11, "General Purpose I/O Module," describes the operation and programming model of the general purpose I/O (GPIO) ports on the device.
- Chapter 12, "Interrupt Controller Module," describes operation of the interrupt controller portion of the SCM. It includes descriptions of the registers in the interrupt controller memory map and the interrupt priority scheme.
- Chapter 13, "Edge Port Module (EPORT)," describes EPORT module functionality, including operation in low power mode.
- Chapter 14, "DMA Controller Module," describes the direct memory access (DMA) controller module. It provides an overview of the module and describes in detail its signals and registers. The latter sections of this chapter describe operations, features, and supported data transfer modes in detail.
- Chapter 15, "ColdFire Flash Module (CFM)," describes implementation of the SuperFlash® technology licensed from SST used on this device. The ColdFire Flash Module (CFM) is constructed with four banks of 32K x 16-bit Flash to generate a 256-Kbyte, 32-bit wide electrically erasable and programmable read-only memory array. The CFM is ideal for program and data storage for single-chip applications and allows for field reprogramming without external high-voltage sources.
- Chapter 16, "EzPort," describes the interface that allows the Flash memory contents on a 32 bit general purpose microcontroller to be read, erased and programmed from off-chip in a format compatible to many standalone Flash memory chips.
- Chapter 17, "Programmable Interrupt Timer Modules (PIT0–PIT1)," describes the functionality of the PIT timers, including operation in low power mode.
- Chapter 18, "General Purpose Timer Module (GPT)," describes the functionality of the 4-channel general purpose timer module (GPT), including the configuration of channel 3 as a 16-bit pulse accumulator that can operate as a simple event counter or as a gated time accumulator.
- Chapter 19, "DMA Timers (DTIM0–DTIM3)," describes the configuration and operation of the four direct memory access (DMA) timer modules (DTIM0, DTIM1, DTIM2, and DTIM3). These 32-bit timers provide input capture and reference compare capabilities with optional signaling of events using interrupts or DMA triggers. Additionally, programming examples are included.
- Chapter 20, "Queued Serial Peripheral Interface (QSPI)," provides a feature-set overview and a description of operation, including details of the QSPI's internal storage organization. The chapter concludes with the programming model and a timing diagram.
- Chapter 21, "UART Modules," describes the use of the universal asynchronous receiver/transmitters (UARTs) implemented on the device and includes programming examples.
- Chapter 22, "I^{2}C Interface," describes the I^{2}C module, including I^{2}C protocol, clock synchronization, and I^{2}C programming model registers.

- Chapter 23, "Analog-to-Digital Converter (ADC)," describes the two separate and complete ADCs, each with their own sample and hold circuits and a common voltage reference and common digital control module.
- Chapter 24, "Pulse Width Modulation (PWM) Module," describes the configuration and operation of the pulse width modulation (PWM) module. It includes a block diagram, programming model, and functional description.
- Chapter 25, "FlexCAN," describes the implementation of the controller area network (CAN) protocol. It describes FlexCAN module operation and provides a programming model.
- Chapter 26, "Debug Module," describes the hardware debug support in the device.
- Chapter 27, "IEEE 1149.1 Test Access Port (JTAG)," describes configuration and operation of the Joint Test Action Group (JTAG) implementation. It describes those items required by the IEEE 1149.1 standard and provides additional information specific to the device. For internal details and sample applications, see the IEEE 1149.1 document.
- Appendix A, "Register Memory Map," summarizes the address, name, and byte assignment for registers within the CPU space, lists an overview of the memory map for the on-chip modules, and provides a detailed memory map including all of the registers for on-chip modules.

Additional literature is published as new processors become available. For a current list of ColdFire documentation, refer to http://www.freescale.com/coldfire.

Conventions

This document uses the following notational conventions:

cleared/set	When a bit takes the value zero, it is said to be cleared; when it takes a value of one, it is said to be set.
MNEMONICS	In text, instruction mnemonics are shown in uppercase.
`mnemonics`	In code and tables, instruction mnemonics are shown in lowercase.
italics	Italics indicate variable command parameters. Book titles in text are set in italics.
0x0	Prefix to denote hexadecimal number
0b0	Prefix to denote binary number
REG[FIELD]	Abbreviations for registers are shown in uppercase. Specific bits, fields, or ranges appear in brackets. For example, RAMBAR[BA] identifies the base address field in the RAM base address register.
nibble	A 4-bit data unit
byte	An 8-bit data unit
word	A 16-bit data unit[1]
longword	A 32-bit data unit
x	In some contexts, such as signal encodings, x indicates a don't care.
n	Used to express an undefined numerical value
~	NOT logical operator

1. The only exceptions to this appear in the discussion of serial communication modules that support variable-length data transmission units. To simplify the discussion these units are referred to as words regardless of length.

&	AND logical operator
|	OR logical operator
$\overline{\text{OVERBAR}}$	An overbar indicates that a signal is active-low.

Register Figure Conventions

This document uses the following conventions for the register reset values:

—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by the polarity of the indicated signal.

The following register fields are used:

R	0
W	

Indicates a reserved bit field in a memory-mapped register. These bits are always read as zeros.

R	1
W	

Indicates a reserved bit field in a memory-mapped register. These bits are always read as ones.

R	FIELDNAME
W	

Indicates a read/write bit.

R	FIELDNAME
W	

Indicates a read-only bit field in a memory-mapped register.

R	
W	FIELDNAME

Indicates a write-only bit field in a memory-mapped register.

R	FIELDNAME
W	w1c

Write 1 to clear: indicates that writing a 1 to this bit field clears it.

R	0
W	FIELDNAME

Indicates a self-clearing bit.

Acronyms and Abbreviations

Table i lists acronyms and abbreviations used in this document.

Table i. Acronyms and Abbreviated Terms

Term	Meaning
ADC	Analog-to-digital conversion
ALU	Arithmetic logic unit
BDM	Background debug mode

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
BIST	Built-in self test
BSDL	Boundary-scan description language
CODEC	Code/decode
DAC	Digital-to-analog conversion
DMA	Direct memory access
DSP	Digital signal processing
EA	Effective address
FIFO	First-in, first-out
GPIO	General-purpose I/O
I^{2}C	Inter-integrated circuit
IEEE	Institute for Electrical and Electronics Engineers
IFP	Instruction fetch pipeline
IPL	Interrupt priority level
JEDEC	Joint Electron Device Engineering Council
JTAG	Joint Test Action Group
LIFO	Last-in, first-out
LRU	Least recently used
LSB	Least-significant byte
lsb	Least-significant bit
MAC	Multiply accumulate unit, also Media access controller
MBAR	Memory base address register
MSB	Most-significant byte
msb	Most-significant bit
Mux	Multiplex
NOP	No operation
OEP	Operand execution pipeline
PC	Program counter
PCLK	Processor clock
PLIC	Physical layer interface controller
PLL	Phase-locked loop
POR	Power-on reset
PQFP	Plastic quad flat pack
PWM	Pulse width modulation

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
QSPI	Queued serial peripheral interface
RISC	Reduced instruction set computing
Rx	Receive
SIM	System integration module
SOF	Start of frame
TAP	Test access port
TTL	Transistor transistor logic
Tx	Transmit
UART	Universal asynchronous/synchronous receiver transmitter

Terminology Conventions

Table ii shows terminology conventions used throughout this document.

Table ii. Notational Conventions

Instruction	Operand Syntax
Opcode Wildcard	
cc	Logical condition (example: NE for not equal)
Register Specifications	
An	Any address register n (example: A3 is address register 3)
Ay,Ax	Source and destination address registers, respectively
Dn	Any data register n (example: D5 is data register 5)
Dy,Dx	Source and destination data registers, respectively
Rc	Any control register (example VBR is the vector base register)
Rm	MAC registers (ACC, MAC, MASK)
Rn	Any address or data register
Rw	Destination register w (used for MAC instructions only)
Ry,Rx	Any source and destination registers, respectively
Xi	Index register i (can be an address or data register: Ai, Di)
Miscellaneous Operands	
#<data>	Immediate data following the 16-bit operation word of the instruction
<ea>	Effective address
<ea>y,<ea>x	Source and destination effective addresses, respectively

Table ii. Notational Conventions (continued)

Instruction	Operand Syntax
<label>	Assembly language program label
<list>	List of registers for MOVEM instruction (example: D3–D0)
<shift>	Shift operation: shift left (<<), shift right (>>)
<size>	Operand data size: byte (B), word (W), longword (L)
bc	Both instruction and data caches
dc	Data cache
ic	Instruction cache
# <vector>	Identifies the 4-bit vector number for trap instructions
<>	identifies an indirect data address referencing memory
<xxx>	identifies an absolute address referencing memory
d*n*	Signal displacement value, *n* bits wide (example: d16 is a 16-bit displacement)
SF	Scale factor (x1, x2, x4 for indexed addressing mode, <<1n>> for MAC operations)
Operations	
+	Arithmetic addition or postincrement indicator
–	Arithmetic subtraction or predecrement indicator
x	Arithmetic multiplication
/	Arithmetic division
~	Invert; operand is logically complemented
&	Logical AND
		Logical OR
^	Logical exclusive OR
<<	Shift left (example: D0 << 3 is shift D0 left 3 bits)
>>	Shift right (example: D0 >> 3 is shift D0 right 3 bits)
→	Source operand is moved to destination operand
←→	Two operands are exchanged
sign-extended	All bits of the upper portion are made equal to the high-order bit of the lower portion
If <condition> then <operations> else <operations>	Test the condition. If true, the operations after 'then' are performed. If the condition is false and the optional 'else' clause is present, the operations after 'else' are performed. If the condition is false and else is omitted, the instruction performs no operation. Refer to the Bcc instruction description as an example.
Subfields and Qualifiers	
{}	Optional operation
()	Identifies an indirect address

Table ii. Notational Conventions (continued)

Instruction	Operand Syntax
d$_n$	Displacement value, n-bits wide (example: d$_{16}$ is a 16-bit displacement)
Address	Calculated effective address (pointer)
Bit	Bit selection (example: Bit 3 of D0)
lsb	Least significant bit (example: lsb of D0)
LSB	Least significant byte
LSW	Least significant word
msb	Most significant bit
MSB	Most significant byte
MSW	Most significant word

Revision History

Table iii provides a revision history for this document.

Table iii. MCF5213 Reference Manual Revision History

Revision Number	Date of Release	Substantive Changes
1	05/2005	Initial public release.
1.1	07/2005	Interim release with corrections.

Chapter 1
Overview

This chapter provides an overview of the major features and functional components of the MCF5213 family of microcontrollers. The MCF5213 family is a highly integrated implementation of the ColdFire® family of reduced instruction set computing (RISC) microcontrollers that also includes the MCF5211 and MCF5212. The differences between these parts are summarized in Table 1-1. This document is written from the perspective of the MCF5213.

The MCF5213 represents a family of highly-integrated 32-bit microcontrollers based on the V2 ColdFire microarchitecture. Featuring up to 32 Kbytes of internal SRAM and 256 Kbytes of Flash memory, four 32-bit timers with DMA request capability, a 4-channel DMA controller, a CAN module, an I²C™ module, 3 UARTs and a queued SPI, the MCF5213 family has been designed for general purpose industrial control applications.

This 32-bit device is based on the Version 2 ColdFire reduced instruction set computer (RISC) core with a multiply-accumulate unit (MAC) and divider providing 76 Drystone 2.1 MIPS at a frequency up to 80 MHz from internal Flash. On-chip modules include the following:

- V2 ColdFire core with multiply-accumulate unit (MAC)
- 32 Kbytes of internal SRAM
- 256 Kbytes of on-chip Flash memory
- Three universal asynchronous receiver/transmitters (UARTs)
- Controller area network 2.0B (FlexCAN) module
- Inter-integrated circuit (I²C) bus controller
- 12-bit analog-to-digital converter (ADC)
- Queued serial peripheral interface (QSPI) module
- Four-channel, 32-bit direct memory access (DMA) controller
- Four-channel, 32-bit input capture/output compare timers with optional DMA support
- Two 16-bit periodic interrupt timers (PITs)
- Programmable software watchdog timer
- Interrupt controller capable of handling up to 63 interrupt sources
- Clock module with 8 MHz on-chip relaxation oxcillator and integrated phase locked loop (PLL)

These devices are ideal for cost-sensitive applications requiring significant control processing for connectivity, data buffering, and user interface, as well as signal processing in a variety of key markets such as security, imaging, networking, gaming, and medical. This leading package of integration and high performance allows fast time to market through easy code reuse and extensive third party tool support.

To locate any published errata or updates for this document, refer to the ColdFire products website at http://www.freescale.com/coldfire.

1.1 MCF5213 Family Configurations

Table 1-1. MCF5213 Family Configurations

Module	5211	5212	5213
ColdFire Version 2 Core with MAC (Multiply-Accumulate Unit)	x	x	x
System Clock	66 MHz	66, 80 MHz	
Performance (Dhrystone 2.1 MIPS)	63	up to 76	
Flash / Static RAM (SRAM)	128/16 Kbytes	256/32 Kbytes	
Interrupt Controller (INTC)	x	x	x
Fast Analog-to-Digital Converter (ADC)	x	x	x
FlexCAN 2.0B Module	-	-	x
Four-channel Direct-Memory Access (DMA)	x	x	x
Watchdog Timer Module (WDT)	x	x	x
Programmable Interval Timer Module (PIT)	2	2	2
Four-Channel General Purpose Timer	3	3	3
32-bit DMA Timers	4	4	4
QSPI	x	x	x
UART(s)	3	3	3
I^2C	x	x	x
PWM	8	8	8
General Purpose I/O Module (GPIO)	x	x	x
Chip Configuration and Reset Controller Module	x	x	x
Background Debug Mode (BDM)	x	x	x
JTAG - IEEE 1149.1 Test Access Port[1]	x	x	x
Package	64-pin LQFP	81-ball MAPBGA 64-pin LQFP	81-ball MAPBGA 100-Lead LQFP

MCF5213 Reference Manual, Rev. 1.1

Freescale Semiconductor

Preliminary

1 The full debug/trace interface is available only on the 100-pin packages. A reduced debug interface is bonded on smaller packages.

1.2   Block Diagram

The superset device in the MCF5213 family comes in a 100-lead leaded quad flat package (LQFP). Figure 1-1 shows a top-level block diagram of the MCF5213.
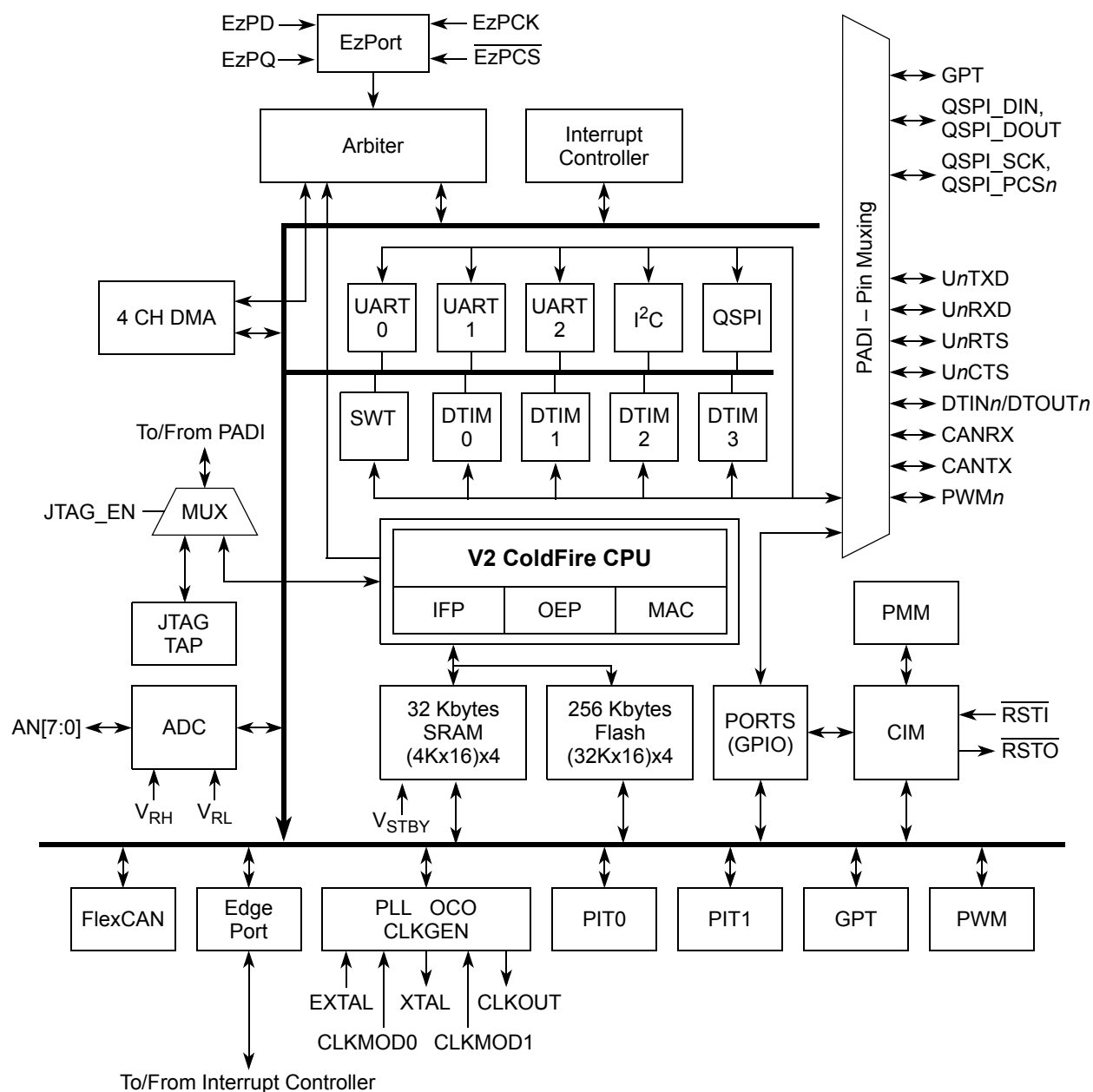
Figure 1-1. MCF5213 Block Diagram

MCF5213 Reference Manual, Rev. 1.1

1.3 Part Numbers and Packaging

Table 1-2. Part Number Summary

Part Number	Flash / SRAM	Key Features	Package	Speed
MCF5211	128 Kbytes / 16 Kbytes	3 UARTs, I²C, QSPI, A/D 16-/32-bit/PWM Timers	64-pin LQFP 81-ball MAPBGA	66 MHz
MCF5212	256 Kbytes / 32 Kbytes	3 UARTs, I²C, QSPI, A/D 16-/32-bit/PWM Timers	64-pin LQFP 81-ball MAPBGA	66 MHz 66, 80 MHz
MCF5213	256 Kbytes / 32 Kbytes	3 UARTs, I²C, QSPI, A/D 16-/32-bit/PWM Timers, CAN	81-ball MAPBGA 100-Lead LQFP	80 MHz 80 MHz

Table 1-2 summarizes the features of the MCF5213 product family. Several speed/package options are available to match cost- or performance-sensitive applications.

1.4 MCF5213 Family Features

The MCF5213 family includes the following features:

- Version 2 ColdFire variable-length RISC processor core
  - Static operation
  - 32-bit address and data paths on-chip
  - Up to 80 MHz processor core frequency
  - Sixteen general-purpose, 32-bit data and address registers
  - Implements ColdFire ISA_A+ with extensions to support the user stack pointer register and four new instructions for improved bit processing
  - Multiply-accumulate (MAC) unit with 32-bit accumulator to support $16 \times 16 \rightarrow 32$ or $32 \times 32 \rightarrow 32$ operations
  - Illegal instruction decode that allows for 68K emulation support
- System debug support
  - Revision B+ debug module on-chip
  - Real time trace for determining dynamic execution path
  - Background debug mode (BDM) for in-circuit debugging
  - Real time debug support, with six hardware breakpoints (4 PC, 1 address, and 1data) that can be configured into a 1- or 2-level trigger
- On-chip memories
  - 32 Kbyte dual-ported SRAM on CPU internal bus, supporting core and DMA access with standby power supply support
  - 256 Kbytes of interleaved Flash memory supporting 2-1-1-1 accesses

- Power management
  - Fully static operation with processor sleep and whole chip stop modes
  - Very rapid response to interrupts from the low-power sleep mode (wake-up feature)
  - Clock enable/disable for each peripheral when not used
- FlexCAN 2.0B Module
  - Based on and includes all existing features of the Freescale TouCAN module
  - Full implementation of the CAN protocol specification version 2.0B
    - Standard data and remote frames (up to 109 bits long)
    - Extended data and remote frames (up to 127 bits long)
    - 0-8 bytes data length
    - Programmable bit rate up to 1 Mbit/sec
  - Flexible message buffers (MBs), totalling up to 16 message buffers of 0–8 byte data length each, configurable as Rx or Tx, all supporting standard and extended messages
  - Unused MB space can be used as general purpose RAM space
  - Listen only mode capability
  - Content-related addressing
  - No read/write semaphores
  - Three programmable mask registers: global for MBs 0-13, special for MB14, and special for MB15
  - Programmable transmit-first scheme: lowest ID or lowest buffer number
  - Time stamp based on 16-bit free-running timer
  - Global network time, synchronized by a specific message
  - Maskable interrupts
- Three universal asynchronous/synchronous receiver transmitters (UARTs)
  - 16-bit divider for clock generation
  - Interrupt control logic
  - Maskable interrupts
  - DMA support
  - Data formats can be 5, 6, 7, or 8 bits with even, odd, or no parity
  - Up to 2 stop bits in 1/16 increments
  - Error-detection capabilities
  - Modem support includes request-to-send ($\overline{URTS}$) and clear-to-send ($\overline{UCTS}$) lines for two UARTs
  - Transmit and receive FIFO buffers

- I²C Module
  - — Interchip bus interface for EEPROMs, LCD controllers, A/D converters, and keypads
  - — Fully compatible with industry-standard I²C bus
  - — Master or slave modes support multiple masters
  - — Automatic interrupt generation with programmable level
- Queued serial peripheral interface (QSPI)
  - — Full-duplex, three-wire synchronous transfers
  - — Up to four chip selects available
  - — Master mode operation only
  - — Programmable bit rates up to half the CPU clock frequency
  - — Up to 16 pre-programmed transfers
- Fast analog-to-digital converter (ADC)
  - — Eight analog input channels
  - — 12-bit resolution ± 2.5 counts accuracy
  - — Minimum 2.25 µs conversion time
  - — Simultaneous sampling of two channels for motor control applications
  - — Single-scan or continuous operation
  - — Optional interrupts on conversion complete, zero crossing (sign change), or under/over low/high limit
  - — Unused analog channels can be used as digital I/O
- Four 32-bit DMA Timers
  - — 12.5-ns resolution at 80 MHz
  - — Programmable sources for clock input, including an external clock option
  - — Programmable prescaler
  - — Input capture capability with programmable trigger edge on input pin
  - — Output compare with programmable mode for the output pin
  - — Free run and restart modes
  - — Maskable interrupts on input capture or output compare
  - — DMA trigger capability on input capture or output compare
- Four-channel general purpose timer
  - — 16-bit architecture
  - — Programmable prescaler
  - — Output pulse widths variable from microseconds to seconds
  - — Single 16-bit input pulse accumulator

- — Toggle-on-overflow feature for pulse-width modulator (PWM) generation
- — One dual-mode pulse accumulation channel per timer
- Pulse-width modulation timer
  - — Operates as eight channels with 8-bit resolution or four channels with 16-bit resolution
  - — Programmable period and duty cycle
  - — Programmable enable/disable for each channel
  - — Software selectable polarity for each channel
  - — Period and duty cycle are double buffered. Change takes effect when the end of the current period is reached (PWM counter reaches zero) or when the channel is disabled.
  - — Programmable center or left aligned outputs on individual channels
  - — Four clock sources (A, B, SA, and SB) provide for a wide range of frequencies
  - — Emergency shutdown
- Two periodic interrupt timers (PITs)
  - — 16-bit counter
  - — Selectable as free running or count down
- Core software watchdog timer
  - — 16-bit counter
  - — Low power mode support
- Clock generation features
  - — 1 to 16 MHz crystal, 8 MHz on-chip relaxation oscillator, or external oscillator reference options
  - — Relaxation oscillator NVM-trimmed to 2% accuracy
  - — 2 to 10 MHz reference frequency for normal PLL mode
  - — System can be clocked from PLL or directly from crystal oscillator or relaxation oscillator
  - — Low power modes supported
  - — 2^n ($n \leq 0 \leq 15$) low-power divider for extremely low frequency operation
- Interrupt controller
  - — Support for up to 57 interrupt sources organized as follows:
    - – 50 fully-programmable interrupt sources
    - – 7 fixed-level interrupt sources
  - — Seven external interrupt signals
  - — Unique vector number for each interrupt source
  - — Ability to mask any individual interrupt source or all interrupt sources (global mask-all)

MCF5213 Reference Manual, Rev. 1.1

Preliminary

- — Support for hardware and software interrupt acknowledge (IACK) cycles
- — Combinatorial path to provide wake-up from low power modes
- DMA controller
  - — Four fully programmable channels
  - — Dual-address transfer support with 8-, 16-, and 32-bit data capability, along with support for 16-byte (4 x 32-bit) burst transfers
  - — Source/destination address pointers that can increment or remain constant
  - — 24-bit byte transfer counter per channel
  - — Auto-alignment transfers supported for efficient block movement
  - — Bursting and cycle steal support
  - — Software-programmable DMA requesters in the UARTs (6), 32-bit timers (4), and DMA channels (4)
- Reset
  - — Separate reset in and reset out signals
  - — Seven sources of reset:
    - – Power-on reset (POR)
    - – External
    - – Software
    - – Watchdog
    - – Loss of clock
    - – Loss of lock
    - – Low-voltage detection (LVD)
  - — Status flag indication of source of last reset
- Chip integration module (CIM)
  - — System configuration during reset
  - — Selects one of six clock modes
  - — Configures output pad drive strength
  - — Unique part identification number and part revision number
- General purpose I/O interface
  - — Up to 56 bits of general purpose I/O
  - — Bit manipulation supported via set/clear functions
  - — Unused peripheral pins may be used as extra GPIO
- JTAG support for system level board testing

1.4.1 V2 Core Overview

The version 2 Coldfire processor core is comprised of two separate pipelines that are decoupled by an instruction buffer. The two-stage instruction fetch pipeline (IFP) is responsible for instruction-address generation and instruction fetch. The instruction buffer is a first-in-first-out (FIFO) buffer that holds prefetched instructions awaiting execution in the operand execution pipeline (OEP). The OEP includes two pipeline stages. The first stage decodes instructions and selects operands (DSOC); the second stage (AGEX) performs instruction execution and calculates operand effective addresses, if needed.

The V2 core implements the ColdFire instruction set architecture revision A+ with added support for a separate user stack pointer register and four new instructions to assist in bit processing. Additionally, the MCF5213 core includes the multiply-accumulate unit (MAC) for improved signal processing capabilities. The MAC implements a three-stage execution pipeline, optimized for 16 x 16 bit operations, with support for one 32-bit accumulator. Supported operands include 16- and 32-bit signed and unsigned integers, as well as signed fractional operands and a complete set of instructions to process these data types. The MAC provides support for execution of DSP operations within the context of a single processor at a minimal hardware cost.

1.4.2 Integrated Debug Module

The ColdFire processor core debug interface is provided to support system debugging in conjunction with low-cost debug and emulator development tools. Through a standard debug interface, users can access debug information; on 100-lead packages, real-time tracing capability is provided. This allows the processor and system to be debugged at full speed without the need for costly in-circuit emulators.

The on-chip breakpoint resources include a total of nine programmable 32-bit registers: an address and an address mask register, a data and a data mask register, and four PC plus one PC mask registers. These registers can be accessed through the dedicated debug serial communication channel or from the processor's supervisor mode programming model. The breakpoint registers can be configured to generate triggers by combining the address, data, and PC conditions in a variety of single- or dual-level definitions. The trigger event can be programmed to generate a processor halt or initiate a debug interrupt exception. The MCF5213 implements revision B+ of the coldfire Debug Architecture.

The MCF5213's interrupt servicing options during emulator mode allow real-time critical interrupt service routines to be serviced while processing a debug interrupt event, thereby ensuring that the system continues to operate even during debugging.

To support program trace, the V2 debug module provides processor status (PST[3:0]) and debug data (DDATA[3:0]) ports. These buses and the PSTCLK output provide execution status, captured operand data, and branch target addresses defining processor activity at the CPU's clock rate. The full debug/trace interface is available only on the 100-pin packages. However, every product

features the dedicated debug serial communication channel (DSI, DSO, DSCLK) and the ALLPST signal.

The MCF5213 includes a new debug signal, ALLPST. This signal is the logical 'AND' of the processor status (PST[3:0]) signals and is useful for detecting when the processor is in a halted state (PST[3:0] = 1111).

1.4.3   JTAG

The MCF5213 supports circuit board test strategies based on the Test Technology Committee of IEEE and the Joint Test Action Group (JTAG). The test logic includes a test access port (TAP) consisting of a 16-state controller, an instruction register, and three test registers (a 1-bit bypass register, a 256-bit boundary-scan register, and a 32-bit ID register). The boundary scan register links the device's pins into one shift register. Test logic, implemented using static logic design, is independent of the device system logic.

The MCF5213 implementation can do the following:

*   Perform boundary-scan operations to test circuit board electrical continuity
*   Sample MCF5213 system pins during operation and transparently shift out the result in the boundary scan register
*   Bypass the MCF5213 for a given circuit board test by effectively reducing the boundary-scan register to a single bit
*   Disable the output drive to pins during circuit-board testing
*   Drive output pins to stable levels

1.4.4   On-chip Memories

1.4.4.1   SRAM

The dual-ported SRAM module provides a general-purpose 32-Kbyte memory block that the ColdFire core can access in a single cycle. The location of the memory block can be set to any 32-Kbyte boundary within the 4-Gbyte address space. This memory is ideal for storing critical code or data structures and for use as the system stack. Because the SRAM module is physically connected to the processor's high-speed local bus, it can quickly service core-initiated accesses or memory-referencing commands from the debug module.

The SRAM module is also accessible by the DMA. The dual-ported nature of the SRAM makes it ideal for implementing applications with double-buffer schemes, where the processor and a DMA device operate in alternate regions of the SRAM to maximize system performance.

1.4.4.2 Flash

The ColdFire flash module (CFM) is a non-volatile memory (NVM) module that connects to the processor's high-speed local bus. The CFM is constructed with four banks of 32K x 16-bit Flash arrays to generate 256 Kbytes of 32-bit Flash memory. These arrays serve as electrically erasable and programmable, non-volatile program and data memory. The Flash memory is ideal for program and data storage for single-chip applications, allowing for field reprogramming without requiring an external high voltage source. The CFM interfaces to the ColdFire core through an optimized read-only memory controller which supports interleaved accesses from the 2-cycle Flash arrays. A backdoor mapping of the Flash memory is used for all program, erase, and verify operations, as well as providing a read datapath for the DMA. Flash memory may also be programmed via the EzPort, which is a serial Flash programming interface that allows the Flash to be read, erased and programmed by an external controller in a format compatible with most SPI bus Flash memory chips.

1.4.5 FlexCAN

The FlexCAN module on the MCF5213 (refer to Table 1-1) is a communication controller implementing the 2.0B CAN protocol. The CAN protocol is commonly used as an industrial control serial data bus, meeting the specific requirements of real-time processing, reliable operation in a harsh EMI environment, cost-effectiveness, and required bandwidth. The instantiation of FlexCAN on the MCF5213 contains 16 message buffers.

1.4.6 UARTs

The MCF5213 contains three full-duplex UARTs that function independently. The three UARTs can be clocked by the system bus clock, eliminating the need for an externally supplied clock. They can use DMA requests on transmit-ready and receive-ready, as well as interrupt requests for servicing. Flow control via $\overline{\text{U}n\text{CTS}}$ and $\overline{\text{U}n\text{RTS}}$ pins is provided on all three UARTS. On smaller packages, the third UART is multiplexed with other digital I/O functions.

1.4.7 I^{2}C Bus

The I^{2}C bus is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange, minimizing the interconnection between devices. This bus is suitable for applications requiring occasional communications over a short distance between many devices.

1.4.8 QSPI

The queued serial peripheral interface module provides a high-speed synchronous serial peripheral interface with queued transfer capability. It allows up to 16 transfers to be queued at once, eliminating CPU intervention between transfers.

1.4.9 DMA Timers (DTIM0-DTIM3)

There are four independent, DMA-transfer-generating 32-bit timers (DTIM[3:0]) on the MCF5213. Each timer module incorporates a 32-bit timer with a separate register set for configuration and control. The timers can be configured to operate from the system clock or from an external clock source using one of the DTIN*n* signals. If the system clock is selected, it can be divided by 16 or 1. The input clock is further divided by a user-programmable 8-bit prescaler which clocks the actual timer counter register (TCR*n*). Each of these timers can be configured for input capture or reference compare mode. By configuring the internal registers, each timer may be configured to assert an external signal, generate an interrupt on a particular event, or cause a DMA transfer.

1.4.10 General Purpose Timer (GPTA/GPTB)

The general purpose timer (GPTB) is a 4-channel timer module. The timer consists of a 16-bit programmable counter driven by a 7-stage programmable prescaler. Each of the four channels can be configured for input capture or output compare. Additionally, one of the channels, channel 3, can be configured as a pulse accumulator.

A timer overflow function allows software to extend the timing capability of the system beyond the 16-bit range of the counter. The input capture and output compare functions allow simultaneous input waveform measurements and output waveform generation. The input capture function can capture the time of a selected transition edge. The output compare function can generate output waveforms and timer software delays. The 16-bit pulse accumulator can operate as a simple event counter or a gated time accumulator.

1.4.11 Pulse Width Modulation Timers (PWM)

The MCF5213 has an 8-channel, 8-bit PWM timer. Each channel has a programmable period and duty cycle as well as a dedicated counter. Each of the modulators can create independent continuous waveforms with software-selectable duty rates from 0% to 100%. The PWM outputs have programmable polarity, and can be programmed as left aligned outputs or center aligned outputs. For higher period and duty cycle resolution, each pair of adjacent channels ([7:6], [5:4], [3:2], and [1:0]) can be concatenated to form a single 16-bit channel. The module can thus be configured to support 8/0, 6/1, 4/2, 2/3, or 0/4 8-/16-bit channels.

1.4.12 Periodic Interrupt Timers (PIT0 and PIT1)

The two periodic interrupt timers (PIT0 and PIT1) are 16-bit timers that provide interrupts at regular intervals with minimal processor intervention. Each timer can either count down from the value written in its PIT modulus register, or it can be a free-running down-counter.

Preliminary

1.4.13 Core Software Watchdog Timer

The watchdog timer is a 32-bit timer that facilitates recovery from runaway code. The watchdog counter is a free-running down-counter that generates a reset on underflow when enabled. To prevent a reset, software must periodically restart the countdown.

1.4.14 Clock Module and Phase Locked Loop (PLL)

The clock module contains a crystal oscillator (OSC), phase-locked loop (PLL), reduced frequency divider (RFD), status/control registers, and control logic. To improve noise immunity, the PLL and OSC have their own power supply inputs, VDDPLL and VSSPLL. All other circuits are powered by the normal supply pins, VDD and VSS.

1.4.15 Interrupt Controller (INTC)

The MCF5213 includes an interrupt controller that can support up to 57 interrupt sources. The interrupt controller is organized as 7 levels with 9 interrupt sources per level. Each interrupt source has a unique interrupt vector, and 50 of the 57 sources provide a programmable level [1-7] and priority within the level.

1.4.16 DMA Controller

The direct memory access (DMA) controller module provides an efficient way to move blocks of data with minimal processor interaction. The DMA module provides four channels (DMA0-DMA3) that allow byte, word, longword or 16-byte burst line transfers. These transfers are triggered by software explicitly setting a DCRn[START] bit. Other sources include the DMA timer and UARTs. The DMA controller supports dual address transfers to on-chip devices.

1.4.17 Reset

The reset controller determines the source of reset, asserts the appropriate reset signals to the system, and keeps track of what caused the last reset. There are seven sources of reset:

- External reset input
- Power-on reset (POR)
- Phase locked-loop (PLL) loss of lock
- PLL loss of clock
- Software
- Low-voltage detector (LVD)

Control of the LVD and its associated reset and interrupt are handled by the reset controller. Other registers provide status flags indicating the last source of reset and a control bit for software assertion of the $\overline{\text{RSTO}}$ pin.

1.4.18 GPIO

Nearly all pins on the MCF5213 have general purpose I/O capability and are grouped into 8-bit ports. Some ports do not use all eight bits. Each port has registers that configure, monitor, and control the port pins.

Chapter 2
Signal Descriptions

This chapter describes signals implemented on this device and includes an alphabetical listing of signals that characterizes each signal as an input or output, defines its state at reset, and identifies whether a pull-up resistor should be used.

NOTE

The terms 'assertion' and 'negation' are used to avoid confusion when dealing with a mixture of active-low and active-high signals. The term 'asserted' indicates that a signal is active, independent of the voltage level. The term 'negated' indicates that a signal is inactive.

Active-low signals, such as $\overline{\text{SRAS}}$ and $\overline{\text{TA}}$, are indicated with an overbar.

2.1 Overview

Figure 2-1 shows the block diagram of the device with the signal interface.
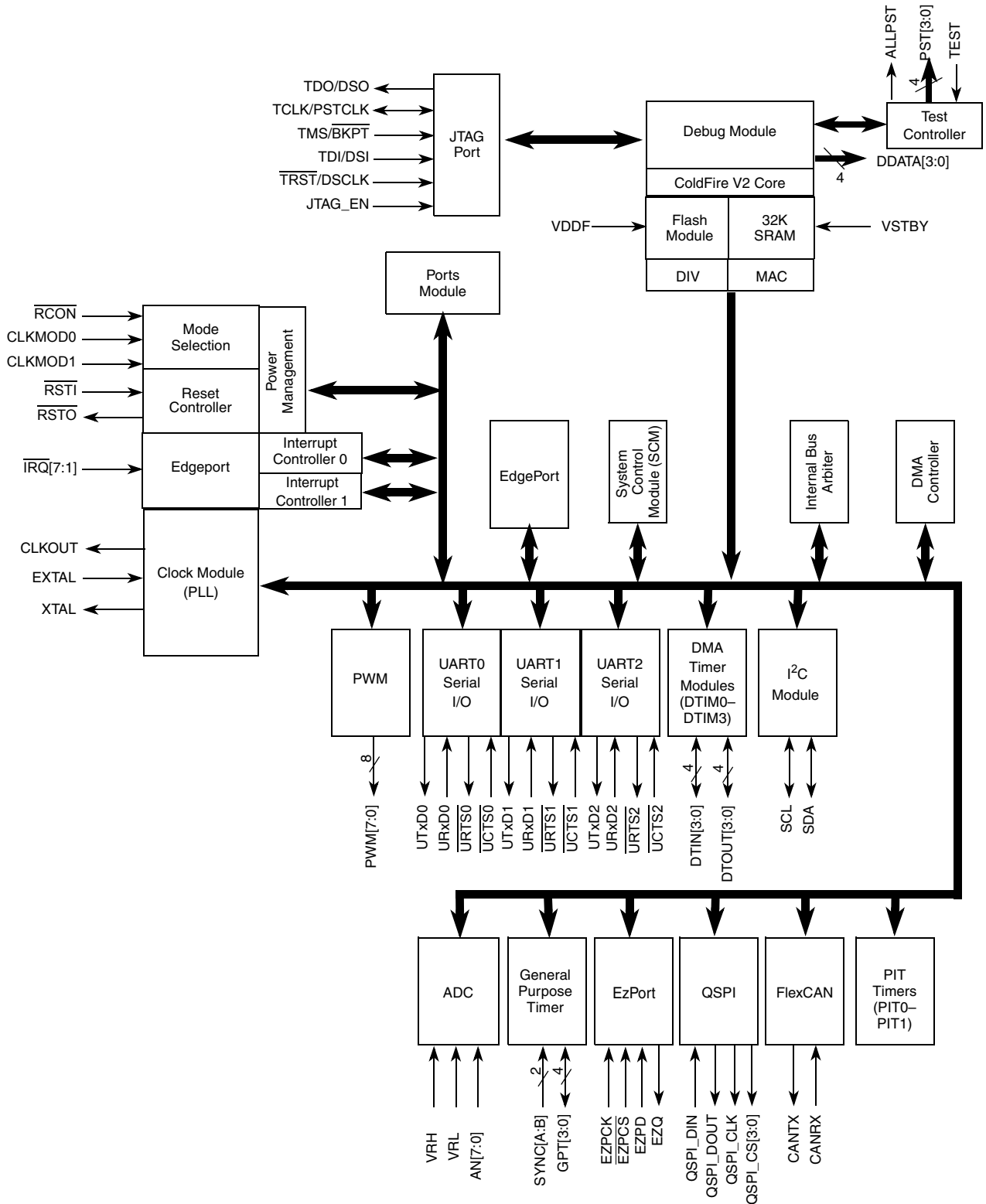
Figure 2-1. Block Diagram with Signal Interfaces

Preliminary

Table 2-1 shows the pin functions by primary and alternate purpose, and illustrates which packages contain each pin.

Table 2-1. Pin Functions by Primary and Alternate Purpose and Package

Pin Group	Primary Function	Secondary Function	Tertiary Function	Quaternary Function	Drive Strength / Control	Slew Rate / Control	Pull-up / Pull-down	Pin on 100 LQFP	Pin on 81 MAPBGA	Pin on 64 LQFP	Notes
ADC	AN7	—	—	GPIO	2mA	FAST	—	51	H9	33	
	AN6	—	—	GPIO	2mA	FAST	—	52	G9	34	
	AN5	—	—	GPIO	2mA	FAST	—	53	G8	35	
	AN4	—	—	GPIO	2mA	FAST	—	54	F9	39	
	AN3	—	—	GPIO	2mA	FAST	—	46	G7	28	
	AN2	—	—	GPIO	2mA	FAST	—	45	G6	27	
	AN1	—	—	GPIO	2mA	FAST	—	44	H6	26	
	AN0	—	—	GPIO	2mA	FAST	—	43	J6	25	
	SYNCA	—	—	—	N/A	N/A	—	—	—	—	No Primary
	SYNCB	—	—	—	N/A	N/A	—	—	—	—	No Primary
	VDDA	—	—	—	N/A	N/A	—	50	H8	32	
	VSSA	—	—	—	N/A	N/A	—	47	H7	39	
	VRH	—	—	—	N/A	N/A	—	49	J8	31	
	VRL	—	—	—	N/A	N/A	—	48	J7	30	
Clock Generation	EXTAL	—	—	—	N/A	N/A	—	73	B9	47	
	XTAL	—	—	—	N/A	N/A	—	72	C9	46	
	VDDPLL	—	—	—	N/A	N/A	—	74	B8	48	
	VSSPLL	—	—	—	N/A	N/A	—	71	C8	45	
Debug Data	ALLPST	—	—	—	10mA	FAST	—	86	A6	55	
	DDATA[3:0]	—	—	GPIO	10mA	FAST	—	84,83,78,77	—	—	
	PST[3:0]	—	—	GPIO	10mA	FAST	—	70,69,66,65	—	—	

MCF5213 Reference Manual, Rev. 1.1

Preliminary

Table 2-1. Pin Functions by Primary and Alternate Purpose and Package

Pin Group	Primary Function	Secondary Function	Tertiary Function	Quaternary Function	Drive Strength / Control	Slew Rate / Control	Pull-up / Pull-down	Pin on 100 LQFP	Pin on 81 MAPBGA	Pin on 64 LQFP	Notes
I²C	SCL	CANTX[1]	UTXD2	GPIO	PDSR[0]	PSRR[0]	—	10	E1	8	
	SDA	CANRX[1]	URXD2	GPIO	PDSR[0]	PSRR[0]	—	11	E2	9	
Interrupts	IRQ7	—	—	GPIO	2 mA	FAST	—	95	C4	58	
	IRQ6	—	—	GPIO	2 mA	FAST	—	94	B4	—	
	IRQ5	—	—	GPIO	2 mA	FAST	—	91	A4	—	
	IRQ4	—	—	GPIO	2 mA	FAST	—	90	C5	57	
	IRQ3	—	—	GPIO	2 mA	FAST	—	89	A5	—	
	IRQ2	—	—	GPIO	2 mA	FAST	—	88	B5	—	
	IRQ1	SYNCA	PWM1	GPIO	10 mA	FAST	—	87	C6	56	
JTAG/BDM	JTAG_EN	—	—	—	N/A	N/A	pull-down	26	J2	17	
	TCLK/ PSTCLK	CLKOUT	—	—	10 mA	FAST	—	64	C7	44	
	TDI/DSI	—	—	—	N/A	N/A	—	79	B7	50	
	TDO/DSO	—	—	—	10 mA	FAST	—	80	B7	50	
	TMS /BKPT	—	—	—	N/A	N/A	—	76	A8	49	
	TRST /DSCLK	—	—	—	N/A	N/A	—	85	B6	54	
Mode Selection[2]	CLKMOD0	—	—	—	N/A	N/A	pull-down[2]	40	G5	24	
	CLKMOD1	—	—	—	N/A	N/A	pull-down[2]	39	H5	—	
	RCON/ EZPCS	—	—	—	N/A	N/A	pull-up	21	G3	16	
PWM	PWM7	—	—	GPIO	PDSR[31]	PSRR[31]	—	63	D7	—	
	PWM5	—	—	GPIO	PDSR[30]	PSRR[30]	—	60	E8	—	
	PWM3	—	—	GPIO	PDSR[29]	PSRR[29]	—	33	J4	—	
	PWM1	—	—	GPIO	PDSR[28]	PSRR[28]	—	38	J5	—	

MCF5213 Reference Manual, Rev. 1.1

Preliminary

Table 2-1. Pin Functions by Primary and Alternate Purpose and Package

Pin Group	Primary Function	Secondary Function	Tertiary Function	Quaternary Function	Drive Strength / Control	Slew Rate / Control	Pull-up / Pull-down	Pin on 100 LQFP	Pin on 81 MAPBGA	Pin on 64 LQFP	Notes
QSPI	QSPI_DIN/ EZPD	CANRX[1]	RXD1	GPIO	PDSR[2]	PSRR[2]	—	16	F3	12	
	QSPI_DOUT /EZPQ	CANTX[1]	TXD1	GPIO	PDSR[1]	PSRR[1]	—	17	G1	13	
	QSPI_SCK/ EZPCK	SCL	RTS1	GPIO	PDSR[3]	PSRR[3]	—	18	G2	14	
	QSPI_CS3	SYNCA	SYNCB	GPIO	PDSR[7]	PSRR[7]	—	12	F1	—	
	QSPI_CS2	—	—	GPIO	PDSR[6]	PSRR[6]	—	13	F2	—	
	QSPI_CS1	—	—	GPIO	PDSR[5]	PSRR[5]	—	19	H2	—	
	QSPI_CS0	SDA	CTS1	GPIO	PDSR[4]	PSRR[4]	—	20	H1	15	
Reset[3]	RSTI	—	—	—	N/A	N/A	pull-up[3]	96	A3	59	
	RSTO	—	—	—	10 mA	FAST	—	97	B3	60	
Test	TEST	—	—	—	N/A	N/A	pull-down	5	C2	3	
Timers, 16-bit	GPT3	—	PWM7	GPIO	PDSR[23]	PSRR[23]	pull-up[4]	62	D8	43	
	GPT2	—	PWM5	GPIO	PDSR[22]	PSRR[22]	pull-up[4]	61	D9	42	
	GPT1	—	PWM3	GPIO	PDSR[21]	PSRR[21]	pull-up[4]	59	E9	41	
	GPT0	—	PWM1	GPIO	PDSR[20]	PSRR[20]	pull-up[4]	58	F7	40	
Timers, 32-bit	DTIN3	DTOUT3	PWM6	GPIO	PDSR[19]	PSRR[19]	—	32	H3	19	
	DTIN2	DTOUT2	PWM4	GPIO	PDSR[18]	PSRR[18]	—	31	J3	18	
	DTIN1	DTOUT1	PWM2	GPIO	PDSR[17]	PSRR[17]	—	37	G4	23	
	DTIN0	DTOUT0	PWM0	GPIO	PDSR[16]	PSRR[16]	—	36	H4	22	
UART 0	UCTS0	CANRX	—	GPIO	PDSR[11]	PSRR[11]	—	6	C1	4	
	URTS0	CANTX	—	GPIO	PDSR[10]	PSRR[10]	—	9	D3	7	
	URXD0	—	—	GPIO	PDSR[9]	PSRR[9]	—	7	D1	5	
	UTXD0	—	—	GPIO	PDSR[8]	PSRR[8]	—	8	D2	6	

MCF5213 Reference Manual, Rev. 1.1

Preliminary

Table 2-1. Pin Functions by Primary and Alternate Purpose and Package

Pin Group	Primary Function	Secondary Function	Tertiary Function	Quaternary Function	Drive Strength / Control	Slew Rate / Control	Pull-up / Pull-down	Pin on 100 LQFP	Pin on 81 MAPBGA	Pin on 64 LQFP	Notes
UART 1	UCTS1	SYNCA	URXD2	GPIO	PDSR[15]	PSRR[15]	—	98	C3	61	
	URTS1	SYNCB	UTXD2	GPIO	PDSR[14]	PSRR[14]	—	4	B1	2	
	URXD1	—	—	GPIO	PDSR[13]	PSRR[13]	—	100	B2	63	
	UTXD1	—	—	GPIO	PDSR[12]	PSRR[12]	—	99	A2	62	
UART 2	UCTS2	—	—	GPIO	PDSR[27]	PSRR[27]	—	27	—	—	
	URTS2	—	—	GPIO	PDSR[26]	PSRR[26]	—	30	—	—	
	URXD2	—	—	GPIO	PDSR[25]	PSRR[25]	—	28	—	—	
	UTXD2	—	—	GPIO	PDSR[24]	PSRR[24]	—	29	—	—	
FlexCAN	CANRX				N/A	N/A	—	—	—	—	See Note [1,5]
	CANTX				N/A	N/A	—	—	—	—	See Note [1,5]
VSTBY	VSTBY	—	—	—	N/A	N/A	—	55	F8	37	
VDD	VDD	—	—	—	N/A	N/A	—	8	6	5	
VSS	VSS	—	—	—	N/A	N/A	—	8	6	5	

[1] The multiplexed CANTX and CANRX signals are not available on the MCF5211 or MCF5212
[2] CLKMOD0 and CLKMOD1 have internal pull-down resistors, however the use of external resistors is very strongly recommended
[3] RSTI has an internal pull-up resistor, however the use of an external resistor is very strongly recommended
[4] For GPIO function. Primary Function has pull-up control within the GPT module
[5] CANTX and CANRX are secondary functions only.

2.2 Reset Signals

describes signals that are used to either reset the chip or as a reset indication.

MCF5213 Reference Manual, Rev. 1.1

Preliminary

Table 2-2. Reset Signals

Signal Name	Abbreviation	Function	I/O
Reset In	$\overline{\text{RSTI}}$	Primary reset input to the device. Asserting $\overline{\text{RSTI}}$ immediately resets the CPU and peripherals.	I
Reset Out	$\overline{\text{RSTO}}$	Driven low for 512 CPU clocks after the reset source has deasserted and PLL locked.	O

Preliminary

2.3 PLL and Clock Signals

Table 2-3 describes signals that are used to support the on-chip clock generation circuitry.

Table 2-3. PLL and Clock Signals

Signal Name	Abbreviation	Function	I/O
External Clock In	EXTAL	Crystal oscillator or external clock input except when the on-chip relaxation oscillator is used.	I
Crystal	XTAL	Crystal oscillator output except when CLKMOD1=1, then sampled as part of the clockmode selection mechanism.	O
Clock Out	CLKOUT	This output signal reflects the internal system clock.	O

2.4 Mode Selection

Table 2-4 describes signals used in mode selection, Table 2-5 describes particular clocking modes.

Table 2-4. Mode Selection Signals

Signal Name	Abbreviation	Function	I/O
Clock Mode Selection	CLKMOD[1:0]	Selects the clock boot mode.	I
Reset Configuration	$\overline{\text{RCON}}$	The Serial Flash Programming mode is entered by asserting the $\overline{\text{RCON}}$ pin (with the TEST pin negated) as the chip comes out of reset. During this mode, the EzPort has access to the Flash memory which can be programmed from an external device.	
Test	TEST	Reserved for factory testing only and in normal modes of operation should be connected to VSS to prevent unintentional activation of test functions.	I

Table 2-5. Clocking Modes

CLKMOD[1:0]	XTAL	Configure the Clock Mode
00	0	PLL disabled, clock driven by external oscillator
00	1	PLL disabled, clock driven by on-chip oscillator
01	N/A	PLL disabled, clock driven by crystal
10	0	PLL in normal mode, clock driven by external oscillator
10	1	PLL in normal mode, clock driven by on-chip oscillator
11	N/A	PLL in normal mode, clock driven by crystal

2.5 External Interrupt Signals

Table 2-6 describes the external interrupt signals.

Table 2-6. External Interrupt Signals

Signal Name	Abbreviation	Function	I/O
External Interrupts	$\overline{\text{IRQ}}$[7:1]	External interrupt sources.	I

2.6 Queued Serial Peripheral Interface (QSPI)

Table 2-7 describes QSPI signals.

Table 2-7. Queued Serial Peripheral Interface (QSPI) Signals

Signal Name	Abbreviation	Function	I/O
QSPI Synchronous Serial Output	QSPI_DOUT	Provides the serial data from the QSPI and can be programmed to be driven on the rising or falling edge of QSPI_CLK.	O
QSPI Synchronous Serial Data Input	QSPI_DIN	Provides the serial data to the QSPI and can be programmed to be sampled on the rising or falling edge of QSPI_CLK.	I
QSPI Serial Clock	QSPI_CLK	Provides the serial clock from the QSPI. The polarity and phase of QSPI_CLK are programmable.	O
Synchronous Peripheral Chip Selects	QSPI_CS[3:0]	QSPI peripheral chip selects that can be programmed to be active high or low.	O

2.7 I^{2}C I/O Signals

Table 2-8 describes the I^{2}C serial interface module signals.

Table 2-8. I^{2}C I/O Signals

Signal Name	Abbreviation	Function	I/O
Serial Clock	SCL	Open-drain clock signal for the for the I^{2}C interface. Either it is driven by the I^{2}C module when the bus is in master mode or it becomes the clock input when the I^{2}C is in slave mode.	I/O
Serial Data	SDA	Open-drain signal that serves as the data input/output for the I^{2}C interface.	I/O

2.8 UART Module Signals

Table 2-9 describes the UART module signals.

Table 2-9. UART Module Signals

Signal Name	Abbreviation	Function	I/O
Transmit Serial Data Output	UTXD*n*	Transmitter serial data outputs for the UART modules. The output is held high (mark condition) when the transmitter is disabled, idle, or in the local loopback mode. Data is shifted out, LSB first, on this pin at the falling edge of the serial clock source.	O
Receive Serial Data Input	URXD*n*	Receiver serial data inputs for the UART modules. Data is received on this pin LSB first. When the UART clock is stopped for power-down mode, any transition on this pin restarts it.	I
Clear-to-Send	$\overline{\text{UCTS}}$*n*	Indicate to the UART modules that they can begin data transmission.	I
Request-to-Send	$\overline{\text{URTS}}$*n*	Automatic request-to-send outputs from the UART modules. This signal can also be configured to be asserted and negated as a function of the RxFIFO level.	O

2.9 DMA Timer Signals

Table 2-10 describes the signals of the four DMA timer modules.

Table 2-10. DMA Timer Signals

Signal Name	Abbreviation	Function	I/O
DMA Timer Input	DTIN	Event input to the DMA timer modules.	I
DMA Timer Output	DTOUT	Programmable output from the DMA timer modules.	O

2.10 ADC Signals

Table 2-11 describes the signals of the analog-to-digital converter.

Table 2-11. ADC Signals

Signal Name	Abbreviation	Function	I/O
Analog Inputs	AN[7:0]	Inputs to the A-to-D converter.	I
Analog Reference	V_{RH}	Reference voltage high and low inputs.	I
	V_{RL}		I
Analog Supply	V_{DDA}	Isolate the ADC circuitry from power supply noise	—
	V_{SSA}		—

2.11 General Purpose Timer Signals

Table 2-12 describes the General Purpose Timer Signals.

Table 2-12. GPT Signals

Signal Name	Abbreviation	Function	I/O
General Purpose Timer Input/Output	GPT[3:0]	Inputs to or outputs from the general purpose timer module	I/O

2.12 Pulse Width Modulator Signals

Table 2-13 describes the PWM signals.

Table 2-13. PWM Signals

Signal Name	Abbreviation	Function	I/O
PWM Output Channels	PWM[7:0]	Pulse width modulated output for PWM channels	O

2.13 Debug Support Signals

The signals in Table 2-14 are used as the interface to the on-chip JTAG controller and also to interface to the BDM logic.

Table 2-14. Debug Support Signals

Signal Name	Abbreviation	Function	I/O
JTAG Enable	JTAG_EN	Select between debug module and JTAG signals at reset	I
Test Reset	$\overline{\text{TRST}}$	This active-low signal is used to initialize the JTAG logic asynchronously.	I
Test Clock	TCLK	Used to synchronize the JTAG logic.	I
Test Mode Select	TMS	Used to sequence the JTAG state machine. TMS is sampled on the rising edge of TCLK.	I
Test Data Input	TDI	Serial input for test instructions and data. TDI is sampled on the rising edge of TCLK.	I
Test Data Output	TDO	Serial output for test instructions and data. TDO is tri-stateable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCLK.	O
Development Serial Clock	DSCLK	Development Serial Clock-Internally synchronized input. (The logic level on DSCLK is validated if it has the same value on two consecutive rising bus clock edges.) Clocks the serial communication port to the debug module during packet transfers. Maximum frequency is PSTCLK/5. At the synchronized rising edge of DSCLK, the data input on DSI is sampled and DSO changes state.	I

Table 2-14. Debug Support Signals (continued)

Signal Name	Abbreviation	Function	I/O
Breakpoint	$\overline{BKPT}$	Breakpoint - Input used to request a manual breakpoint. Assertion of $\overline{BKPT}$ puts the processor into a halted state after the current instruction completes. Halt status is reflected on processor status/debug data signals (PSTDDATA[7:0]) as the value 0xF. If CSR[BKD] is set (disabling normal $\overline{BKPT}$ functionality), asserting $\overline{BKPT}$ generates a debug interrupt exception in the processor.	I
Development Serial Input	DSI	Development Serial Input -Internally synchronized input that provides data input for the serial communication port to the debug module, once the DSCLK has been seen as high (logic 1).	I
Development Serial Output	DSO	Development Serial Output -Provides serial output communication for debug module responses. DSO is registered internally. The output is delayed from the validation of DSCLK high.	O
Debug Data	DDATA[3:0]	Display captured processor data and breakpoint status. The CLKOUT signal can be used by the development system to know when to sample DDATA[3:0].	O
Processor Status Clock	PSTCLK	Processor Status Clock - Delayed version of the processor clock. Its rising edge appears in the center of valid PST and DDATA output. PSTCLK indicates when the development system should sample PST and DDATA values. If real-time trace is not used, setting CSR[PCD] keeps PSTCLK, and PST and DDATA outputs from toggling without disabling triggers. Non-quiescent operation can be reenabled by clearing CSR[PCD], although the external development systems must resynchronize with the PST and DDATA outputs. PSTCLK starts clocking only when the first non-zero PST value (0xC, 0xD, or 0xF) occurs during system reset exception processing.	O
Processor Status Outputs	PST[3:0]	Indicate core status. Debug mode timing is synchronous with the processor clock; status is unrelated to the current bus transfer. The CLKOUT signal can be used by the development system to know when to sample PST[3:0].	O
All Processor Status Outputs	ALLPST	Logical "AND" of PST[3.0]	O

2.14 EzPort Signal Descriptions

Table 2-15 contains a list of EzPort external signals

Table 2-15. EzPort Signal Descriptions

Signal Name	Abbreviation	Function	I/O
EzPort Clock	EZPCK	Shift clock for EzPort transfers	I
EzPort Chip Select	$\overline{EZPCS}$	Chip select for signalling the start and end of serial transfers	I
EzPort Serial Data In	EZPD	EZPD is sampled on the rising edge of EZPCK	I
EzPort Serial Data Out	EZPQ	EZPQ transitions on the falling edge of EZPCK	O

2.15 Power and Ground Pins

The pins described in Table 2-16 provide system power and ground to the chip. Multiple pins are provided for adequate current capability. All power supply pins must have adequate decoupling (bypass capacitance) for high-frequency noise suppression.

Table 2-16. Power and Ground Pins

Signal Name	Abbreviation	Function	I/O
PLL Analog Supply	VDDPLL, VSSPLL	Dedicated power supply signals to isolate the sensitive PLL analog circuitry from the normal levels of noise present on the digital power supply.	I
Positive Supply	VDD	These pins supply positive power to the core logic.	I
Ground	VSS	This pin is the negative supply (ground) to the chip.	

Chapter 3
ColdFire Core

This section describes the organization of the Version ColdFire® processor core and an overview of the program-visible registers. For detailed information on instructions, see the ISA_A+ definition in the *ColdFire Family Programmer's Reference Manual*.

3.1   Processor Pipelines

Figure 3-2 is a block diagram showing the processor pipelines of a CF ColdFire version core.
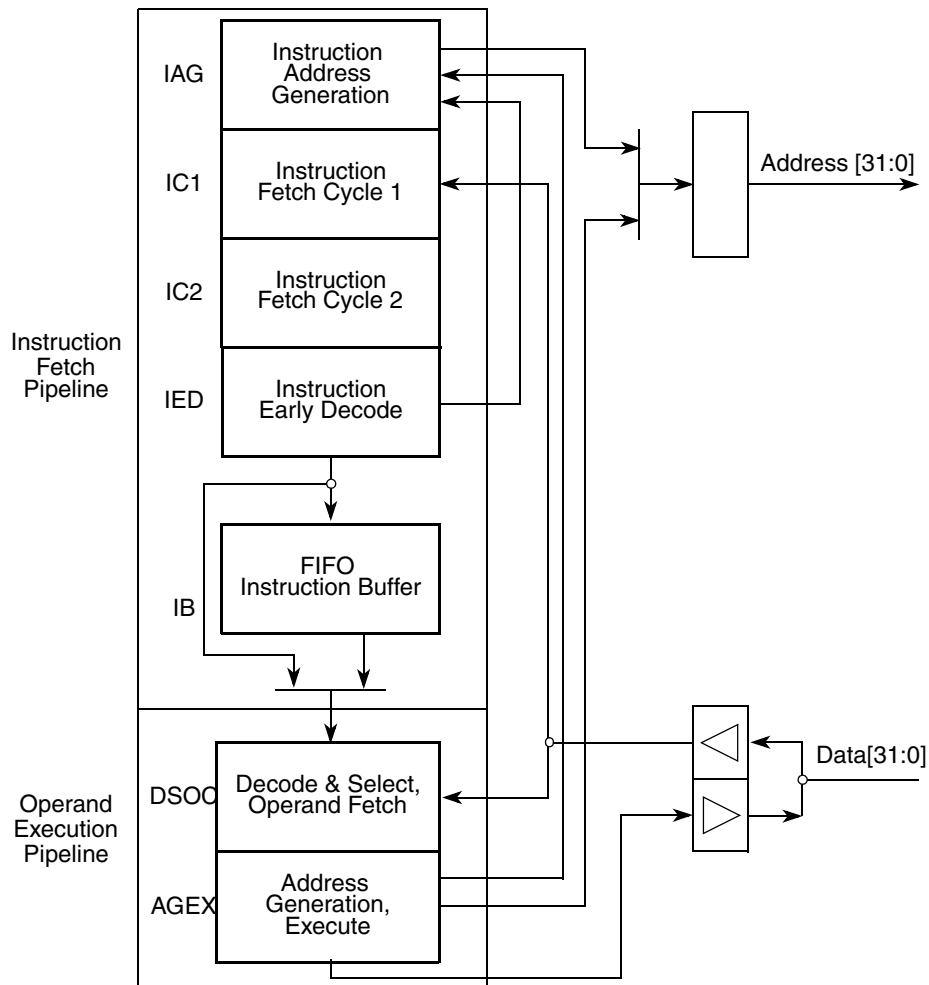
As with all ColdFire cores, the CF processor core is comprised of two separate pipelines that are decoupled by an instruction buffer.

The Instruction Fetch Pipeline (IFP) is a -stage pipeline for prefetching instructions. The prefetched instruction stream is then gated into the two-stage Operand Execution Pipeline (OEP), which decodes the instruction, fetches the required operands and then executes the required function. Since the IFP and OEP pipelines are decoupled by an instruction buffer which serves 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 CF pipeline stages include the following:

- Instruction fetch pipeline (IFP)
    - Instruction address generation (IAG)—Calculates the next prefetch address
    - Instruction fetch cycle 1 (IC1)—Initiates prefetch on the processor's local bus
    - Instruction fetch cycle 2 (IC2)—Completes prefetch on the processor's local bus
    - Instruction early decode (IED)—Generates time-critical decode signals needed for the OEP
    - Instruction buffer (IB)—Optional buffer stage minimizes effects of fetch latency using FIFO queue
- 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

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 allows the CF32 processor core to deliver sustained high performance across a variety of demanding embedded applications. See Table 1-1 for details on the performance of this device.

3.2   Processor Register Description

The following sections describe the processor registers in the user and supervisor programming models. The appropriate programming model is selected based on the privilege level (user mode or supervisor mode) of the processor as defined by the S bit of the status register (SR). Table 3-3 lists the processor registers.

3.2.1   User Programming Model

Table 3-1 illustrates the user programming model. The user programming model is the same as the M68000 family microprocessors, consisting 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)
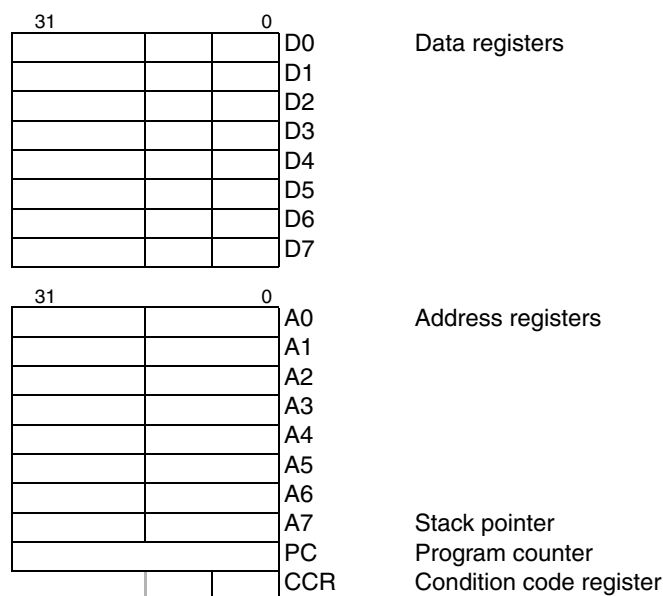
Figure 3-1. ColdFire Family User Programming Model

3.2.2   Data Registers (D0–D7)

Registers D0–D7 are used as data registers 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 3.7.14, "Reset Exception" for more details.

3.2.3   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.

3.2.4   Stack Pointers (A7)

This ColdFire implementationsupports two unique stack pointer (A7) registers—the supervisor stack pointer (SSP) and the user stack pointer (USP). This support provides the required isolation between operating modes of the processor. The SSP is described in Section 3.2.8.2, "Supervisor/User Stack Pointers (A7 and OTHER_A7)."

A subroutine call saves the PC on the stack and the return restores it from the stack. Both the PC and the SR are saved on the supervisor stack during the processing of exceptions and interrupts. The return from exception (RTE) instruction restores the SR and PC values from the supervisor stack.

3.2.5 Program Counter (PC)

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

3.2.6 Condition Code Register (CCR)

The CCR is the LSB of the processor status register (SR). The branch prediction bit, bit 7, provides a mechanism to alter the static algorithm used by the branch acceleration logic on the IFP. Bits 4–0 act as indicator flags for results generated by processor operations. The extend bit (X) is also used as an input operand during multiprecision arithmetic computations.

	7	6	5	4	3	2	1	0
Field	0	0	0	X	N	Z	V	C
Reset	0^1	0^1	0^1	—	—	—	—	—
Address	LSB of Status Register (SR)							

Figure 3-2. Condition Code Register (CCR)

[1] Read-only

Table 3-1. CCR Field Descriptions

Bits	Field	Description
7	P	Branch prediction bit. Alters the static prediction algorithm used by the branch acceleration logic in the IFP on forward conditional branches. 0 Forward conditional branch instructions are predicted as not taken. 1 Forward conditional branch instructions are predicted as taken. **Note:** All backward Bcc instructions are predicted as taken, regardless of this bit.
6-5	—	Reserved, should be cleared.
4	X	Extend condition code bit. Set to the value of the C-bit for arithmetic operations; otherwise not affected or set to a specified result.
3	N	Negative condition code bit. Set if the most significant bit of the result is set; otherwise cleared.
2	Z	Zero condition code bit. Set if the result equals zero; otherwise cleared.
1	V	Overflow condition code bit. Set if an arithmetic overflow occurs implying that the result cannot be represented in the 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

3.2.7 MAC Register Description

The registers in the MAC portion of the user programming model, are described in Chapter 4, "Enhanced Multiply-Accumulate Unit (EMAC)," and include the following registers:

- A 32-bit accumulator (ACC)
- A 16-bit mask register (MASK)
- A 32-bit status register (MACSR)

These registers are shown in Table 3-2.

Table 3-2. MAC Register Set

31:24	23:16	15:8	7:0	Mnemonic
MAC Status Register				MACSR
MAC Accumulator 0				ACC0
MAC Accumulator 1				ACC1
MAC Accumulator 2				ACC2
MAC Accumulator 3				ACC3
Extensions for ACC0 and ACC1				ACCext01
Extensions for ACC2 and ACC3				ACCext23
MAC Mask Register				MASK

3.2.8 Supervisor Register Description

Only system control software is intended to use the supervisor programming model 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, which 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 cache control register (CACR)
- Two 32-bit access control registers (ACR0, ACR1)
- Two 32-bit memory base address register (RAMBAR, FLASHBAR)

The supervisor programming model consists of the registers available to users as well as the registers listed in Figure 3-3.

31	19	15	0		
		(CCR)		SR	Status register
				OTHER_A7	Supervisor A7 stack pointer
	Must be zeros			VBR	Vector base register
				CACR	Cache control register
				ASID	Address space ID register
				ACR0	Access control register 0 (data)
				ACR1	Access control register 1 (data)
				ACR2	Access control register 2 (instruction)
				ACR3	Access control register 3 (instruction)
				MMUBAR	MMU base address register
				RAMBAR0	RAM base address register 0
				RAMBAR1	RAM base address register 1
				FLASHBAR	FLASH base address register
				IPSBAR	Module base address register

Figure 3-3. Supervisor Programming Model

The following paragraphs describe the supervisor programming model registers.

3.2.8.1   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 are accessible (CCR). 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. SR is set to 0x27 after reset. The SR register must be explicitly loaded after reset and before any compare, Bcc or Scc instructions are executed.

			System Byte							**Condition Code Register (CCR)**					
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	T	—	S	M	—		I			——————		X	N	Z	V	C
Reset	0	0	0	0	0	0	0	0	0^1	0^1	0^1	—	—	—	—	—
Address						CPU @ 0x80E										
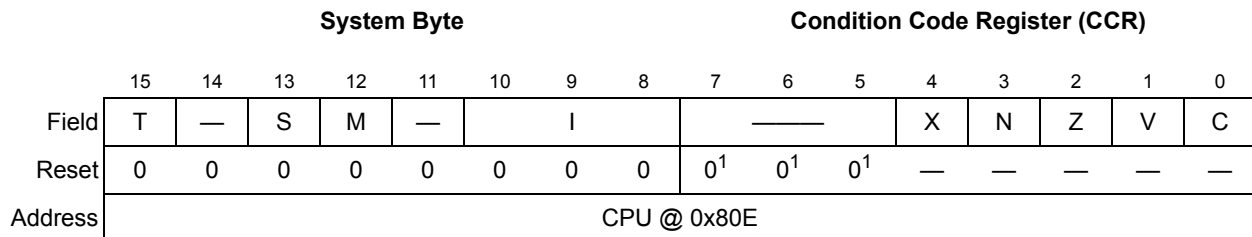
Figure 3-4. Status Register (SR)

[1] Read-only

Table 3-3. SR Field Descriptions

Bits	Field	Description
15	T	Trace enable. When set, the processor performs a trace exception after every instruction.
14	—	Reserved, should be cleared.
13	S	Supervisor/user state. Denotes whether the processor is in supervisor mode (S = 1) or user mode (S = 0).
12	M	Master/interrupt state. This bit is cleared by an interrupt exception, and can be set by software during execution of the RTE or move to SR instructions.
11	—	Reserved, should be cleared.

Table 3-3. SR Field Descriptions (continued)

Bits	Field	Description
10-8	I	Interrupt level mask. Defines the current interrupt level. Interrupt requests are inhibited for all priority levels less than or equal to the current level, except the edge-sensitive level 7 request, which cannot be masked.
7-0	CCR	Refer to Table 3-1.

3.2.8.2 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 programmable-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*.

3.2.8.3 Vector Base Register (VBR)

The VBR contains the base address of the exception vector table in 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 MByte boundary.

3.2.8.4 Cache Control Register (CACR)

The CACR controls operation of the instruction/data cache memories. It includes bits for enabling, freezing, and invalidating cache contents. It also includes bits for defining the default cache mode and write-protect fields. The CACR is described in Section 5.2.1.1, "Cache Control Register (CACR)."

3.2.8.5 Access Control Registers (ACR0, ACR1)

The access control registers, ACR0 and ACR1, define attributes for two user-defined memory regions. These attributes include the definition of cache mode, write protect, and buffer write enables. The ACRs are described in Section 5.2.1.2, "Access Control Registers (ACR0, ACR1)."

3.2.8.6 Memory Base Address Register (RAMBAR, FLASHBAR)

Memory base address registers are used to specify the base address of the internal SRAM and Flash modules and indicate the types of references mapped to each. Each base address register includes a base address, write-protect bit, address space mask bits, and an enable bit. For the MCF5213, FLASHBAR determines the base address of the on-chip Flash, and RAMBAR determines the base address of the on-chip RAM. For more information, refer to Section 5.2.1, "SRAM Base Address Register (RAMBAR)."

3.3 Memory Map/Register Definition

Table 3-4 lists register names, the CPU space location, and whether the register is written from the processor using the MOVEC instruction.

Table 3-4. ColdFire CPU Registers

Name	CPU Space (Rc)	Written with MOVEC		Register Name
Memory Management Control Registers				
CACR	0x002	Yes	0x0000_0000	Cache control register
ACR0, ACR1	0x004–0x005	Yes	Undefined	Access control registers 0 and 1
Processor General-Purpose Registers				
Processor Miscellaneous Registers				
OTHER_A7	0x800	No	Undefined	Other stack pointer
VBR	0x801	Yes	0x0000_0000	Vector base register
MACSR	0x804	No	0x0000_0000	MAC status register
MASK	0x805	No	0xFFFF_FFFF	MAC address mask register
ACC0–ACC3	0x806, 0x809, 0x80A, 0x80B	No	Undefined	MAC accumulators 0-3
ACCext01	0x807	No	Undefined	MAC accumulator 0, 1 extension bytes
ACCext23	0x808	No	Undefined	MAC accumulator 2, 3 extension bytes
SR	0x80E	No	0x27—	Status register
PC	0x80F	Yes	Undefined	Program counter
Local Memory Registers				
FLASHBAR	0xC04	Yes	0x0000_0000	Flash base address register
RAMBAR	0xC05	Yes	0x0000_0000	SRAM base address register

3.4   Additions to the Instruction Set Architecture

The original ColdFire instruction set architecture (ISA) was derived from the M68000-family opcodes based on extensive analysis of embedded application code. After the initial ColdFire compilers were created, developers identified ISA additions that would enhance both code density and overall performance. Additionally, as users implemented ColdFire-based designs into a wide range of embedded systems, they identified frequently used instruction sequences that could be improved by the creation of new instructions. This observation was especially prevalent in development environments that made use of substantial amounts of assembly language code.

Table 3-5 summarizes the new instructions added to the Revision A+ ISA. For more details see the *ColdFire Family Programmer's Reference Manual*.

Table 3-5. ISA Revision A+ New Instructions

Instruction	Description
BITREV	The contents of the destination data register are bit-reversed; that is, new Dn[31] = old Dn[0], new Dn[30] = old Dn[1], ..., new Dn[0] = old Dn[31].
BYTEREV	The contents of the destination data register are byte-reversed; that is, new Dn[31:24] = old Dn[7:0], ..., new Dn[7:0] = 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.
MOVE FROM USP	USP $\rightarrow$ Destination
MOVE TO USP	Source $\rightarrow$ USP
STLDSR	Pushes the contents of the status register onto the stack and then reloads the status register with the immediate data value.

3.5   Exception Processing Overview

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

- A simplified exception vector table
- Reduced relocation capabilities using the vector base register
- A single exception stack frame format
- Use of a single self-aligning stack pointer (for ISA_A implementations only)

All ColdFire processors use an instruction restart exception model, but certain microarchitectures (CF2 and CF3) require more software support to recover from certain access errors. See Section 3.7.1, "Access Error Exception" for details.

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

First, the processor makes an internal copy of the SR and then enters supervisor mode by asserting the S bit and disabling trace mode by negating the T bit. The occurrence of an interrupt exception also forces the M bit to be cleared and the interrupt priority mask to be set to the level of the current interrupt request.

Second, 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 interrupt-acknowledge (IACK) bus cycle to obtain the vector number from the interrupt controller. The IACK cycle is mapped to a special acknowledge address space with the interrupt level encoded in the address.

Third, the processor saves the current context by creating an exception stack frame on the system stack. Processors implementing ISA_A support a single stack pointer in the A7 address register; therefore, there is not notion of separate supervisor and user stack pointer. As a result, the exception stack frame is created at a 0-modulo-4 address on top of the current system stack. For processors implementing all other ISA revisions and supporting 2 stack pointers, the exception stack frame is created at a 0-modulo-4 address on top of the system stack defined by the Supervisor Stack Pointer (SSP). Additionally, 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).

Fourth, the processor calculates the address of the first instruction of the exception handler. By definition, the exception vector table is aligned on a 1 Mbyte 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 x vector number). Once the exception vector has been fetched, the contents of the vector determine 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 Mbyte address boundary (see Table 3-6). The table contains 256 exception vectors; the first 64 are defined by Freescale and the remaining 192 are user-defined interrupt vectors.

Table 3-6. Exception Vector Assignments

Vector Number(s)	Vector Offset (Hex)	Stacked Program Counter	Assignment
0	0x000	—	Initial stack pointer
1	0x004	—	Initial program counter
2	0x008	Fault	Access error
3	0x00C	Fault	Address error
4	0x010	Fault	Illegal instruction
5	0x014	Fault	Divide by zero
6–7	0x018–0x01C	—	Reserved
8	0x020	Fault	Privilege violation
9	0x024	Next	Trace

Table 3-6. Exception Vector Assignments (continued)

Vector Number(s)	Vector Offset (Hex)	Stacked Program Counter	Assignment
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
48–63	0x0C0–0x0FC	—	Reserved
64–255	0x100–0x3FC	Next	User-defined interrupts

"Fault" refers to the PC of the instruction that caused the exception; "Next" refers to the PC of the next 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 effectively disable interrupts, if necessary, by raising the interrupt mask level contained in the status register. In addition, the ISA_A+ 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 which services multiple interrupt requests with different interrupt levels. For more details see the *ColdFire Family Programmer's Reference Manual*.

3.6 Exception Stack Frame Definition

The exception stack frame is shown in Figure 3-5. The first longword of the exception stack frame 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.
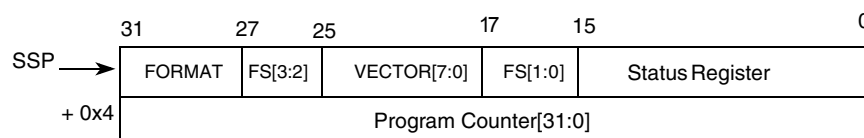
Figure 3-5. Exception Stack Frame Form

The 16-bit format/vector word contains 3 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 3-7.

MCF5213 Reference Manual, Rev. 1.1

Table 3-7. Format Field Encodings

Original SSP @ Time of Exception, Bits 1:0	SSP @ 1st Instruction of Handler	Format Field
00	Original SSP - 8	4
01	Original SSP - 9	5
10	Original SSP - 10	6
11	Original SSP - 11	7

- 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 types of exceptions. See Table 3-8.

Table 3-8. Fault Status Encodings

FS[3:0]	Definition
00xx	Reserved
0100	Error on instruction fetch
0101	Reserved
011x	Reserved
1000	Error on operand write
1001	Attempted write to write-protected space
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 the case of an interrupt. Refer to Table 3-6.

3.7   Processor Exceptions

3.7.1   Access Error Exception

The exact processor response to an access error depends on the type of 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 that occur during instruction prefetches that are then 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 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 the execution of a MOVEM instruction loading from memory, any registers already updated before the fault occurs contain the operands from memory.

The CF 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.

3.7.2   Address Error Exception

Any attempted execution transferring control to an odd instruction address (that is, 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 8 on an indexed effective addressing mode generates an address error as does an attempted execution of a full-format indexed addressing mode.

3.7.3   Illegal Instruction Exception

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 opcode generates 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.

3.7.4   Divide-By-Zero

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

3.7.5   Privilege Violation

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

3.7.6 Trace Exception

To aid in program development, all ColdFire processors provide an instruction-by-instruction tracing capability. While in trace mode, indicated by setting of the T bit in the status register (SR[15] = 1), 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 is executed, 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 the execution of a TRAP instruction while in trace mode. The processor will initiate the TRAP exception and then pass 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 asserted) and pass control to the trace handler before returning from the original exception.

3.7.7 Unimplemented Line-A Opcode

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.

3.7.8 Unimplemented Line-F Opcode

A line-F opcode is defined when bits 15-12 of the opword are 0b1111. This exception is generated by attempted execution of an undefined line-F opcode.

3.7.9 Debug Interrupt

This special type of program interrupt is discussed in detail in ." This exception 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).

3.7.10   RTE and Format Error Exception

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.

3.7.11   TRAP Instruction Exception

The TRAP #n instruction always forces an exception as part of its execution and is useful for implementing system calls.

3.7.12   Interrupt Exception

Interrupt exception processing includes interrupt recognition and the fetch of the appropriate vector from the interrupt controller using an IACK cycle. See ," for details on the interrupt controller.

3.7.13   Fault-on-Fault Halt

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 force the processor to exit this halted state.

3.7.14   Reset Exception

Asserting the reset input signal 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 S bit and disables tracing by clearing the T bit in the SR. This exception also clears the M bit and sets the processor's interrupt priority mask in the SR to the highest level (level 7). Next, the VBR is initialized to zero (0x00000000). 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 of the modules in this user's manual for details on these registers.

Once the processor is granted the bus, it then performs two longword read bus cycles. The first longword at address 0 is loaded into the stack pointer and the second longword at address 4 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 halted 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 BDM to determine the hardware configuration.

Information loaded into D0 defines the processor hardware configuration as shown in Figure 3-11.

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAC	DIV	EMAC	FPU	MMU	0	0	0	ISA_REV				DBG_REV			
W																
Reset	1	1	0	0	0	0	0	0	1	0	0	0	1	0	0	1

Figure 3-6. D0 Hardware Configuration Info

Table 3-9. D0 Hardware Configuration Info Field Description

Bits	Field	Description
23–20	VERSION	ColdFire core version number. This 4-bit field defines the hardware microarchitecture (version) of the ColdFire core. if Version = 0b0010, then Version 2 ColdFire if Version = 0b0011, then Version 3 ColdFire if Version = 0b0100, then Version 4 ColdFire if Version = 0b0101, then Version 5 ColdFire All other values are reserved for future use. The upper 12 bits of the D0 reset value directly identify the ColdFire core version, e.g., "CF2" for a Version 2 core, "CF3" for a Version 3 core, etc.
19–16	PREV	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 the 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.) If an EMAC is present, this bit is cleared.

Table 3-9. D0 Hardware Configuration Info Field Description

Bits	Field	Description
14	DIV	Divide present. This bit signals if the hardware divider (DIV) is present in the processor core. Certain early V2 core implementations, e.g., MCF5202, MCF5204, MCF5206, did not include hardware support for integer divide operations. 0 Divide execute engine not present in core. 1 Divide execute engine is present in core.(This is the value used for this device.)
13	EMAC	EMAC present. This bit signals if the optional enhanced multiply-accumulate (EMAC) execution engine is present in the processor core. 0 EMAC execute engine not present in core. (This is the value used for this device.) 1 EMAC execute engine is present in core.If an MAC is present, this bit is cleared.
12	FPU	FPU present. This bit signals if the optional floating-point (FPU) execution engine is present in the processor core. 0 FPU execute engine not present in core. (This is the value used for this device.) 1 FPU execute engine is present in core.
11	MMU	MMU present. This bit signals if the optional virtual memory management unit (MMU) is present in the processor core. 0 MMU execute engine not present in core.(This is the value used for this device.) 1 MMU execute engine is present in core.
10–	—	Reserved.
7–4	ISA_REV	ISA revision. This 4-bit field defines the instruction set architecture (ISA) revision level implemented in the ColdFire processor core. if ISA_REV = 0b0000, then ISA_A if ISA_REV = 0b1000, then ISA_A+. (This is the value used for this device.) if ISA_REV = 0b0001, then ISA_B if ISA_REV = 0b0010, then ISA_C All other values are reserved for future use.
3–0	DBG_REV	Debug module revision number. This 4-bit field defines the revision level of the debug module implemented in the ColdFire processor core. if DBG_REV = 0b0000, then Debug_A. (if DBG_REV = 0b1000, then Debug_A+ if DBG_REV = 0b0001, then Debug_B if DBG_REV = 0b1001, then Debug_B+. (This is the value used for this device.) if DBG_REV = 0b0010, then Debug_C if DBG_REV = 0b0011, then Debug_D if DBG_REV = 0b0100, then Debug_E All other values are reserved for future use.

Information loaded into D1 defines the local memory hardware configuration as shown in Figure 3-11.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLSZ		ICAS		ICSZ				SRAM0SZ				—			
W																
Reset	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0
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	MBSZ		DCAS		DCSZ[3:0]				SRAM1SZ[3:0]				—			
W																
Reset	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0
Reset	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0
Reset	0	0	0	1	0	0	0	0	0	1	1	1	0	0	0	0

Figure 3-7. D1 Hardware Configuration Info

Table 3-10. D1 Local Memory Hardware Configuration Information Field Description

Bits	Name	Description
31–30	CLSZ	Cache line size. This field is fixed to a hex value of 0x0 indicating a 16-byte cache line size.
29–28	ICAS	Instruction cache associativity. 00 Four-way. 01 Direct mapped. (This is the value used for this device)
27–24	ICSZ	Instruction cache size. 0000 No instruction cache. 0001 512B instruction cache. 0010 1KB instruction cache. 0011 2KB instruction cache. 0100 4KB instruction cache. 0101 8KB instruction cache. 0110 16KB instruction cache. 0111 32KB instruction cache. 1000 64KB instruction cache. 0x9–0xF Reserved. All other values do not apply for this device.

Table 3-10. D1 Local Memory Hardware Configuration Information Field Description

Bits	Name	Description
23–20	SRAM0SZ	SRAM bank 0 size. The first RAM bank can be used for either SRAM or Flash. The first encoding shown are used to indicate the size of a RAM bank, and the second set of encodings indicate the size for a Flash bank. RAM size encodings: if SRAM0SZ = 0b0000, then no SRAM0. if SRAM0SZ = 0b0001, then SRAM0 size is 512 bytes if SRAM0SZ = 0b0010, then SRAM0 size is 1 Kbytes if SRAM0SZ = 0b0011, then SRAM0 size is 2 Kbytes if SRAM0SZ = 0b0100, then SRAM0 size is 4 Kbytes if SRAM0SZ = 0b0101, then SRAM0 size is 8 Kbytes if SRAM0SZ = 0b0110, then SRAM0 size is 16 Kbytes if SRAM0SZ = 0b0111, then SRAM0 size is 32 Kbytes if SRAM0SZ = 0b1000, then SRAM0 size is 64 Kbytes if SRAM0SZ = 0b1001, then SRAM0 size is 128 Kbytes All other values are reserved for future use. Flash size encodings: 0000-0111, then no flash. if SRAM0SZ = 0b1000, then 64KB Flash. if SRAM0SZ = 0b1001, then 128KB Flash. if SRAM0SZ = 0b1010, then 256KB Flash. if SRAM0SZ = 0b1011, then 512KB Flash. All other values are reserved for future use.
19–16	—	Reserved
15–14	MBSZ	Mbus size. Encoded bus data width. This 2-bit field defines the width of the ColdFire Master Bus datapath. if MBSZ = 0b00, then 32-bit system bus datapath if MBSZ = 0b01, then 64-bit system bus datapath All other values are reserved for future use.
13–12	DCAS	D-Cache Associativity .This 2-bit field defines the D-Cache set-associativity. if DCAS = 0b00, then D-Cache is 4-way set-associative organization. if DCAS = 0b01, then D-Cache is direct-mapped organization. (This is the value used for this device) All other values are reserved for future use.
11–8	DCSZ	D-Cache size. Unified (instruction or data) cache size. if DCSZ = 0b0000, then no D-Cache. (This is the value used for this device) if DCSZ = 0b0001, then D-Cache size is 512 bytes if DCSZ = 0b0010, then D-Cache size is 1 Kbytes if DCSZ = 0b0011, then D-Cache size is 2 Kbytes if DCSZ = 0b0100, then D-Cache size is 4 Kbytes if DCSZ = 0b0101, then D-Cache size is 8 Kbytes if DCSZ = 0b0110, then D-Cache size is 16 Kbytes if DCSZ = 0b0111, then D-Cache size is 32 Kbytes if DCSZ = 0b1000, then D-Cache size is 64 Kbytes All other values are reserved for future use.

Table 3-10. D1 Local Memory Hardware Configuration Information Field Description

Bits	Name	Description
7–4	SRAM1SZ	SRAM bank 1 size. The second RAM bank can is used only for SRAM if SRAM1SZ = 0b0000, then no SRAM1 if SRAM1SZ = 0b0001, then SRAM1 size is 512 bytes if SRAM1SZ = 0b0010, then SRAM1 size is 1 Kbytes if SRAM1SZ = 0b0011, then SRAM1 size is 2 Kbytes if SRAM1SZ = 0b0100, then SRAM1 size is 4 Kbytes. if SRAM1SZ = 0b0101, then SRAM1 size is 8 Kbytes. if SRAM1SZ = 0b0110, then SRAM1 size is 16 Kbytes if SRAM1SZ = 0b0111, then SRAM1 size is 32 Kbytes if SRAM1SZ = 0b1000, then SRAM1 size is 64 Kbytes. if SRAM1SZ = 0b1001, then SRAM1 size is 128 Kbytes All other values are reserved for future use.
3-0	—	Reserved

Information loaded into D1 defines the local memory hardware configuration as shown in Figure 3-11.

3.8   Instruction Execution Timing

This section presents CF 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.

3.8.1   Timing Assumptions

For the timing data presented in this section, the following 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 this type 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 2 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, i.e., 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 3-11.

Table 3-11. Misaligned Operand References

address[1:0]	Size	Bus Operations	additional C(R/W)
X1	Word	Byte, Byte	2(1/0) if read 1(0/1) if write
X1	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

NOTE

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, as well as all internal core cycles required too 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).

3.8.2 MOVE Instruction Execution Times

Table 3-12 lists execution times for MOVE.{B,W} instructions; Table 3-13 lists timings for MOVE.L. For all tables in this section, the execution time of any instruction using the PC-relative effective addressing modes is the same for the comparable An-relative mode.The nomenclature "xxx.wl" refers to both forms of absolute addressing, xxx.w and xxx.l.

Table 3-12. 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)	(1/0)	(1/1)	(1/1)	(1/1)	(1/1)	(1/1))	(1/1)
(Ay)+	(1/0)	(1/1)	(1/1)	(1/1)	(1/1)	(1/1))	(1/1)
-(Ay)	(1/0)	(1/1)	(1/1)	(1/1)	(1/1)	(1/1)	(1/1)

Table 3-12. MOVE Byte and Word Execution Times (continued)

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

Table 3-13. 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)	(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)	—	—	—

3.9   Standard One Operand Instruction Execution Times

Table 3-14. 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)	—

Table 3-14. One Operand Instruction Execution Times (continued)

Opcode	<EA>	Effective Address							
		Rn	**(An)**	**(An)+**	**-(An)**	**(d16,An)**	**(d8,An,Xn*SF)**	**xxx.wl**	**#xxx**
clr.l	<ea>	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
ext.w	Dx	1(0/0)	—	—	—	—	—	—	—
ext.l	Dx	1(0/0)	—	—	—	—	—	—	—
extb.l	Dx	1(0/0)	—	—	—	—	—	—	—
ff1	Dx	1(0/0)	—	—	—	—	—	—	—
neg.l	Dx	1(0/0)	—	—	—	—	—	—	—
negx.l	Dx	1(0/0)	—	—	—	—	—	—	—
not.l	Dx	1(0/0)	—	—	—	—	—	—	—
scc	Dx	1(0/0)	—	—	—	—	—	—	—
swap	Dx	1(0/0)	—	—	—	—	—	—	—
tst.b	<ea>	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
tst.w	<ea>	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(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)

3.10 Standard Two Operand Instruction Execution Times

Table 3-15. 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)	—	—	—

Table 3-15. 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
bset	Dy,<ea>	2(0/0)	4(1/1)	41/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)	—
bset	#imm,<ea>	2(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	—	—	—
btst	Dy,<ea>	2(0/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
btst	#imm,<ea>	1(0/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	—	—	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.l	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
divs.w	<ea>,Dx	20(0/0)	23(1/0)	23(1/0)	23(1/0)	23(1/0)	24(1/0)	23(1/0)	20(0/0)
divu.w	<ea>,Dx	20(0/0)	23(1/0)	23(1/0)	23(1/0)	23(1/0)	24(1/0)	23(1/0)	20(0/0)
divs.l	<ea>,Dx	≤35(0/0)	≤38(1/0)	≤38(1/0)	≤38(1/0)	≤38(1/0)	—	—	—
divu.l	<ea>,Dx	≤35(0/0)	≤38(1/0)	≤38(1/0)	≤38(1/0)	≤38(1/0)	—	—	—
eor.l	Dy,<ea>	1(0/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
eori.l	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
lea	<ea>,Ax	—	1(0/0)	—	—	1(0/0)	2(0/0)	1(0/0)	—
lsl.l	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
lsr.l	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
moveq	#imm,Dx	—	—	—	—	—	—	—	1(0/0)
muls.w	<ea>y, Dx	4(0/0)	6(1/0)	6(1/0)	6(1/0)	6(1/0)	7(1/0)	6(1/0)	4(1/0)
mulu.w	<ea>y, Dx	4(0/0)	6(1/0)	6(1/0)	6(1/0)	6(1/0)	7(1/0)	6(1/0)	4(1/0)
muls.l	<ea>y, Dx	4(0/0)	6(1/0)	6(1/0)	6(1/0)	6(1/0)	—	—	—
mulu.l	<ea>y, Dx	4(0/0)	6(1/0)	6(1/0)	6(1/0)	6(1/0)	—	—	—
or.l	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
or.l	Dy,<ea>	—	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
ori.l	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
rems.l	<ea>,Dx	≤35(0/0)	≤38(1/0)	≤38(1/0)	≤38(1/0)	≤38(1/0)	—	—	—
remu.l	<ea>,Dx	≤35(0/0)	≤38(1/0)	≤38(1/0)	≤38(1/0)	≤38(1/0)	—	—	—
sub.l	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
sub.l	Dy,<ea>	—	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
subi.l	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
subq.l	#imm,<ea>	1(0/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
subx.l	Dy,Dx	1(0/0)	—	—	—	—	—	—	—

3.11 Miscellaneous Instruction Execution Times

Table 3-16. 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)	—	—	—	—	—	—	—
move.l	Ay,USP	3(0/0)	—	—	—	—	—	—	—
move.l	USP,Ax	3(0/0)	—	—	—	—	—	—	—
move.w	CCR,Dx	1(0/0)	—	—	—	—	—	—	—
move.w	<ea>,CCR	1(0/0)	—	—	—	—	—	—	1(0/0)
move.w	SR,Dx	1(0/0)	—	—	—	—	—	—	—
move.w	<ea>,SR	7(0/0)	—	—	—	—	—	—	7(0/0) [2]
movec	Ry,Rc	9(0/1)	—	—	—	—	—	—	—
movem.l	<ea>,&list	—	1+n(n/0)	—	—	1+n(n/0)	—	—	—
movem.l	&list,<ea>	—	1+n(0/n)	—	—	1+n(0/n)	—	—	—
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	—	—	—	—	—	—	—	15(1/2)
trapf		1(0/0)	—	—	—	—	—	—	—
trapf.w		1(0/0)	—	—	—	—	—	—	—
trapf.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)	3(1/0)
wdebug	<ea>	—	5(2/0)	—	—	5(2/0)	—	—	—

[1]n is the number of registers moved by the MOVEM opcode.

[2]If a MOVE.W #imm,SR instruction is executed and imm[13] = 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).

3.12 MAC Instruction Execution Times

Table 3-17. EMAC Instruction Execution Times

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An, Xn*SF)	xxx.wl	#xxx
muls.w	<ea>y, Dx	4(0/0)	(1/0)	(1/0)	(1/0)	(1/0)	1/0)	(1/0)	4(/0)
mulu.w	<ea>y, Dx	4(0/0)	(1/0)	(1/0)	(1/0)	(1/0)	1/0)	(1/0)	4(/0)
muls.l	<ea>y, Dx	4(0/0)	(1/0)	(1/0)	(1/0)	(1/0)	—	—	—
mulu.l	<ea>y, Dx	4(0/0)	(1/0)	(1/0)	(1/0)	(1/0)	—	—	—
mac.w	Ry, Rx, Raccx	1(0/0)	—	—	—	—	—	—	—
mac.l	Ry, Rx, Raccx	1(0/0)	—	—	—	—	—	—	—
msac.w	Ry, Rx, Raccx	1(0/0)	—	—	—	—	—	—	—
msac.l	Ry, Rx, Raccx	1(0/0)	—	—	—	—	—	—	—
mac.w	Ry, Rx, <ea>, Rw, Raccx	—	(1/0)	(1/0)	(1/0)	(1/0)[1]	—	—	—
mac.l	Ry, Rx, <ea>, Rw, Raccx	—	(1/0)	(1/0)	(1/0)	(1/0)[1]	—	—	—
msac.w	Ry, Rx, <ea>, Rw	—	(1/0)	(1/0)	(1/0)	(1/0)[1]	—	—	—
msac.l	Ry, Rx, <ea>, Rw, Raccx	—	(1/0)	(1/0)	(1/0)	(1/0)[1]	—	—	—
mov.l	<ea>y, Raccx	1(0/0)	—	—	—	—	—	—	1(0/0)
mov.l	Raccy,Raccx	1(0/0)	—	—	—	—	—	—	—
mov.l	<ea>y, MACSR	5(0/0)	—	—	—	—	—	—	5(0/0)
mov.l	<ea>y, Rmask	4(0/0)	—	—	—	—	—	—	4(0/0)
mov.l	<ea>y,Raccext01	1(0/0)	—	—	—	—	—	—	1(0/0)
mov.l	<ea>y,Raccext23	1(0/0)	—	—	—	—	—	—	1(0/0)
mov.l	Raccx,<ea>x	1(0/0)[2]	—	—	—	—	—	—	—
mov.l	MACSR,<ea>x	1(0/0)	—	—	—	—	—	—	—
mov.l	Rmask, <ea>x	1(0/0)	—	—	—	—	—	—	—
mov.l	Raccext01,<ea.x	1(0/0)	—	—	—	—	—	—	—
mov.l	Raccext23,<ea>x	1(0/0)	—	—	—	—	—	—	—

[1] Effective address of (d16,PC) not supported

[2] Storing an accumulator requires one additional processor clock cycle when saturation is enabled, or fractional rounding is performed (MACSR[7:4] = 1---, -11-, --11)

3.13 Branch Instruction Execution Times

Table 3-18. 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**
bsr		—	—	—	—	(0/1)	—	—	—
jmp	<ea>	—	(0/0)	—	—	(0/0)	(0/0)	(0/0)	—
jsr	<ea>	—	(0/1)	—	—	(0/1)	(0/1)	(0/1)	—
rte		—	—	(2/0)	—	—	—	—	—
rts		—	—	(1/0)	—	—	—	—	—

Table 3-19. BRA, Bcc Instruction Execution Times

Opcode	Forward Taken	Forward Not Taken	Backward Taken	Backward Not Taken
bra	(0/0)	—	(0/0)	—
bcc	(0/0)	1(0/0)	(0/0)	(0/0)

Chapter 4
Hardware Multiply/Accumulate (MAC) Unit

This chapter describes the multiply/accumulate (MAC) unit, which executes integer multiply, multiply-accumulate, and miscellaneous register instructions. The MAC is integrated into the operand execution pipeline (OEP).

4.1   Overview

The MAC unit provides hardware support for a limited set of digital signal processing (DSP) operations used in embedded code, while supporting the integer multiply instructions in the ColdFire microprocessor family.

The MAC unit provides signal processing capabilities which are useful in a variety of applications including digital audio and servo control. Integrated as an execution unit in the processor's OEP, the MAC unit implements a three-stage arithmetic pipeline optimized for 16 x 16 multiplies. Both 16- and 32-bit input operands are supported by this design in addition to a full set of extensions for signed and unsigned integers plus signed, fixed-point fractional input operands.

The MAC unit provides functionality in three related areas:
- Signed and unsigned integer multiplies
- Multiply-accumulate operations supporting signed, unsigned, and signed fractional operands
- Miscellaneous register operations

Each of the three areas of support is addressed in detail in the succeeding sections. Logic that supports this functionality is contained in a MAC module, as shown in Figure 4-1.

The MAC unit is tightly coupled to the OEP and features a three-stage execution pipeline. To minimize silicon costs, the ColdFire MAC is optimized for 16 x 16 multiply instructions. The OEP can issue a 16 x 16 multiply with a 32-bit accumulation and fetch a 32-bit operand in the same cycle. A 32 x 32 multiply with a 32-bit accumulation takes three cycles before the next instruction can be issued. Figure 4-1 shows the basic functionality of the ColdFire MAC. A full set of instructions is provided for signed and unsigned integers plus signed, fixed-point, fractional input operands.
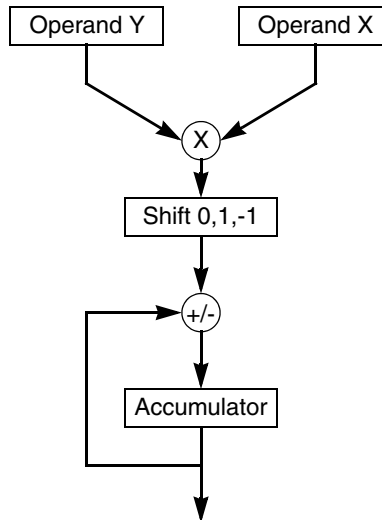
Figure 4-1. ColdFire MAC Multiplication and Accumulation

The MAC unit is an extension of the basic multiplier found on most microprocessors. It can perform operations native to signal processing algorithms in an acceptable number of cycles, given the application constraints. For example, small digital filters can tolerate some variance in the execution time of the algorithm; larger, more complicated algorithms, such as orthogonal transforms, may have more demanding speed requirements exceeding the scope of any processor architecture and requiring a fully developed DSP implementation.

The M68000 architecture was not designed for high-speed signal processing, and a large DSP engine would be excessive in an embedded environment. In striking a middle ground between speed, size, and functionality, the ColdFire MAC unit is optimized for a small set of operations that involve multiplication and cumulative additions. Specifically, the multiplier array is optimized for single-cycle, 16 x 16 multiplies producing a 32-bit result, with a possible accumulation cycle following. This is common in a large portion of signal processing applications. In addition, the ColdFire core architecture has been modified to allow for an operand fetch in parallel with a multiply, increasing overall performance for certain DSP operations.

4.1.1   MAC Programming Model

Figure 4-2 shows the registers in the MAC portion of the user programming model.
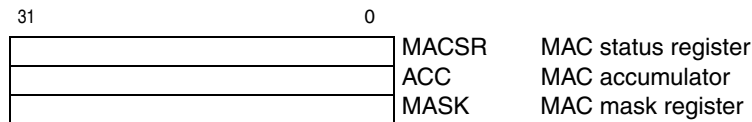
Figure 4-2. MAC Programming Model

These registers are described as follows:

- Accumulator (ACC)—This 32-bit, read/write, general-purpose register is used to accumulate the results of MAC operations.
- Mask register (MASK)—This 16-bit general-purpose register provides an optional address mask for MAC instructions that fetch operands from memory. It is useful in the implementation of circular queues in operand memory.
- MAC status register (MACSR)—This 8-bit register defines configuration of the MAC unit and contains indicator flags affected by MAC instructions. Unless noted otherwise, the setting of MACSR indicator flags is based on the final result, that is, the result of the final operation involving the product and accumulator.

4.1.2 General Operation

The MAC unit supports the ColdFire integer multiply instructions (MULS and MULU) and provides additional functionality for multiply-accumulate operations. The added MAC instructions to the ColdFire ISA provide for the multiplication of two numbers, followed by the addition or subtraction of this number to or from the value contained in the accumulator. The product may be optionally shifted left or right one bit before the addition or subtraction takes place. Hardware support for saturation arithmetic may be enabled to minimize software overhead when dealing with potential overflow conditions using signed or unsigned operands.

These MAC operations treat the operands as one of the following formats:

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

To maintain compactness, the MAC module is optimized for 16-bit multiplications. 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 63-bit product is calculated and either truncated or rounded to a 32-bit result using the round-to-nearest (even) method.

Because the multiplier array is implemented in a 3-stage pipeline, MAC instructions can have an effective issue rate of one clock for word operations, three for longword integer operations, and four for 32-bit fractional operations. Arithmetic operations use register-based input operands, and summed values are stored internally in the accumulator. Thus, an additional MOVE instruction is necessary to store data in a general-purpose register. MAC instructions can choose the upper or lower word of a register as the input, which helps filtering operations in which one data register is loaded with input data and another is loaded with coefficient data. Two 16-bit MAC operations can be performed without fetching additional operands between instructions by alternating the word choice during the calculations.

The need to move large amounts of data quickly can limit throughput in DSP engines. However, data can be moved efficiently by using the MOVEM instruction, which automatically generates line-sized burst references and is ideal for filling registers quickly with input data, filter coefficients, and output data. Loading an operand from memory into a register during a MAC operation makes some DSP operations, especially filtering and convolution, more manageable.

The MACSR has a 4-bit operational mode field and three condition flags. The operational mode bits control the overflow/saturation mode, whether operands are signed or unsigned, whether operands are treated as integers or fractions, and how rounding is performed. Negative, zero, and overflow flags are also provided.

The three program-visible MAC registers, a 32-bit accumulator (ACC), the MAC mask register (MASK), and MACSR, are described in Section 4.1.1, "MAC Programming Model."

4.1.3   MAC Instruction Set Summary

The MAC unit supports the integer multiply operations defined by the baseline ColdFire architecture, as well as the new multiply-accumulate instructions. Table 4-1 summarizes the MAC unit instruction set.

Table 4-1. MAC Instruction Summary

Instruction	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 or subtracts the product to/from the accumulator
Multiply Accumulate with Load	MAC Ry,RxSF,Rw MSAC Ry,RxSF,Rw	Multiplies two operands, then adds or subtracts the product to/from the accumulator while loading a register with the memory operand
Load Accumulator	MOV.L {Ry,#imm},ACC	Loads the accumulator with a 32-bit operand
Store Accumulator	MOV.L ACC,Rx	Writes the contents of the accumulator to a register
Load MACSR	MOV.L {Ry,#imm},MACSR	Writes a value to the MACSR
Store MACSR	MOV.L MACSR,Rx	Writes the contents of MACSR to a register
Store MACSR to CCR	MOV.L MACSR,CCR	Writes the contents of MACSR to the processor's CCR register
Load MASK	MOV.L {Ry,#imm},MASK	Writes a value to MASK
Store MASK	MOV.L MASK,Rx	Writes the contents of MASK to a register

4.1.4   Data Representation

The MAC unit supports three basic operand types:

- Two's complement signed integer: In this format, an N-bit operand represents a number within the range $-2^{(N-1)} \leq$ operand $\leq 2^{(N-1)}$ - 1. The binary point is to the right of the least significant bit.
- Two's complement unsigned integer: In this format, an N-bit operand represents a number within the range $0 \leq$ operand $\leq 2^N$ - 1. The binary point is to the right of the least significant bit.
- 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 following formula:

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

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

For words and longwords, the greatest negative number that can be represented is -1, whose internal representation is 0x8000 and 0x0x8000_0000, respectively. The most positive word is 0x7FFF or $(1 - 2^{-15})$; the most positive longword is 0x7FFF_FFFF or $(1 - 2^{-31})$.

4.2   MAC Instruction Execution Timings

For information on MAC instruction execution timings, refer to Section 3.12, "MAC Instruction Execution Times Check CF2 MAC (EMAC #'s)."

Chapter 5
Static RAM (SRAM)

5.1   Introduction

This chapter is a description of the on-chip static RAM (SRAM) implementation that covers general operations, configuration, and initialization. It also provides information and examples showing how to minimize power consumption when using the SRAM.

5.1.1   Features

Features include the following:

- One 32-Kbyte SRAM
- Single-cycle access
- Physically located on processor's high-speed local bus
- Memory location programmable on any 0-modulo-32 Kbyte address
- Byte, word, and longword address capabilities

5.1.2   Operation

The SRAM module provides a general-purpose memory block that the ColdFire processor can access in a single cycle. The location of the memory block can be specified to any 0-modulo-32K address within the 256-Mbyte address space (0x8000_0000 - 0x8FFF_FFFF). The memory is ideal for storing critical code or data structures or for use as the system stack. Because the SRAM module is physically connected to the processor's high-speed local bus, it can service processor-initiated accesses or memory-referencing commands from the debug module.

The SRAM is dual-ported to provide DMA access. The SRAM is partitioned into two physical memory arrays to allow simultaneous access to both arrays by the processor core and another bus master. See Chapter 8, "System Control Module (SCM)," for more information.

5.2   Register Description

The SRAM programming model includes a description of the SRAM base address register (RAMBAR), SRAM initialization, and power management.

5.2.1 SRAM Base Address Register (RAMBAR)

The configuration information in the SRAM base address register (RAMBAR) controls the operation of the SRAM module.

- The RAMBAR holds the base address of the SRAM. The MOVEC instruction provides write-only access to this register.
- The RAMBAR can be read or written from the debug module.
- All undefined bits in the register are reserved. These bits are ignored during writes to the RAMBAR and return zeroes when read from the debug module.
- The RAMBAR valid bit is cleared by reset, disabling the SRAM module. All other bits are unaffected.

NOTE

Do not confuse this RAMBAR with the SCM RAMBAR in Section 8.4.2, "Memory Base Address Register (RAMBAR)." Although similar, this core RAMBAR enables core access to the SRAM memory, while the SCM RAMBAR enables peripheral (e.g. DMA) access to the SRAM.

The RAMBAR contains several control fields. These fields are shown in Figure 5-1.

Address: CPU + 0x0C05   Access: Core write-only
                                                                    Debug 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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W						BA											0	0	0	0	PRIU	PRIL	SPV	WP	0	0	C/I	SC	SD	UC	UD	V
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
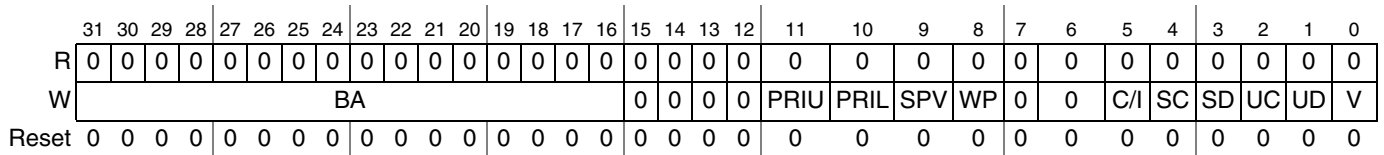
Figure 5-1. SRAM Base Address Register (RAMBAR)

Table 5-1. RAMBAR Field Descriptions

Field	Description
31–16 BA	Base address. Defines the 0-modulo-32K base address of the SRAM module. By programming this field, the SRAM may be located on any 32-Kbyte boundary within the processor's 256-Mbyte address space (0x8000_0000 - 0x8FFF_FFFF).
15–12	Reserved, should be cleared
11–10 PRIU PRIL	Priority bit. PRIU determines if DMA or CPU has priority in the upper 16K bank of memory. PRIL determines if DMA or CPU has priority in the lower 16K bank of memory. If a bit is set, the CPU has priority. If a bit is cleared, DMA has priority. Priority is determined according to the following table: <table><tr><th>PRIU,PRIL</th><th>Upper Bank Priority</th><th>Lower Bank Priority</th></tr><tr><td>00</td><td>CPU</td><td>CPU</td></tr><tr><td>01</td><td>CPU</td><td>DMA</td></tr><tr><td>10</td><td>DMA</td><td>CPU</td></tr><tr><td>11</td><td>DMA</td><td>DMA</td></tr></table> **Note:** The recommended setting for the priority bits is 00.

Table 5-1. RAMBAR Field Descriptions (continued)

Field	Description
9 SPV	Secondary port valid. Allows access by DMA 0 DMA access to memory is disabled. 1 DMA access to memory is enabled. **Note:** The BDE bit in the second RAMBAR register must also be set to allow dual port access to the SRAM. For more information, see Section 8.4.2, "Memory Base Address Register (RAMBAR)."
8 WP	Write protect. Allows only read accesses to the SRAM. When this bit is set, any attempted write access will generate an access error exception to the ColdFire processor core. 0 Allows read and write accesses to the SRAM module 1 Allows only read accesses to the SRAM module
7–6	Reserved, should be cleared.
5–1 C/I, SC, SD, UC, UD	Address space masks (AS*n*) These five bit fields allow certain types of accesses to be "masked," or inhibited from accessing the SRAM module. The address space mask bits are: C/I = CPU space/interrupt acknowledge cycle mask SC = Supervisor code address space mask SD = Supervisor data address space mask UC = User code address space mask UD = User data address space mask For each address space bit: 0 An access to the SRAM module can occur for this address space 1 Disable this address space from the SRAM module. If a reference using this address space is made, it is inhibited from accessing the SRAM module, and is processed like any other non-SRAM reference. These bits are useful for power management as detailed in Section 5.2.4, "Power Management." In most applications the C/I bit is set
0 V	Valid. When set, this bit enables the SRAM module; otherwise, the module is disabled. A hardware reset clears this bit. 0 Contents of RAMBAR are not valid 1 Contents of RAMBAR are valid

5.2.2   SRAM Initialization

After a hardware reset, the contents of the SRAM module are undefined. The valid bit of the RAMBAR is cleared, disabling the module. If the SRAM requires initialization with instructions or data, the following steps should be performed:

1. Load the RAMBAR, mapping the SRAM module to the desired location within the address space.
2. Read the source data and write it to the SRAM. There are various instructions to support this function, including memory-to-memory move instructions, or the MOVEM opcode. The MOVEM instruction is optimized to generate line-sized burst fetches on 0-modulo-16 addresses, so this opcode generally provides maximum performance.
3. After the data has been loaded into the SRAM, it may be appropriate to load a revised value into the RAMBAR with a new set of attributes. These attributes consist of the write-protect and address space mask fields.

The ColdFire processor or an external debugger using the debug module can perform these initialization functions.

5.2.3   SRAM Initialization Code

The following code segment describes how to initialize the SRAM. The code sets the base address of the SRAM at 0x2000_0000 and initializes the SRAM to zeros.

```
RAMBASE          EQU 0x20000000          ;set this variable to 0x20000000
RAMVALID         EQU 0x00000001
move.l           #RAMBASE+RAMVALID,D0    ;load RAMBASE + valid bit into D0.
movec.l          D0, RAMBAR              ;load RAMBAR and enable SRAM
```

The following loop initializes the entire SRAM to zero:

```
lea.l            RAMBASE,A0              ;load pointer to SRAM
move.l           #8192,D0                ;load loop counter into D0 (SRAM size/4)

SRAM_INIT_LOOP:
clr.l            (A0)+                   ;clear 4 bytes of SRAM
subq.l           #1,D0                   ;decrement loop counter
bne.b            SRAM_INIT_LOOP          ;if done, then exit; else continue looping
```

5.2.4   Power Management

If the SRAM is used only for data operands, setting the ASn bits associated with instruction fetches can decrease power dissipation. Additionally, if the SRAM contains only instructions, masking operand accesses can reduce power dissipation. Table 5-2 shows some examples of typical RAMBAR settings.

Table 5-2. Typical RAMBAR Setting Examples

Data Contained in SRAM	RAMBAR[7:0]
Instruction Only	0x2B
Data Only	0x35
Both Instructions And Data	0x21

Chapter 6
Clock Module

The clock module allows the MCF5213 to be configured for one of several clocking methods. Clocking modes include internal phase-locked loop (PLL) clocking with either an external clock reference or an external crystal reference supported by an internal crystal amplifier. The PLL can also be disabled and an external oscillator can be used to clock the device directly. The clock module contains the following:

- Crystal amplifier and oscillator (OSC)
- Phase-locked loop (PLL)
- Reduced frequency divider (RFD)
- Status and control registers
- Control logic

6.1   Features

Features of the clock module include the following:

- 1- to 16-MHz crystal, 8-MHz on-chip relaxation oscillator, or external oscillator reference options
- 2- to 10-MHz reference crystal oscillator for normal PLL mode
- System can be clocked from PLL or directly from crystal oscillator or relaxation oscillator
- Support for low-power modes
- Separate clock out signal
- 2^n ($0 \leq n \leq 15$) low-power divider for extremely low frequency operation

6.2   Modes of Operation

The clock module can be operated in normal PLL mode (default), 1:1 PLL mode, or external clock mode (PLL disabled).

6.2.1   Normal PLL Mode

In normal PLL mode, the PLL is fully programmable. It can synthesize frequencies ranging from 4x to 18x the reference frequency and has a post divider capable of reducing this synthesized frequency without disturbing the PLL. The PLL reference can be either a crystal oscillator or an external clock.

6.2.2   1:1 PLL Mode

In 1:1 PLL mode, the PLL synthesizes a frequency equal to the external clock input reference frequency. The post divider is not active.

6.2.3   External Clock Mode

In external clock mode, the PLL is bypassed, and the external clock is applied to EXTAL. The resulting operating frequency is equal to the external clock frequency.

6.3 Low-power Mode Operation

This subsection describes the operation of the clock module in low-power and halted modes of operation. Low-power modes are described in Chapter 7, "Power Management." Table 6-1 shows the clock module operation in low-power modes.

Table 6-1. Clock Module Operation in Low-power Modes

Low-power Mode	Clock Operation	Mode Exit
Wait	Clocks sent to peripheral modules only	Exit not caused by clock module, but normal clocking resumes upon mode exit
Doze	Clocks sent to peripheral modules only	Exit not caused by clock module, but normal clocking resumes upon mode exit
Stop	All system clocks disabled	Exit not caused by clock module, but clock sources are re-enabled and normal clocking resumes upon mode exit
Halted	Normal	Exit not caused by clock module

In wait and doze modes, the system clocks to the peripherals are enabled, and the clocks to the CPU and SRAM are stopped. Each module can disable its clock locally at the module level.

In stop mode, all system clocks are disabled. There are several options for enabling or disabling the PLL or crystal oscillator in stop mode, compromising between stop mode current and wakeup recovery time. The PLL can be disabled in stop mode, but requires a wakeup period before it can relock. The oscillator can also be disabled during stop mode, but requires a wakeup period to restart.

When the PLL is enabled in stop mode (STPMD[1:0]), the external CLKOUT signal can support systems using CLKOUT as the clock source.

There is also a fast wakeup option for quickly enabling the system clocks during stop recovery. This eliminates the wakeup recovery time but at the risk of sending a potentially unstable clock to the system. To prevent a non-locked PLL frequency overshoot when using the fast wakeup option, change the RFD divisor to the current RFD value plus one before entering stop mode.

In external clock mode, there are no wakeup periods for oscillator startup or PLL lock.

6.4 Block Diagram

Figure 6-1 shows a block diagram of the entire clock module. The PLL block in this diagram is expanded in detail in Figure 6-2.
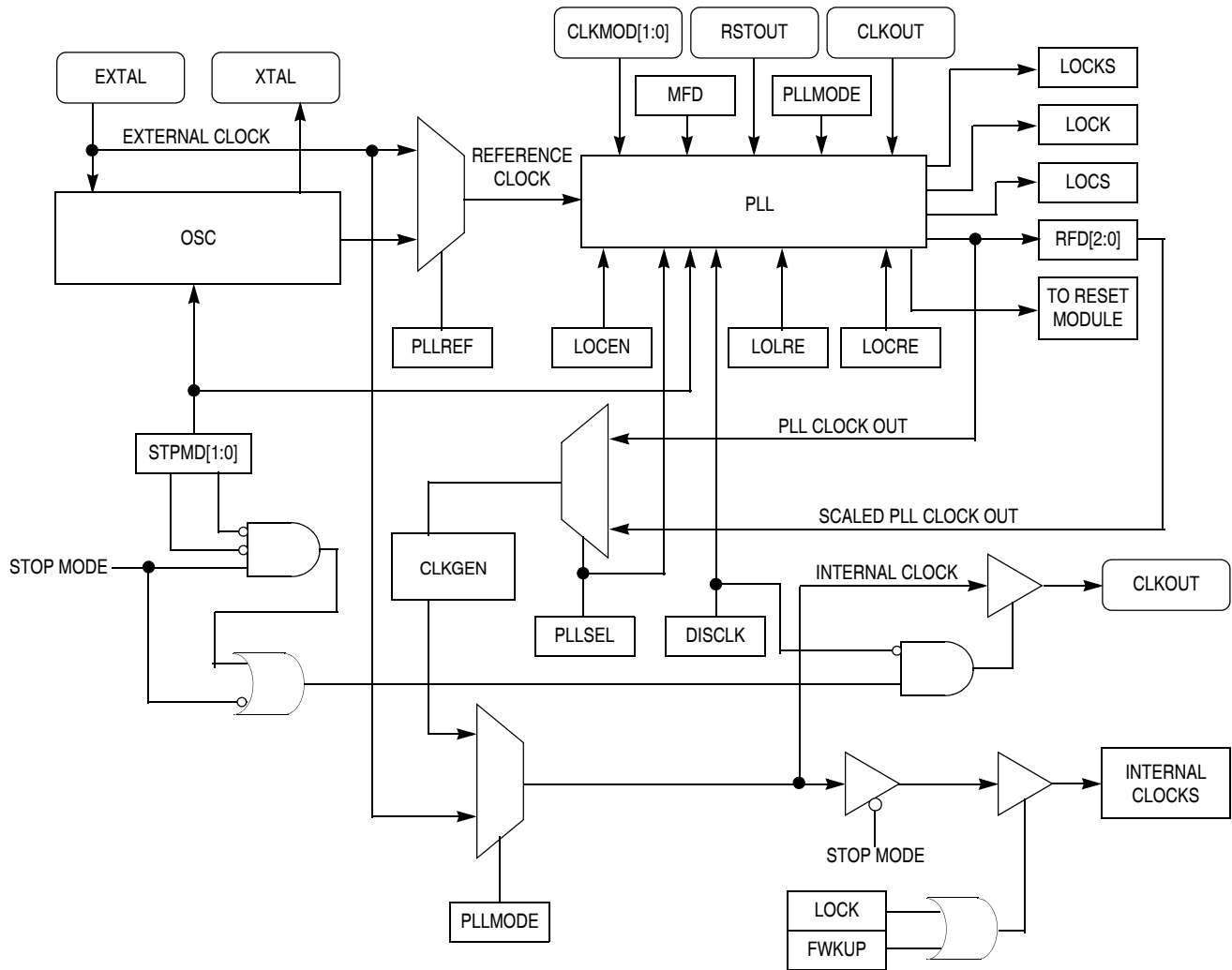
Figure 6-1. Clock Module Block Diagram
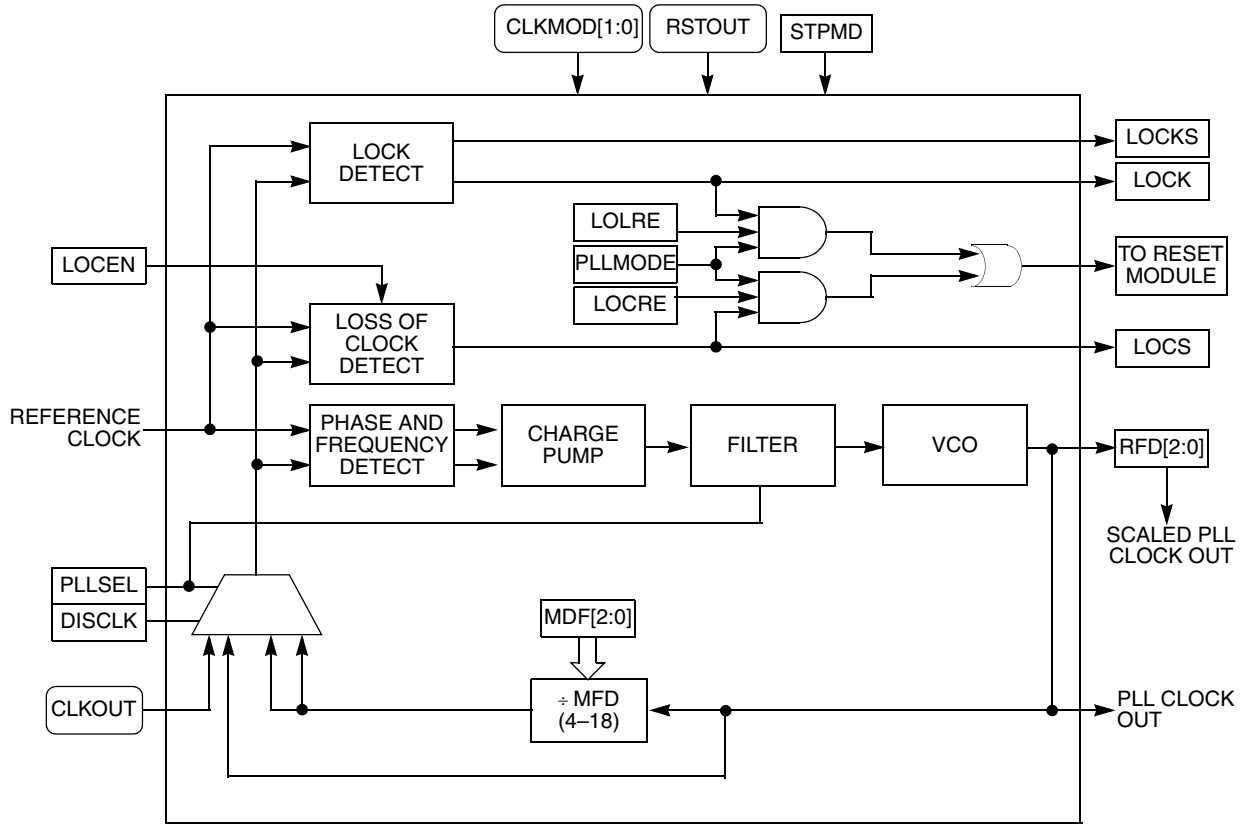
Figure 6-2. PLL Block Diagram

6.5   Signal Descriptions

The clock module signals are summarized in Table 6-2 and a brief description follows. For more detailed information, refer to Chapter 14, "Signal Descriptions."

Table 6-2. Signal Properties

Name	Function
EXTAL	Oscillator or clock input
XTAL	Oscillator output
CLKOUT	System clock output
CLKMOD[1:0]	Clock mode select inputs
$\overline{\text{RSTO}}$	Reset signal from reset controller

6.5.1   EXTAL

This input is driven by an external clock except when used as a connection to the external crystal when using the internal oscillator.

6.5.2   XTAL

This output is an internal oscillator connection to the external crystal. If CLKMOD0 is asserted during reset, XTAL is sampled to determine clocking mode.

6.5.3   CLKOUT

This output reflects the internal system clock.

6.5.4   CLKMOD[1:0]

These inputs are used to select the clock mode during chip configuration as described in Table 6-3.

Table 6-3. Clocking Modes

CLKMOD[1:0]	XTAL	Clocking Mode
00	0	PLL disabled, clock driven by external oscillator.
00	1	PLL disabled, clock driven by on-chip oscillator.
01	n/a	PLL disabled, clock driven by external crystal.
10	0	PLL in normal mode, clock driven by external oscillator.
10	1	PLL in normal mode, clock driven by on-chip oscillator.
11	n/a	PLL in normal mode, clock driven by external crystal.

6.5.5 $\overline{\text{RSTO}}$

The $\overline{\text{RSTO}}$ pin is asserted by one of the following:
- Internal system reset signal
- FRCRSTOUT bit in the reset control status register (RCR); see Section 29.4.1, "Reset Control Register (RCR)."

6.6   Memory Map and Registers

The clock module programming model consists of these registers:
- Synthesizer control register (SYNCR)—defines clock operation
- Synthesizer status register (SYNSR)— reflects clock status

6.6.1   Module Memory Map

Table 6-4. Clock Module Memory Map

IPSBAR Offset	Register Name	Access[1]
0x0012_0000	Synthesizer Control Register (SYNCR)	S
0x0012_0002	Synthesizer Status Register (SYNSR)	S
0x0012_0004	Reserved	—
0x0012_0006	LPCR - low power control register	S

[1] S = CPU supervisor mode access only.

Table 6-5. SCM Additional Registers Memory Map

IPSBAR Offset	Register Name	Access
0x0000_000C	PPRH - Peripheral Power Management Register - High	S
0x0000_0018	PPRL - Peripheral Power Management Register - Low	S

6.6.2   Register Descriptions

This subsection provides a description of the clock module registers.

6.6.2.1   Synthesizer Control Register (SYNCR)

	15	14	13	12	11	10	9	8
Field	LOLRE	MFD2	MFD1	MFD0	LOCRE	RFD2	RFD1	RFD0
Reset				0001_0000				
R/W				R/W				

	7	6	5	4	3	2	1	0
Field	LOCEN	DISCLK	FWKUP	—	—	CLKSRC[1]	PLLMODE	PLLEN[1]
Reset				0000_0010				
R/W				R/W				
Address				IPSBAR + 0x0012_0000				

Figure 6-3. Synthesizer Control Register (SYNCR)

[1] The reset value of PLLEN and CLKSRC depend on the value of CLKMOD1 during reset (set to 1 if PLL is enabled when the device emerges from reset)

Table 6-6. SYNCR Field Descriptions

Bit(s)	Name	Description
15	LOLRE	Loss of lock reset enable. Determines how the system handles a loss of lock indication. When operating in normal mode or 1:1 PLL mode, the PLL must be locked before setting the LOLRE bit. Otherwise reset is immediately asserted. To prevent an immediate reset, the LOLRE bit must be cleared before writing the MFD[2:0] bits or entering stop mode with the PLL disabled. 1 Reset on loss of lock 0 No reset on loss of lock Note: In external clock mode, the LOLRE bit has no effect.

Table 6-6. SYNCR Field Descriptions (continued)

Bit(s)	Name	Description
14–12	MFD	Multiplication Factor Divider. Contain the binary value of the divider in the PLL feedback loop. The MFD[2:0] value is the multiplication factor applied to the reference frequency. When MFD[2:0] are changed or the PLL is disabled in stop mode, the PLL loses lock. In 1:1 PLL mode, MFD[2:0] are ignored, and the multiplication factor is one. Note: In external clock mode, the MFD[2:0] bits have no effect. The following table illustrates the system frequency multiplier of the reference frequency[1] in normal PLL mode. *(see table below)* [1] $f_{sys} = f_{ref}$ x 2(MFD + 2)/2 exp RFD; f_{ref} x 2(MFD + 2) $\leq$ 80 MHz, $f_{sys} \leq$ 80 MHz [2] MFD = 000 not valid for f_{ref} < 3 MHz [3] Default value out of reset
11	LOCRE	Loss-of-clock reset enable. Determines how the system handles a loss-of-clock condition. When the LOCEN bit is clear, LOCRE has no effect. If the LOCS flag in SYNSR indicates a loss-of-clock condition, setting the LOCRE bit causes an immediate reset. To prevent an immediate reset, the LOCRE bit must be cleared before entering stop mode with the PLL disabled. 1 Reset on loss-of-clock 0 No reset on loss-of-clock Note: In external clock mode, the LOCRE bit has no effect.
10–8	RFD	Reduced frequency divider field. The binary value written to RFD[2:0] is the PLL frequency divisor. See table in MFD bit description. Changing RFD[2:0] does not affect the PLL or cause a relock delay. Changes in clock frequency are synchronized to the next falling edge of the current system clock. To avoid surpassing the allowable system operating frequency, write to RFD[2:0] only when the LOCK bit is set.
7	LOCEN	Enables the loss-of-clock function. LOCEN does not affect the loss of lock function. 1 Loss-of-clock function enabled 0 Loss-of-clock function disabled Note: In external clock mode, the LOCEN bit has no effect.
6	DISCLK	Disable CLKOUT determines whether CLKOUT is driven. Setting the DISCLK bit holds CLKOUT low. 1 CLKOUT disabled 0 CLKOUT enabled

Table embedded in MFD description:

		MFD[2:0]							
		000[2] (4x)	001 (6x)	010 (8x)[3]	011 (10x)	100 (12x)	101 (14x)	110 (16x)	111 (18x)
RFD[2:0]	000 (÷ 1)	4	6	8	10	12	14	16	18
	001 (÷ 2)[3]	2	3	4	5	6	7	8	9
	010 (÷ 4)	1	3/2	2	5/2	3	7/2	4	9/2
	011 (÷ 8)	1/2	3/4	1	5/4	3/2	7/4	2	9/4
	100 (÷ 16)	1/4	3/8	1/2	5/8	3/4	7/8	1	9/8
	101 (÷ 32)	1/8	3/16	1/4	5/16	3/8	7/16	1/2	9/16
	110 (÷ 64)	1/16	3/32	1/8	5/32	3/16	7/32	1/4	9/32
	111 (÷ 128)	1/32	3/64	1/16	5/64	3/32	7/64	1/8	9/64

Preliminary

Table 6-6. SYNCR Field Descriptions (continued)

Bit(s)	Name	Description
5	FWKUP	Fast wakeup determines when the system clocks are enabled during wakeup from stop mode. 1 System clocks enabled on wakeup regardless of PLL lock status 0 System clocks enabled only when PLL is locked or operating normally Note: When FWKUP = 0, if the PLL or oscillator is enabled and unintentionally lost in stop mode, the PLL wakes up in self-clocked mode or reference clock mode depending on the clock that was lost. In external clock mode, the FWKUP bit has no effect on the wakeup sequence.
4–3	—	Reserved, should be cleared.
2	CLKSRC	Determines whether the PLL output clock or the PLL reference clock is to drive the system clock. This bit is ignored when the PLL is disabled, in which case the PLL reference clock will drive the system clock. Having this separate bit allows the PLL to first be enabled, and then the system clock can be switched to the PLL output clock only after the PLL has locked. When disabling the PLL, the clock can be switched before disabling the PLL so that a smooth transfer is ensured. 0) PLLreference clock (input clock) drives the system clock. 1) PLL output clock drives the system clock (provided the PLL is enabled).
1	PLLMODE	Determines the operating mode of the PLL. This bit should only be changed after reset with the PLL disabled. 0) PLL operates in 1:1 mode 1) PLL operates in normal mode
0	PLLEN	Enables and disables the PLL. If the PLL is enabled out of reset the chip will not leave the reset state until the PLL is locked and the system clock will be driven by the PLL output clock. Use the CLKSRC control bit to switch the system clock between the PLL output clock and PLL bypass clock once the PLL is enabled. 0) PLL is disabled 1) PLL is enabled

6.6.2.2 Synthesizer Status Register (SYNSR)

The SYNSR is a read-only register that can be read at any time. Writing to the SYNSR has no effect and terminates the cycle normally.

Figure 6-4. Synthesizer Status Register (SYNSR)

	7	6	5	4	3	2	1	0
Field	EXTOSC	OCOSC	CRYOSC	LOCKS	LOCK	LOCS	—	
Reset	See note 1			See note 2		000		
R/W	R							
Address	IPSBAR + 0x0012_0002							

Note: 1. Reset state determined during reset configuration.
   2. See the LOCKS and LOCK bit descriptions.

Table 6-7. SYNSR Field Descriptions

Bit(s)	Name	Description
7	EXTOSC	Indicates if an external oscillator is providing the reference clock source 0) Reference clock is not external oscillator 1 Reference clock is external oscillator
6	OCOSC	Indicates if the on-chip oscillator is providing the reference clock source. 0 Reference clock is not on-chip oscillator 1 Reference clock is on-chip oscillator
5	CRYOSC	Indicates if an external crystal is providing the reference clock source 0 Reference clock is not external crystal 1 Crystal clock reference
4	LOCKS	Sticky indication of PLL lock status. 1 No unintentional PLL loss of lock since last system reset or MFD change 0 PLL loss of lock since last system reset or MFD change or currently not locked due to exit from STOP with FWKUP set The lock detect function sets the LOCKS bit when the PLL achieves lock after: • A system reset • A write to SYNCR that changes the MFD[2:0] bits When the PLL loses lock, LOCKS is cleared. When the PLL relocks, LOCKS remains cleared until one of the two listed events occurs. In stop mode, if the PLL is intentionally disabled, then the LOCKS bit reflects the value prior to entering stop mode. However, if FWKUP is set, then LOCKS is cleared until the PLL regains lock. Once lock is regained, the LOCKS bit reflects the value prior to entering stop mode. Furthermore, reading the LOCKS bit at the same time that the PLL loses lock does not return the current loss of lock condition. In external clock mode, LOCKS remains cleared after reset. In normal PLL mode and 1:1 PLL mode, LOCKS is set after reset.
3	LOCK	Set when the PLL is locked. PLL lock occurs when the synthesized frequency is within approximately 0.75 percent of the programmed frequency. The PLL loses lock when a frequency deviation of greater than approximately 1.5 percent occurs. Reading the LOCK flag at the same time that the PLL loses lock or acquires lock does not return the current condition of the PLL. The power-on reset circuit uses the LOCK bit as a condition for releasing reset. If operating in external clock mode, LOCK remains cleared after reset. 1 PLL locked 0 PLL not locked

Bit(s)	Name	Description
2	LOCS	Sticky indication of whether a loss-of-clock condition has occurred at any time since exiting reset in normal PLL and 1:1 PLL modes. LOCS = 0 when the system clocks are operating normally. LOCS = 1 when system clocks have failed due to a reference failure or PLL failure. After entering stop mode with FWKUP set and the PLL and oscillator intentionally disabled (STPMD[1:0] = 11), the PLL exits stop mode in the SCM while the oscillator starts up. During this time, LOCS is temporarily set regardless of LOCEN. It is cleared once the oscillator comes up and the PLL is attempting to lock. If a read of the LOCS flag and a loss-of-clock condition occur simultaneously, the flag does not reflect the current loss-of-clock condition. A loss-of-clock condition can be detected only if LOCEN = 1 or the oscillator has not yet returned from exit from stop mode with FWKUP = 1. 1 Loss-of-clock detected since exiting reset or oscillator not yet recovered from exit from stop mode with FWKUP = 1 0 Loss-of-clock not detected since exiting reset Note: The LOCS flag is always 0 in external clock mode.
1–0	—	Reserved, should be cleared.

6.6.2.3   Low Power Control Register (LPCR)

Figure 6-5. Low Power Control Register

	7	6	5	4	3	2	1	0
Field	—	—	—	—	LPD3	LPD2	LPD1	LPD0
Reset	0000_0000							
R/W	R				R/W			
Address	IPSBAR + 0x0000_0007							

The low power divider is a 4-bit field that divides down the system clock (regardless if the reference clock or PLL clock is driving the system clock) by a factor of 2^n (where n is a number from 0 to 15 represented by the 4 bit field). The clock change takes effect with the next rising edge of the system clock

6.6.3   PPM Register Descriptions

This subsection provides a description of the peripheral power management registers in the SCM.

6.6.3.1 Peripheral Power Management Register High (PPMRH)

Figure 6-6. Peripheral Power Management Register High

	31	30	29	28	27	26	25	24
Field	—	—	—	—	—	—	—	—
Reset	0000_0000							
R/W	R							

	23	22	21	20	19	18	17	16
Field	—	—	—	—	—	—	—	—
Reset	0000_0000							
R/W	R							

	15	14	13	12	11	10	9	8
Field	—	—	—	—	CDCFM	CDFCAN	CDPWM	CDICOC
Reset	0000_0000							
R/W	R/W							

	7	6	5	4	3	2	1	0
Field	CDADC	—	—	CDPIT1	CDPIT0	—	CDEPORT	CDPORTS
Reset	0000_0000							
R/W	R/W							
Address	IPSBAR+0000_000C							

Table 6-8. PPMRH Field Descriptions

Bit(s)	Name	Description
31–12	—	Reserved, should be cleared.
11	CDCFM	Disable clock to the CFM (Common Flash Module) 0) CFM module clock is enabled 1) CFM module clock is disabled
10	CDFCAN	Disable clock to the FlexCAN module. 0) FlexCAN module clock is enabled 1) FlexCAN module clock is disabled
9	CDPWM	Disable clock to the PWM module. 0) PWM module clock is enabled 1) PWM module clock is disabled
8	CDICOC	Disable clock to the 16 bit timer module (ICOC). 0) ICOC module clock is enabled 1) ICOC module clock is disabled
7	CDADC	Disable clock to the ADC module. 0) ADC module clock is enabled 1) ADC module clock is disabled
6–5	—	Reserved, should be cleared.
4	CDPIT1	Disable clock to the PIT1 module. 0) PIT0 module clock is enabled 1) PIT1 module clock is disabled
3	CDPIT0	Disable clock to the PIT0 module. 0) PIT0 module clock is enabled 1) PIT0 module clock is disabled
2	Reserved	—

MCF5213 Reference Manual, Rev. 1.1

Preliminary

| 1 | CDEPORT | Disable clock to the EPORT module.
0) EPORT module clock is enabled
1) EPORT module clock is disabled |
| 0 | CDPORTS | Disable clock to the Ports module.
0) Ports module clock is enabled
1) Ports module clock is disabled |

6.6.3.2 Peripheral Power Management Register Low (PPMRL)

Figure 6-7. Peripheral Power Management Register Low

	31	30	29	28	27	26	25	24
Field	—	—	—	—	—	—	—	—
Reset	0000_0000							
R/W	R/W							
	23	22	21	20	19	18	17	16
Field	—	—	—	—	—	—	CDINTC0	CDTMR3
Reset	0000_0000							
R/W	R/W							
	15	14	13	12	11	10	9	8
Field	CDTMR2	CDTMR1	CDTMR0	—	—	CDQSPI	CDI2C	—
Reset	0000_0000							
R/W	R/W							
	7	6	5	4	3	2	1	0
Field	CDUART2	CDUART1	CDUART0	CDDMA	CDEIM	—	CDG	—
Reset	0000_1000							
R/W	R/W							
Address	Base + 18							
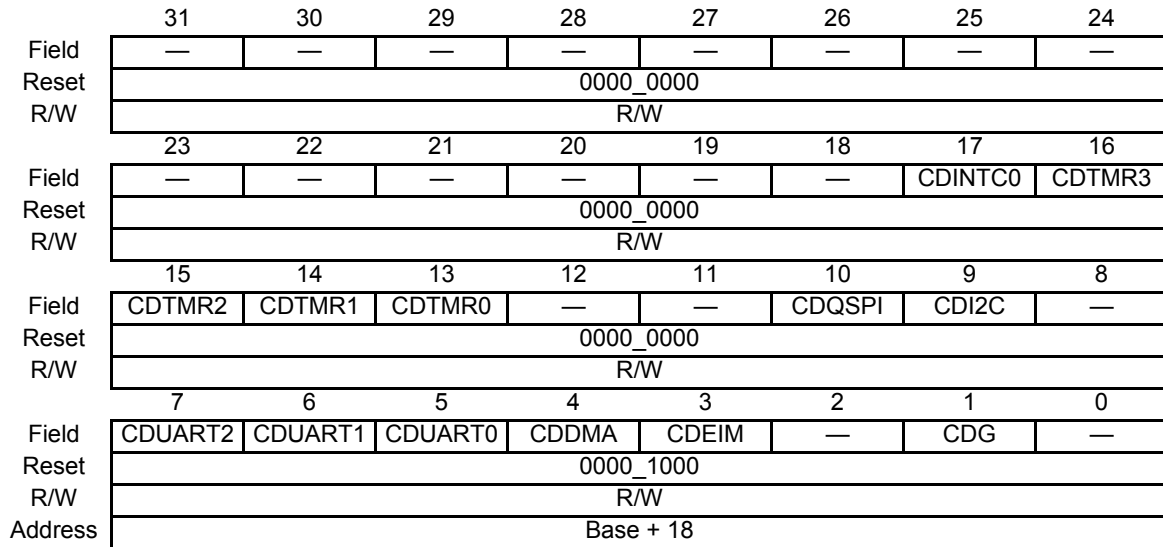
Bit(s)	Name	Description
31–18	—	Reserved, should be cleared.
17	CDINTC0	Disable clock to the INTC0 module. 0) INTC0 module clock is enabled 1) INTC0 module clock is disabled
16	CDTMR3	Disable clock to the TMR3 module. 0) TMR3 module clock is enabled 1) TMR3 module clock is disabled
15	CDTMR2	Disable clock to the TMR2 module. 0) TMR2 module clock is enabled 1) TMR2 module clock is disabled
14	CDTMR1	Disable clock to the TMR1 module. 0) TMR1 module clock is enabled 1) TMR1 module clock is disabled
13	CDTMR0	Disable clock to the TMR0 module. 0) TMR0 module clock is enabled 1) TMR0 module clock is disabled
12–11	—	Reserved, should be cleared.

10	CDQSPI	Disable clock to the QSPI module. 0) QSPI module clock is enabled 1) QSPI module clock is disabled
9	CDI2C	Disable clock to the I2C module. 0) I2C module clock is enabled 1) I2C module clock is disabled
8	—	Reserved, should be cleared.
7	CDUART2	Disable clock to the UART2 module. 0) UART1 module clock is enabled 1) UART2 module clock is disabled
6	CDUART1	Disable clock to the UART1 module. 0) UART1 module clock is enabled 1) UART1 module clock is disabled
5	CDUART0	Disable clock to the UART0 module. 0) UART0 module clock is enabled 1) UART0 module clock is disabled
4	CDDMA	Disable clock to the DMA module. 0) DMA module clock is enabled 1) DMA module clock is disabled
3	CDEIM	Disable clock to the EIM module. 0) EIM module clock is enabled 1) EIM module clock is disabled
2	—	Reserved, should be cleared.
1	CDG	Disable clock to the Global off-platform modules. 0) Global off-platform module clocks are enabled 1) Global off-platform module clocks are disabled
0	—	Reserved, should be cleared.

6.7 Functional Description

This subsection provides a functional description of the clock module.

6.7.1 System Clock Modes

The system clock source and PLL mode (enabled/disabled) are determined during reset (see Table 27-10). The values of CLKMOD[1:0] (and XTAL if CLKMOD0 does not equal 1) are latched during reset and are of no importance after reset is negated. If CLKMOD1 or CLKMOD0 change during a reset other than power-on reset, the internal clocks may glitch as the system clock source is changed between external clock mode and PLL clock mode. Whenever CLKMOD1 or CLKMOD0 is changed in reset, an immediate loss-of-lock condition occurs.

Table 6-9 shows the clockout frequency to clockin frequency relationships for the possible system clock modes.

Table 6-9. Clock Out and Clock In Relationships

System Clock Mode	PLL Options[1]
Normal PLL clock mode	$f_{sys} = f_{ref} \times 2(MFD + 2)/2^{RFD}$
1:1 PLL clock mode	$f_{sys} = f_{ref}$
External clock mode	$f_{sys} = f_{ref}$

[1] f_{ref} = input reference frequency
    f_{sys} = CLKOUT frequency
    MFD ranges from 0 to 7.
    RFD ranges from 0 to 7.

The external clock is divided by two internally to produce the system clocks.

6.7.2 Clock Operation During Reset

In external clock mode, the system is static and does not recognize reset until a clock is generated from the reference clock source selected by the CLKMOD pins (see Section 6.5.4, "CLKMOD[1:0]").

In PLL mode, the PLL operates in self-clocked mode (SCM) during reset until the input reference clock to the PLL begins operating within the limits given in the electrical specifications.

If a PLL failure causes a reset, the system enters reset using the reference clock. Then the system clock source changes to the PLL operating in SCM. If SCM is not functional, the system becomes static. Alternately, if the LOCEN bit in SYNCR is cleared when the PLL fails, the system becomes static. If external reset is asserted, the system cannot enter reset unless the PLL is capable of operating in SCM.

6.7.3 System Clock Generation

In normal PLL clock mode, the default system frequency is two times the reference frequency after reset. The RFD[2:0] and MFD[2:0] bits in the SYNCR select the frequency multiplier. The LPD[3:0] field in the LPCR register provides additional settings for dividing down the system clock (including when the PLL is disabled) for low power operation.

When programming the PLL, do not exceed the maximum system clock frequency listed in the electrical specifications. Use this procedure to accommodate the frequency overshoot that occurs when the MFD bits are changed:

1. Determine the appropriate value for the MFD and RFD fields in the SYNCR. The amount of jitter in the system clocks can be minimized by selecting the maximum MFD factor that can be paired with an RFD factor to provide the required frequency.

2. Write a value of RFD (from step 1) + 1 to the RFD field of the SYNCR.

3. Write the MFD value from step 1 to the SYNCR.

4. Monitor the LOCK flag in SYNSR. When the PLL achieves lock, write the RFD value from step 1 to the RFD field of the SYNCR. This changes the system clocks frequency to the required frequency.

NOTE

Keep the maximum system clock frequency below the limit given in the Electrical Characteristics.

6.7.4 PLL Operation

In PLL mode, the PLL synthesizes the system clocks. The PLL can multiply the reference clock frequency by 4x to 18x, provided that the system clock frequency remains within the range listed in electrical specifications. For example, if the reference frequency is 2 MHz, the PLL can synthesize frequencies of 8 MHz to 36 MHz. In addition, the RFD can reduce the system frequency by dividing the output of the PLL. The RFD is not in the feedback loop of the PLL, so changing the RFD divisor does not affect PLL operation.

Figure 6-8 shows the external support circuitry for the crystal oscillator with example component values. Actual component values depend on crystal specifications.

The following subsections describe each major block of the PLL. Refer to Figure 6-8 to see how these functional sub-blocks interact.
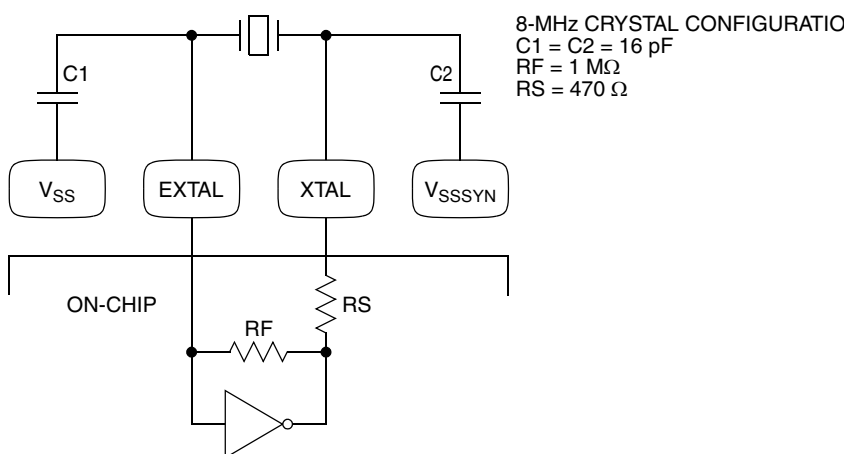
Figure 6-8. Crystal Oscillator Example

6.7.4.1 Phase and Frequency Detector (PFD)

The PFD is a dual-latch phase-frequency detector. It compares both the phase and frequency of the reference and feedback clocks. The reference clock comes from either the crystal oscillator or an external clock source.

The feedback clock comes from one of the following:
- CLKOUT in 1:1 PLL mode
- VCO output divided by two if CLKOUT is disabled in 1:1 PLL mode
- VCO output divided by the MFD in normal PLL mode

When the frequency of the feedback clock equals the frequency of the reference clock, the PLL is frequency-locked. If the falling edge of the feedback clock lags the falling edge of the reference clock, the PFD pulses the UP signal. If the falling edge of the feedback clock leads the falling edge of the reference clock, the PFD pulses the DOWN signal. The width of these pulses relative to the reference clock depends on how much the two clocks lead or lag each other. Once phase lock is achieved, the PFD continues to pulse the UP and DOWN signals for very short durations during each reference clock cycle. These short pulses continually update the PLL and prevent the frequency drift phenomenon known as dead-banding.

6.7.4.2 Charge Pump/Loop Filter

In 1:1 PLL mode, the charge pump uses a fixed current. In normal mode the current magnitude of the charge pump varies with the MFD as shown in Table 6-10.

Table 6-10. Charge Pump Current and MFD in Normal Mode Operation

Charge Pump Current	MFD
1X	$0 \leq \text{MFD} < 2$
2X	$2 \leq \text{MFD} < 6$
4X	$6 \leq \text{MFD}$

The UP and DOWN signals from the PFD control whether the charge pump applies or removes charge, respectively, from the loop filter. The filter is integrated on the chip.

6.7.4.3 Voltage Control Output (VCO)

The voltage across the loop filter controls the frequency of the VCO output. The frequency-to-voltage relationship (VCO gain) is positive, and the output frequency is four times the target system frequency.

6.7.4.4 Multiplication Factor Divider (MFD)

When the PLL is not in 1:1 PLL mode, the MFD divides the output of the VCO and feeds it back to the PFD. The PFD controls the VCO frequency via the charge pump and loop filter such that the reference and feedback clocks have the same frequency and phase. Thus, the frequency of the input to the MFD, which is also the output of the VCO, is the reference frequency multiplied by the same amount that the MFD divides by. For example, if the MFD divides the VCO frequency by six, the PLL is frequency locked when the VCO frequency is six times the reference frequency. The presence of the MFD in the loop allows the PLL to perform frequency multiplication, or synthesis.

In 1:1 PLL mode, the MFD is bypassed, and the effective multiplication factor is one.

6.7.4.5 PLL Lock Detection

The lock detect logic monitors the reference frequency and the PLL feedback frequency to determine when frequency lock is achieved. Phase lock is inferred by the frequency relationship, but is not guaranteed. The LOCK flag in the SYNSR reflects the PLL lock status. A sticky lock flag, LOCKS, is also provided.

The lock detect function uses two counters: one is clocked by the reference, and the other is clocked by the PLL feedback. When the reference counter has counted N cycles, its count is compared to that of the feedback counter. If the feedback counter has also counted N cycles, the process is repeated for $N + K$ counts. Then, if the two counters still match, the lock criteria is relaxed by 1/2 and the system is notified that the PLL has achieved frequency lock.

After lock is detected, the lock circuit continues to monitor the reference and feedback frequencies using the alternate count and compare process. If the counters do not match at any comparison time, then the LOCK flag is cleared to indicate that the PLL has lost lock. At this point, the lock criteria is tightened and the lock detect process is repeated.

The alternate count sequences prevent false lock detects due to frequency aliasing while the PLL tries to lock. Alternating between tight and relaxed lock criteria prevents the lock detect function from randomly toggling between locked and non-locked status due to phase sensitivities. Figure 6-9 shows the sequence for detecting locked and non-locked conditions.

In external clock mode, the PLL is disabled and cannot lock.
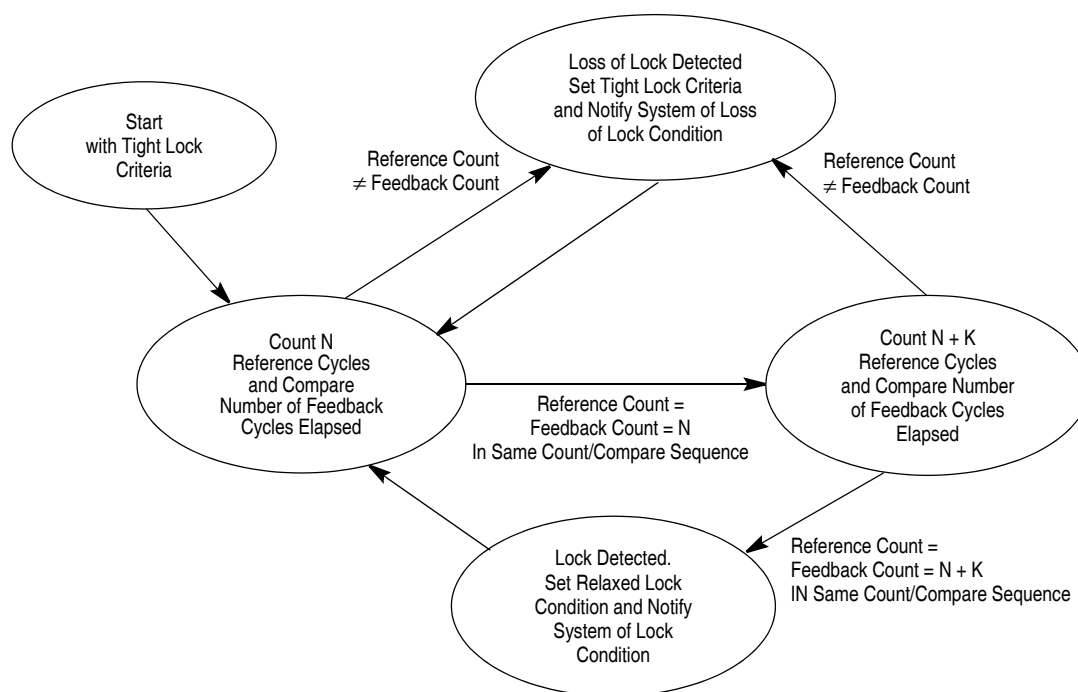
Figure 6-9. Lock Detect Sequence

6.7.4.6 PLL Loss of Lock Conditions

Once the PLL acquires lock after reset, the LOCK and LOCKS flags are set. If the MFD is changed, or if an unexpected loss of lock condition occurs, the LOCK and LOCKS flags are negated. While the PLL is in the non-locked condition, the system clocks continue to be sourced from the PLL as the PLL attempts to relock. Consequently, during the relocking process, the system clocks frequency is not well defined and may exceed the maximum system frequency, violating the system clock timing specifications.

However, once the PLL has relocked, the LOCK flag is set. The LOCKS flag remains cleared if the loss of lock was unexpected. The LOCKS flag is set when the loss of lock is caused by changing MFD. If the PLL is intentionally disabled during stop mode, then after exit from stop mode, the LOCKS flag reflects the value prior to entering stop mode once lock is regained.

6.7.4.7 PLL Loss of Lock Reset

If the LOLRE bit in the SYNCR is set, a loss of lock condition asserts reset. Reset reinitializes the LOCK and LOCKS flags. Therefore, software must read the LOL bit in the reset status register (RSR) to determine if a loss of lock caused the reset. See Section 29.4.2, "Reset Status Register (RSR)."

To exit reset in PLL mode, the reference must be present, and the PLL must achieve lock.

In external clock mode, the PLL cannot lock. Therefore, a loss of lock condition cannot occur, and the LOLRE bit has no effect.

6.7.4.8 Loss of Clock Detection

The LOCEN bit in the SYNCR enables the loss of clock detection circuit to monitor the input clocks to the phase and frequency detector (PFD). When either the reference or feedback clock frequency falls below the minimum frequency, the loss of clock circuit sets the sticky LOCS flag in the SYNSR.

NOTE

In external clock mode, the loss of clock circuit is disabled.

6.7.4.9 Loss of Clock Reset

The clock module can assert a reset when a loss of clock or loss of lock occurs. When a loss-of-clock condition is recognized, reset is asserted if the LOCRE bit in SYNCR is set. The LOCS bit in SYNSR is cleared after reset. Therefore, the LOC bit must be read in RSR to determine that a loss of clock condition occurred. LOCRE has no effect in external clock mode.

To exit reset in PLL mode, the reference must be present, and the PLL must acquire lock.

Reset initializes the clock module registers to a known startup state as described in Section 6.6, "Memory Map and Registers."

6.7.4.10 Alternate Clock Selection

Depending on which clock source fails, the loss-of-clock circuit switches the system clocks source to the remaining operational clock. The alternate clock source generates the system clocks until reset is asserted. As Table 6-11 shows, if the reference fails, the PLL goes out of lock and into self-clocked mode (SCM). The PLL remains in SCM until the next reset. When the PLL is operating in SCM, the system frequency depends on the value in the RFD field. The SCM system frequency stated in electrical specifications assumes that the RFD has been programmed to binary 000. If the loss-of-clock condition is due to PLL failure, the PLL reference becomes the system clocks source until the next reset, even if the PLL regains and relocks.

Table 6-11. Loss of Clock Summary

Clock Mode	System Clock Source Before Failure	Reference Failure Alternate Clock Selected by LOC Circuit[1] Until Reset	PLL Failure Alternate Clock Selected by LOC Circuit Until Reset
PLL	PLL	PLL self-clocked mode	PLL reference
External	External clock	None	NA

[1] The LOC circuit monitors the reference and feedback inputs to the PFD. See Figure 6-8.

A special loss-of-clock condition occurs when both the reference and the PLL fail. The failures may be simultaneous, or the PLL may fail first. In either case, the reference clock failure takes priority and the PLL attempts to operate in SCM. If successful, the PLL remains in SCM until the next reset. If the PLL cannot operate in SCM, the system remains static until the next reset. Both the reference and the PLL must be functioning properly to exit reset.

6.7.4.11 Loss of Clock in Stop Mode

Table 6-12 shows the resulting actions for a loss of clock in stop mode when the device is being clocked by the various clocking methods.

Table 6-12. Stop Mode Operation (Sheet 1 of 5)

MODE In	LOCEN	LOCRE	LOLRE	PLL	OSC	FWKUP	Expected PLL Action at Stop	PLL Action During Stop	MODE Out	LOCKSS	LOCK	LOCS	Comments
EXT	X	X	X	X	X	X	—	—	EXT	0	0	0	
								Lose reference clock	Stuck	—	—	—	
NRM	0	0	0	Off	Off	0	Lose lock, f.b. clock, reference clock	Regain	NRM	'LK	1	'LC	
								No regain	Stuck	—	—	—	
NRM	X	0	0	Off	Off	1	Lose lock, f.b. clock, reference clock	Regain clocks, but don't regain lock	SCM–> unstable NRM	0–>'LK	0–>1	1–>'LC	Block LOCS and LOCKS until clock and lock respectively regain; enter SCM regardless of LOCEN bit until reference regained
								No reference clock regain	SCM–>	0–>	0–>	1–>	Block LOCS and LOCKS until clock and lock respectively regain; enter SCM regardless of LOCEN bit
								No f.b. clock regain	Stuck	—	—	—	
NRM	0	0	0	Off	On	0	Lose lock	Regain	NRM	'LK	1	'LC	Block LOCKS from being cleared
								Lose reference clock or no lock regain	Stuck	—	—	—	
								Lose reference clock, regain	NRM	'LK	1	'LC	Block LOCKS from being cleared
NRM	0	0	0	Off	On	1	Lose lock	No lock regain	Unstable NRM	0–>'LK	0–>1	'LC	Block LOCKS until lock regained
								Lose reference clock or no f.b. clock regain	Stuck	—	—	—	
								Lose reference clock, regain	Unstable NRM	0–>'LK	0–>1	'LC	LOCS not set because LOCEN = 0

Table 6-12. Stop Mode Operation (Sheet 2 of 5)

MODE In	LOCEN	LOCRE	LOLRE	PLL	OSC	FWKUP	Expected PLL Action at Stop	PLL Action During Stop	MODE Out	LOCKSS	LOCK	LOCS	Comments
NRM	0	0	0	On	On	0	—	—	NRM	'LK	1	'LC	
								Lose lock or clock	Stuck	—	—	—	
								Lose lock, regain	NRM	0	1	'LC	
								Lose clock and lock, regain	NRM	0	1	'LC	LOCS not set because LOCEN = 0
NRM	0	0	0	On	On	1	—	—	NRM	'LK	1	'LC	
								Lose lock	Unstable NRM	0	0–>1	'LC	
								Lose lock, regain	NRM	0	1	'LC	
								Lose clock	Stuck	—	—	—	
								Lose clock, regain without lock	Unstable NRM	0	0–>1	'LC	
								Lose clock, regain with lock	NRM	0	1	'LC	
NRM	X	X	1	Off	X	X	Lose lock, f.b. clock, reference clock	RESET	RESET	—	—	—	Reset immediately
NRM	0	0	1	On	On	X	—	—	NRM	'LK	1	'LC	
								Lose lock or clock	RESET	—	—	—	Reset immediately
NRM	1	0	0	Off	Off	0	Lose lock, f.b. clock, reference clock	Regain	NRM	'LK	1	'LC	REF not entered during stop; SCM entered during stop only during oscillator startup
								No regain	Stuck	—	—	—	
NRM	1	0	0	Off	On	0	Lose lock, f.b. clock	Regain	NRM	'LK	1	'LC	REF mode not entered during stop
								No f.b. clock or lock regain	Stuck	—	—	—	
								Lose reference clock	SCM	0	0	1	Wakeup without lock

Table 6-12. Stop Mode Operation (Sheet 3 of 5)

MODE In	LOCEN	LOCRE	LOLRE	PLL	OSC	FWKUP	Expected PLL Action at Stop	PLL Action During Stop	MODE Out	LOCKSS	LOCK	LOCS	Comments
NRM	1	0	0	Off	On	1	Lose lock, f.b. clock	Regain f.b. clock	Unstable NRM	0–>'LK	0–>1	'LC	REF mode not entered during stop
								No f.b. clock regain	Stuck	—	—	—	
								Lose reference clock	SCM	0	0	1	Wakeup without lock
NRM	1	0	0	On	On	0	—	—	NRM	'LK	1	'LC	
								Lose reference clock	SCM	0	0	1	Wakeup without lock
								Lose f.b. clock	REF	0	X	1	Wakeup without lock
								Lose lock	Stuck	—	—	—	
								Lose lock, regain	NRM	0	1	'LC	
NRM	1	0	0	On	On	1	—	—	NRM	'LK	1	'LC	
								Lose reference clock	SCM	0	0	1	Wakeup without lock
								Lose f.b. clock	REF	0	X	1	Wakeup without lock
								Lose lock	Unstable NRM	0	0–>1	'LC	
NRM	1	0	1	On	On	X	—	—	NRM	'LK	1	'LC	
								Lose lock or clock	RESET	—	—	—	Reset immediately
NRM	1	1	X	Off	X	X	Lose lock, f.b. clock, reference clock	RESET	RESET	—	—	—	Reset immediately
NRM	1	1	0	On	On	0	—	—	NRM	'LK	1	'LC	
								Lose clock	RESET	—	—	—	Reset immediately
								Lose lock	Stuck	—	—	—	
								Lose lock, regain	NRM	0	1	'LC	

Table 6-12. Stop Mode Operation (Sheet 4 of 5)

MODE In	LOCEN	LOCRE	LOLRE	PLL	OSC	FWKUP	Expected PLL Action at Stop	PLL Action During Stop	MODE Out	LOCKSS	LOCK	LOCS	Comments
NRM	1	1	0	On	On	1	—	—	NRM	'LK	1	'LC	
								Lose clock	RESET	—	—	—	Reset immediately
								Lose lock	Unstable NRM	0	0–>1	'LC	
								Lose lock, regain	NRM	0	1	'LC	
NRM	1	1	1	On	On	X	—	—	NRM	'LK	1	'LC	
								Lose clock or lock	RESET	—	—	—	Reset immediately
REF	1	0	0	X	X	X	—	—	REF	0	X	1	
								Lose reference clock	Stuck	—	—	—	
SCM	1	0	0	Off	X	0	PLL disabled	Regain SCM	SCM	0	0	1	Wakeup without lock
SCM	1	0	0	Off	X	1	PLL disabled	Regain SCM	SCM	0	0	1	
SCM	1	0	0	On	On	0	—	—	SCM	0	0	1	Wakeup without lock
								Lose reference clock	SCM				

Table 6-12. Stop Mode Operation (Sheet 5 of 5)

MODE In	LOCEN	LOCRE	LOLRE	PLL	OSC	FWKUP	Expected PLL Action at Stop	PLL Action During Stop	MODE Out	LOCKSS	LOCK	LOCS	Comments
SCM	1	0	0	On	On	1	—	—	SCM	0	0	1	
								Lose reference clock	SCM				

Note:

PLL = PLL enabled during STOP mode. PLL = On when STPMD[1:0] = 00 or 01

OSC = oscillator enabled during STOP mode. Oscillator is on when STPMD[1:0] = 00, 01, or 10

MODES

NRM = normal PLL crystal clock reference or normal PLL external reference or PLL 1:1 mode. During PLL 1:1 or normal external reference mode, the oscillator is never enabled. Therefore, during these modes, refer to the OSC = On case regardless of STPMD values.

EXT = external clock mode

REF = PLL reference mode due to losing PLL clock or lock from NRM mode

SCM = PLL self-clocked mode due to losing reference clock from NRM mode

RESET = immediate reset

LOCKS

'LK -= expecting previous value of LOCKS before entering stop

0–>'LK = current value is 0 until lock is regained which then will be the previous value before entering stop

0–> = current value is 0 until lock is regained but lock is never expected to regain

LOCS

'LC = expecting previous value of LOCS before entering stop

1–>'LC = current value is 1 until clock is regained which then will be the previous value before entering stop

1–> = current value is 1 until clock is regained but CLK is never expected to regain

Chapter 7
Power Management

7.1 Introduction

This chapter explains the low-power operation of the device.

7.1.1 Features

The following features support low-power operation.

- Four modes of operation: Run, Wait, Doze, and Stop
- Ability to shut down most peripherals independently
- Ability to shut down clocks to most peripherals independently
- Ability to run the device in low-frequency LIMP mode
- Ability to shut down the external FB_CLK pin

7.2 Memory Map/Register Definition

The power management programming model consists of registers from both the SCM and CCM memory space, as shown below:

Table 7-1. Power Management Memory Map

Address	Register	Access	Reset Value	Section/Page
Supervisor Access Only Registers[1]				
0xFC04_0013	Wakup Control Register (WCR)	R/W	0x00	7.2.1/7-2
0xFC04_002C	Peripheral Power Management Set Register 0 (PPMSR0)	W	0x00	7.2.2/7-3
0xFC04_002D	Peripheral Power Management Clear Register 0 (PPMCR0)	W	0x00	7.2.3/7-4
0xFC04_002E	Peripheral Power Management Set Register 1 (PPMSR1)	W	0x00	7.2.2/7-3
0xFC04_002F	Peripheral Power Management Clear Register 1 (PPMCR1)	W	0x00	7.2.3/7-4
0xFC04_0030	Peripheral Power Management High Register 0 (PPMHR0)	R/W	0x0000_0000	7.2.4/7-4
0xFC04_0034	Peripheral Power Management Low Register 0 (PPMLR0)	R/W	0x0000_0000	7.2.4/7-4
0xFC04_0038	Peripheral Power Management High Register 1 (PPMHR1)	R/W	0x0000_0000	7.2.4/7-4
0xFC0A_0007	Low-Power Control Register (LPCR)	R/W	0x00	7.2.5/7-6
0xFC0A_0010	Miscellaneous Control Register (MISCCR)	R/W	See Section	7.2.6/7-7

[1] User access to supervisor only address locations have no effec t and result in a cycle termination transfer error

7.2.1 Wake-up Control Register

Implementation of low-power stop mode and exit from a low-power mode via an interrupt require communication between the core and logic associated with the interrupt controller. The WCR is an 8-bit register that enables entry into low-power stop mode, and includes the setting of the interrupt level needed to exit a low-power mode.

NOTE

The setting of the low-power mode select field, LPCR[LPMD], determines which low-power mode the device enters when a STOP instruction is issued.

If this field is set to enter stop mode, then the WCR[ENBWCR] bit must also be set.

The following sequence of operations is generally needed to enable this functionality.

1. The WCR register is programmed, setting the ENBWCR bit and the desired interrupt priority level.
2. At the appropriate time, the processor executes the privileged STOP instruction. Once the processor has stopped execution, it asserts a specific Processor Status (PST) encoding. Issuing the STOP instruction when the WCR[ENBWCR] bit is set causes the SCM to enter stop mode.
3. The entry into a low-power mode is processed by the low-power mode control logic, and the appropriate clocks (usually those related to the high-speed processor core) are disabled.
4. After entering the low-power mode, the interrupt controller enables a combinational logic path which evaluates any unmasked interrupt requests. The device waits for an event to generate an interrupt request with a priority level greater than the value programmed in WCR[PRILVL].
5. Once an appropriately high interrupt request level arrives, the interrupt controller signals its presence, and the SCM responds by asserting the request to exit low-power mode.
6. The low-power mode control logic senses the request signal and re-enables the appropriate clocks.
7. With the processor clocks enabled, the core processes the pending interrupt request.

Address: 0x13 (WCR)   Access: Supervisor Read/Write

	7	6	5	4	3	2	1	0
R	ENBWCR	0	0	0	0	PRILVL		
W								
Reset:	0	0	0	0	0	0	0	0
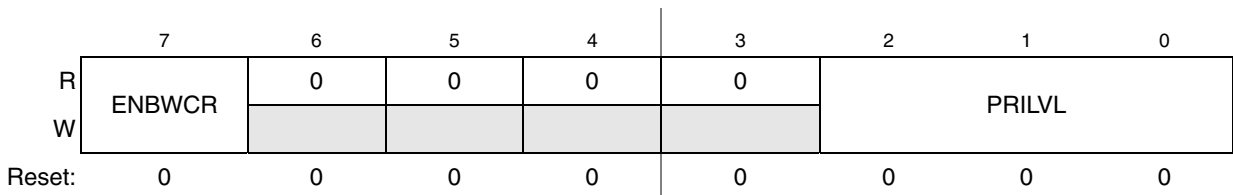
Figure 7-1. Wake-up Control Register (WCR)

Table 7-2. WCR Field Descriptions

Field	Description
7 ENBWCR	Enable wake-up control. 0 Wake-up control is disabled 1 Wake-up control is enabled

Preliminary

Table 7-2. WCR Field Descriptions (continued)

Field	Description
6–3	Reserved, should be cleared.
2–0 PRILVL	Exit low-power mode interrupt priority level. This field defines the interrupt priority level needed to exit the low-power mode.

PRILVL	Interrupt Level Needed to Exit Low-Power Mode
000	Any interrupt request exits low-power mode
001	Interrupt request levels [2-7] exit low-power mode
010	Interrupt request levels [3-7] exit low-power mode
011	Interrupt request levels [4-7] exit low-power mode
100	Interrupt request levels [5-7] exit low-power mode
101	Interrupt request levels [6-7] exit low-power mode
11x	Interrupt request level [7] exits low-power mode

7.2.2 Peripheral Power Management Set Registers (PPMSR0 & PPMSR1)

The PPMSR registers provide a simple mechanism to set a given bit in the PPMHR & PPMLR registers to disable the clock for a given peripheral module without the need to perform a read-modify write on the PPMR. The data value on a register write causes the corresponding bit in the PPM{H,L}R to be set. The SAMCD bit provides a global set function forcing the entire contents of the PPMR to be set, disabling all peripheral module clocks. Reads of these registers return all zeroes.

Address: 0x2C (PPMSR0)   Access: Supervisor Write-only
0x2E (PPMSR1)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W		SAMCD	SMCD					
Reset:	0	0	0	0	0	0	0	0
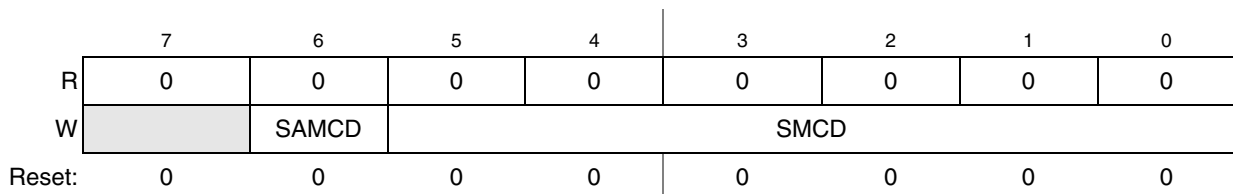
Figure 7-2. Peipheral Power Management Set Registers (PPMSR*n*)

Table 7-3. PPMSR*n* Field Descriptions

Field	Description
7	Reserved, should be cleared.
6 SAMCD	Set all module clock disables. 0 Set only those bits specified in the SMCD field 1 Set all bits in PPMRH and PPMRL, disabling all peripheral clocks
5–0 SMCD	Set module clock disable. Set the corresponding bit in PPMR{H,L}, disabling the peripheral clock.

7.2.3 Peripheral Power Management Clear Registers (PPMCR0 & PPMCR1)

The PPMCR registers provide a simple mechanism to clear a given bit in the PPMHR & PPMLR registers to enable the clock for a given peripheral module without the need to perform a read-modify write on the PPMR. The data value on a register write causes the corresponding bit in the PPM{H,L}R to be clear. A value of 64 to 127 provides a global clear function forcing the entire contents of the PPMR to be clear, enabling all peripheral module clocks. Reads of these registers return all zeroes.

Address: 0x2D (PPMCR0)   Access: Supervisor Write-only
            0x2F (PPMCR1)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W		CAMCD	CMCD					
Reset:	0	0	0	0	0	0	0	0
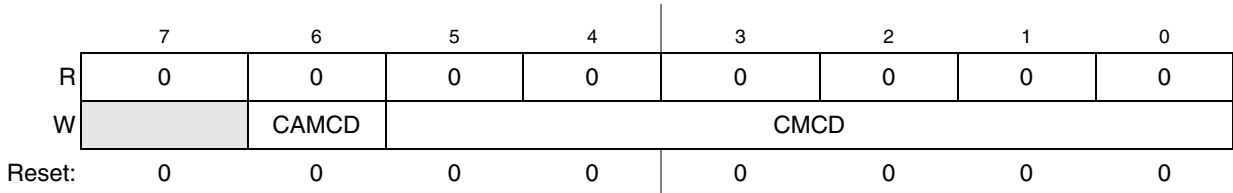
Figure 7-3. Peipheral Power Management Clear Registers (PPMCR*n*)

Table 7-4. PPMCR*n* Field Descriptions

Field	Description
7	Reserved, should be cleared.
6 CAMCD	Clear all module clock disables. 0 Clear only those bits specified in the CMCD field 1 Clear all bits in PPMRH and PPMRL, enabling all peripheral clocks
5–0 CMCD	Clear module clock disable. Clear the corresponding bit in PPMR{H,L}, enabling the peripheral clock.

7.2.4 Peripheral Power Management Registers (PPMHR0 & PPMLR0)

The PPMR registers provide a bit map for controlling the generation of the peripheral clocks for each decoded address space. Recall each peripheral module is mapped into 16 kByte slots within the memory map. The PPMR registers provide a unique control bit for each of these address spaces that defines whether the module clock for the given space is enabled or disabled.

Since the operation of the crossbar switch and the system control module (SCM) are fundamental to the operation of the device, the clocks for these modules cannot be disabled.

The individual bits of the PPMR can be modified using a read-modify-write to this register directly or indirectly through writes to the PPMSR and PPMCR registers to set/clear individual bits.

Address: 0x30 (PPMHR0)   Access: Supervisor Read/Write

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

Figure 7-4. Peipheral Power Management High Register (PPMHR)

Address: 0x34 (PPMLR0)   Access: Supervisor Read/Write

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

Figure 7-5. Peipheral Power Management Low Registers (PPMLR0)

Table 7-5. PPM Register Assignments

IPSBAR	Peripheral	
–	Off platform plobal space	PPMR[1]
–	Reserved	PPMR[2]
0x0080	EIM	PPMR[3]
0x0100	DMA	PPMR[4]
0x0200	UART0	PPMR[5]
0x0240	UART1	PPMR[6]
0x0280	UART2	PPMR[7]
0x02C0	Reserved	PPMR[8]
0x0300	I^2C	PPMR[9]
0x0340	QSPI	PPMR[10]
0x-380	Reserved	PPMR[11]
0x03C0	Reserved	PPMR[12]
0x0400	DMA Timer0	PPMR[13]
0x0440	DMA Timer1	PPMR[14]
0x0480	DMA Timer2	PPMR[15]
0x04C0	DMA Timer3	PPMR[16]
0x0C00	INTC0	PPMR[17]
0x0D00	Reserved	PPMR[18]
0x0E00	Reserved	PPMR[19]

Preliminary

Table 7-5. PPM Register Assignments (continued)

IPSBAR	Peripheral	
0xFC05_0000	Reserved	PPMR[20]
0x1000	Reserved	PPMR[21]
–	Reserved	PPMR[22]
–	Reserved	PPMR[23]
–	Reserved	PPMR[24]
–	Reserved	PPMR[25]
–	Reserved	PPMR[26]
–	Reserved	PPMR[27]
–	Reserved	PPMR[28]
–	Reserved	PPMR[29]
–	Reserved	PPMR[30]
–	Reserved	PPMR[31]
0x0010_0000 – 0xFC0F_FFFF	Off-platform Slaves	PPMR[32-47]
0xFC10_0000 – 0xFFFF-FFFF	63 MByte Off-platform global space	PPMF[1]

7.2.5   Low-Power Control Register (LPCR)

The LPCR register controls chip operation and module operation during low-power modes.

Address: 0x07 (LPCR)   Access: Supervisor Read/Write

	7	6	5	4	3	2	1	0
R	LPMD		FWKUP	STPMD		0	0	0
W								
Reset:	0	0	0	0	0	0	0	0
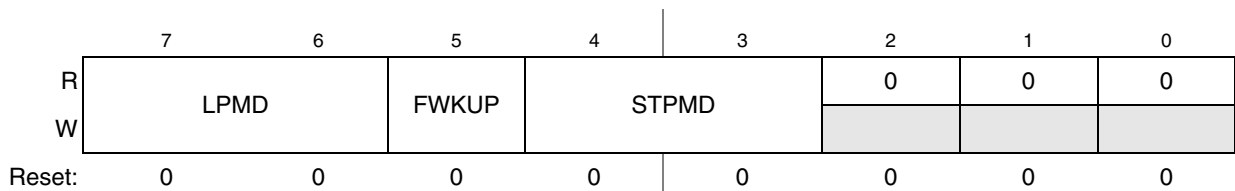
Figure 7-6. Low-Power Control Register (LPCR)

Table 7-6. LPCR Field Descriptions

Field	Description
7–6 LPMD	Low-power mode select. Used to select the low-power mode the chip enters once the ColdFire core executes the STOP instruction. These bits must be written prior to instruction execution for them to take effect. The LPMD bits are readable and writable in all modes. Below illustrates the four different power modes that can be configured with the LPMD bit field. 00 RUN 01 DOZE 10 WAIT 11 STOP **Note:** If LPCR[LPMD] is cleared, then the device will stop executing code upon issue of a STOP instruction. However, no clocks will be disabled.
5 FWKUP	Fast wake-up. Determines whether the system clocks are enabled upon wake-up from stop mode. This bit must be written before execution of the STOP instruction for it to take effect. 0 System clocks enabled only when PLL is locked or operating normally. 1 System clocks enabled upon wake-up from stop mode, regardless of PLL lock status. **Note:** If FWKUP is set before entering stop mode, it should not be cleared upon wake-up from stop mode until after the PLL has actually acquired lock. Lock status may be obtained by reading the MISCCR[PLLLOCK] bit. Also note that FWKUP is not effective in limp mode, since the PLL never locks in this mode. The system clocks are always enabled upon wake-up from stop mode, regardless of the value of FWKUP.
4–3 STPMD	FB_CLK stop mode bits. Controls the operation of the clocks, PLL, and oscillator in stop mode as shown below. | STPMD	System Clocks	FB_CLK	PLL	Oschillator
---	---	---	---	---
00	Disabled	Enabled	Enabled	Enabled
01	Disabled	Disabled	Enabled	Enabled
10	Disabled	Disabled	Disabled	Enabled
11	Disabled	Disabled	Disabled	Disabled	
2–0	Reserved, should be cleared.

7.2.6 Miscellaneous Control Register (MISCCR)

The MISCCR provides clock source selection and configuration for internal clocks.

Address: 0x10 (MISCCR)   Access: Supervisor read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	PLL LOCK	LIMP	0	0	0	0	0	0	0	0			LPDDIV	
W																
Reset	0	0	1	See Note	0	0	0	0	0	0	0	0	0	0	0	0

Note: Reset value depends upon chosen reset configuration (RCON[RLIMP]).

Figure 7-7. Miscellaneous Control Register (MISCCR)

Table 7-7. MISCCR Field Descriptions

Field	Description
15–14	Reserved, should be cleared.
13 PLLLOCK	PLL lock status. 0 PLL is not locked. 1 PLL is locked.
12 LIMP	Limp mode enable. Selects between the PLL and the low-power clock divider as the source of all system clocks. 0 Normal operation; PLL drives all internal clocks. 1 Limp mode; low-power clock divider drivers internal clocks.
11–4	Reserved, should be cleared.
3–0 LPDDIV	Low power divider. Specifies the limp mode divide value used to produce the clock for the ColdFire core, bus, and other system clocks. This field is used only when LIMP is set, and ignored otherwise. Limp mode clock = Oscillator clock / 2^{LPDIV}

7.3   Functional Description

The functions and characteristics of the low-power modes, and how each module is affected by, or affects these modes are discussed in this section.

7.3.1   Peripheral Shut Down

All peripherals, except for the SCM and crossbar switch, may have their input clocks individually removed by software in order to reduce power consumption. See Section 7.2.4, "Peripheral Power Management Registers (PPMHR0 & PPMLR0)," for more information. A peripheral may be disabled at any time and will remain disabled during any low-power mode of operation.

7.3.2   LIMP mode

The device may also be booted into a low-frequency limp mode, in which the PLL is bypassed and the device runs from a factor of the input clock (EXTAL). In this mode, EXTAL feeds a 5-bit programmable counter that divides the input clock by 2^n, where n is the value of the programmable counter field, MISCCR[LPDIV]. The programmed value of the divider may be changed without glitches or otherwise negative affects to the system. While in this mode, the PLL is placed in bypass mode to reduce overall system power consumption. When switching from LIMP mode to normal functional mode, the user must wait for the PLL to lock before continuing with code execution.

LIMP mode may also be entered and exited from by writing to the MISCCR[LIMP] bit.

While in this mode a 2:1 ratio is maintained between the core and the primary bus clock.

7.3.3   Low-Power Modes

The system enters a low-power mode by executing a STOP instruction. Which mode the device actually enters (either stop, wait, or doze) depends on what the user programs into LPCR[LPMD]. Entry into any

of these modes idles the CPU with no cycles active, powers down the system, and stops all internal clocks appropriately. During stop mode, the system clock is stopped low.

A wake-up event is required to exit a low-power mode and return to run mode. Wake-up events consist of any of these conditions:

- Any type of reset
- Any valid, enabled interrupt request

Exiting from low power mode via an interrupt request requires:

- An interrupt request whose priority is higher than the value programmed in the WCR[PRILVL].
- An interrupt request whose priority is higher than the value programmed in the interrupt priority mask (I) field of the core's status register.
- An interrupt request from a source which is not masked in the interrupt controller's interrupt mask register.
- An interrupt request which has been enabled at the module of the interrupt's origin.

7.3.3.1 Run Mode

Run mode is the normal system operating mode. Current consumption in this mode is related directly to the system clock frequency.

7.3.3.2 Wait Mode

Wait mode is intended to be used to stop only the CPU and memory clocks until a wake-up event is detected. In this mode, peripherals may be programmed to continue operating and can generate interrupts, which cause the CPU to exit from wait mode.

7.3.3.3 Doze Mode

Doze mode affects the CPU in the same manner as wait mode, except that each peripheral defines individual operational characteristics in doze mode. Peripherals which continue to run and have the capability of producing interrupts may cause the CPU to exit the doze mode and return to run mode. Peripherals which are stopped will restart operation on exit from doze mode as defined for each peripheral.

7.3.3.4 Stop Mode

Stop mode affects the CPU in the same manner as the wait and doze modes, except that all clocks to the system are stopped and the peripherals cease operation.

Stop mode must be entered in a controlled manner to ensure that any current operation is properly terminated. When exiting stop mode, most peripherals retain their pre-stop status and resume operation.

The following subsections specify the operation of each module while in and when exiting low-power modes.

7.3.4 Peripheral Behavior in Low-Power Modes

7.3.4.1 ColdFire Core

The ColdFire core is disabled during any low-power mode. No recovery time is required when exiting any low-power mode.

7.3.4.2 Static Random-Access Memory (SRAM)

SRAM is disabled during any low-power mode. No recovery time is required when exiting any low-power mode.

7.3.4.3 Clock Module

In wait and doze modes, the clocks to the CPU and SRAM will be stopped and the system clocks to the peripherals are enabled. Each module may disable the module clocks locally at the module level. In stop mode, all clocks to the system will be stopped.

During stop mode, there are several options for enabling/disabling the PLL and/or crystal oscillator (OSC); each of these options requires a compromise between wake-up recovery time and stop mode power. The PLL may be disabled during stop mode. A wake-up time of up to 200 μs is required for the PLL to re-lock. The OSC may also be disabled during stop mode. The time required for the OSC to restart is dependent upon the startup time of the crystal used. Power consumption can be reduced in stop mode by disabling either or both of these functions via the SYNCR[STMPD] bits.

The external FB_CLK signal may be enabled during low-power stop (if the PLL is still enabled or bypassed) to support systems using this signal as the clock source.

The system clocks may be enabled during wake-up from stop mode without waiting for the PLL to lock. This eliminates the wake-up recovery time, but at the risk of sending a potentially unstable clock to the system. It is recommended, if this option is used, that the PLL frequency divider is set so that the targeted system frequency is no more than half the maximum allowed. This will allow for any frequency overshoot of the PLL while still keeping the system clock within specification.

In external clock mode, there are no wait times for the OSC startup or PLL lock.

The external FB_CLK output pin may be disabled in the low state to lower power consumption via the SYNCR[DISCLK] bit. The external FB_CLK pin function is enabled by default at reset.

7.3.4.4 Chip Configuration Module

The chip configuration module is unaffected by entry into a low-power mode. If low-power mode is exited by a reset, chip configuration may be executed if configured to do so.

7.3.4.5 Reset Controller

A power-on reset (POR) will always cause a chip reset and exit from any low-power mode.

In wait and doze modes, asserting the external $\overline{\text{RSTI}}$ pin for at least four clocks will cause an external reset that will reset the chip and exit any low-power modes.

In stop mode, the $\overline{\text{RSTI}}$ pin synchronization is disabled and asserting the external $\overline{\text{RSTI}}$ pin will asynchronously generate an internal reset and exit any low-power modes. Registers will lose current values and must be reconfigured from reset state if needed.

If the phase lock loop (PLL) in the clock module is active and if the LOLRE bit in the synthesizer control register are set, then any loss-of-lock will reset the device and exit any low-power modes.

If the core or on-chip watchdog timer is still enabled during wait or doze modes, then a watchdog timer timeout may generate a reset to exit these low-power modes.

When the CPU is inactive, a software reset cannot be generated to exit any low-power mode.

7.3.4.6 System Control Module (SCM)

The SCM's core watchdog timer can bring the device out of all low-power modes except stop mode. In stop mode all clocks stop, and the core watchdog timer does not operate.

When enabled, the core watchdog can bring the device out of low-power mode in one of two ways. Depending on the setting of the CWCR[CWRI] field, a core watchdog timeout may cause a reset of the device. Other settings of the CWRI field may enable a core watchdog interrupt and upon a watchdog timeout, this interrupt can bring the device out of low-power mode. This system setup must meet the conditions specified in Section 7.3.3, "Low-Power Modes," for the core watchdog interrupt to bring the part out of low-power mode.

7.3.4.7 GPIO Ports

The GPIO ports are unaffected by entry into a low-power mode. These pins may impact low-power current draw if they are configured as outputs and are sourcing current to an external load. If low-power mode is exited by a reset, the state of the I/O pins will revert to their default direction settings.

7.3.4.8 Interrupt Controllers (INTC0)

The interrupt controller is not affected by any of the low-power modes. All logic between the input sources and generating the interrupt to the processor will be combinational to allow the ability to wake up the core during low-power stop mode when all system clocks are stopped.

An interrupt request will cause the CPU to exit a low-power mode only if that interrupt's priority level is at or above the level programmed in the interrupt priority mask field of the CPU's status register (SR) and above the level programmed in the WCR[PRILVL]. The interrupt must also be enabled in the interrupt controller's interrupt mask register as well as at the module from which the interrupt request would originate.

7.3.4.9 Edge Port

In wait and doze modes, the edge port continues to operate normally and may be configured to generate interrupts (either an edge transition or low level on an external pin) to exit the low-power modes.

In stop mode, there is no system clock available to perform the edge detect function. Thus, only the level detect logic is active (if configured) to allow any low level on the external interrupt pin to generate an interrupt (if enabled) to exit the stop mode.

7.3.4.10 DMA Controller

In wait and doze modes, the DMA controller is capable of bringing the device out of a low-power mode by generating an interrupt either upon completion of a transfer or upon an error condition. The completion of transfer interrupt is generated when DMA interrupts are enabled by the setting of a DMA_INTR[INTn] bit, and an interrupt is generated when the TCDn[DONE] bit is set. The interrupt upon error condition is generated when the DMA_EEIR[EEIn] bit is set, and an interrupt is generated when any of the DMA_ESR bits becomes set.

The DMA controller is stopped in stop mode and thus cannot cause an exit from this low-power mode.

7.3.4.11 On-chip Watchdog Timer

In stop mode (or in wait/doze mode, if so programmed), the watchdog ceases operation and freezes at the current value. When exiting these modes, the watchdog resumes operation from the stopped value. It is the responsibility of software to avoid erroneous operation.

When not stopped, the watchdog may generate a reset to exit the low-power modes.

7.3.4.12 Programmable Interrupt Timers (PIT0, PIT1, PIT2 and PIT3)

In stop mode (or in doze mode, if so programmed), the programmable interrupt timer (PIT) ceases operation, and freezes at the current value. When exiting these modes, the PIT resumes operation from the stopped value. It is the responsibility of software to avoid erroneous operation.

When not stopped, the PIT may generate an interrupt to exit the low-power modes.

7.3.4.13 DMA Timers (DTIM0–DTIM3)

In wait and doze modes, the DMA timers may generate an interrupt to exit a low-power mode. This interrupt can be generated when the DMA Timer is in either input capture mode or reference compare mode.

In input capture mode, where the capture enable (CE) field of the timer mode register (DTMR) has a non-zero value and the DMA enable (DMAEN) bit of the DMA timer extended mode register (DTXMR) is cleared, an interrupt is issued upon a captured input. In reference compare mode, where the output reference request interrupt enable (ORRI) bit of DTMR is set and the DTXMR[DMAEN] bit is cleared, an interrupt is issued when the timer counter reaches the reference value.

DMA timer operation is disabled in stop mode, but the DMA timer is unaffected by either the wait or doze modes and may generate an interrupt to exit these modes. Upon exiting stop mode, the timer will resume operation unless stop mode was exited by reset.

7.3.4.14 Queued Serial Peripheral Interface (QSPI)

In wait and doze modes, the queued serial peripheral interface (QSPI) may generate an interrupt to exit the low-power modes.

- Clearing the QSPI enable bit (SPE) disables the QSPI function.
- The QSPI is unaffected by wait mode and may generate an interrupt to exit this mode.

In stop mode, the QSPI stops immediately and freezes operation, register values, state machines, and external pins. During this mode, the QSPI clocks are shut down. Coming out of stop mode returns the QSPI to operation from the state prior to the low-power mode entry.

7.3.4.15 UART Modules (UART0, UART1, and UART2)

In wait and doze modes, the UART may generate an interrupt to exit the low-power modes.

- Clearing the transmit enable bit (TE) or the receiver enable bit (RE) disables UART functions.
- The UARTs are unaffected by wait mode and may generate an interrupt to exit this mode.

In stop mode, the UARTs stop immediately and freeze their operation, register values, state machines, and external pins. During this mode, the UART clocks are shut down. Coming out of stop mode returns the UARTs to operation from the state prior to the low-power mode entry.

7.3.4.16 I^{2}C Module

When the I^{2}C Module is enabled by the setting of the I2CR[IEN] bit and when the device is not in stop mode, the I^{2}C module is operable and may generate an interrupt to bring the device out of a low-power mode. For an interrupt to occur, the I2CR[IIE] bit must be set to enable interrupts, and the setting of the I2SR[IIF] generates the interrupt signal to the CPU and interrupt controller. The setting of I2SR[IIF] signifies either the completion of one byte transfer or the reception of a calling address matching its own specified address when in slave receive mode.

In stop mode, the I^{2}C Module stops immediately and freezes operation, register values, and external pins. Upon exiting stop mode, the I^{2}C resumes operation unless stop mode was exited by reset.

7.3.4.17 JTAG

The JTAG (Joint Test Action Group) controller logic is clocked using the TCLK input and is not affected by the system clock. The JTAG cannot generate an event to cause the CPU to exit any low-power mode. Toggling TCLK during any low-power mode will increase the system current consumption.

7.3.4.18 BDM

Entering halt mode via the BDM port (by asserting the external $\overline{\text{BKPT}}$ pin) will cause the CPU to exit any low-power mode.

7.3.5 Summary of Peripheral State During Low-Power Modes

The functionality of each of the peripherals and CPU during the various low-power modes is summarized in Table 7-8. The status of each peripheral during a given mode refers to the condition the peripheral automatically assumes when the STOP instruction is executed and the LPCR[LPMD] field is set for the particular low-power mode. Individual peripherals may be disabled by programming its dedicated control bits. The wake-up capability field refers to the ability of an interrupt or reset by that peripheral to force the CPU into run mode.

Table 7-8. CPU and Peripherals in Low-Power Modes

Module	Peripheral Status[1] / Wake-up Capability					
	Wait Mode		**Doze Mode**		**Stop Mode**	
ColdFire Core	Stopped	No	Stopped	No	Stopped	No
SRAM	Stopped	No	Stopped	No	Stopped	No
Clock Module	Enabled	Yes[3]	Enabled	Yes[3]	Program	Yes[3]
Power Management	Enabled	No	Enabled	No	Stopped	No
Chip Configuration Module	Enabled	No	Enabled	No	Stopped	No
Reset Controller	Enabled	Yes[2]	Enabled	Yes[2]	Enabled	Yes[2]
System Control Module	Enabled	Yes[2]	Enabled	Yes[2]	Stopped	No
GPIO Module	Enabled	No	Enabled	No	Enabled	No
Interrupt controller	Enabled	Yes[3]	Enabled	Yes[3]	Enabled	Yes[3]
Edge port	Enabled	Yes[3]	Enabled	Yes[3]	Stopped	Yes[3]
eMA Controller	Enabled	Yes	Enabled	Yes	Stopped	No
Programmable Interrupt Timers	Enabled	Yes[3]	Program	Yes[3]	Stopped	No
DMA Timers	Enabled	Yes[3]	Enabled	Yes[3]	Stopped	No
QSPI	Enabled	Yes[3]	Enabled	Yes[3]	Stopped	No
UARTs	Enabled	Yes[3]	Enabled	Yes[3]	Stopped	No
I^{2}C Module	Enabled	Yes[3]	Enabled	Yes[3]	Stopped	No
JTAG	Enabled	No	Enabled	No	Enabled	No
BDM	Enabled	Yes[4]	Enabled	Yes[4]	Enabled	Yes[4]

[1] "Program" indicates that the peripheral function during the low-power mode is dependent on programmable bits in the peripheral register map.

[2] These modules can generate a reset which will exit any low-power mode.

[3] These modules can generate a interrupt which will exit a low-power mode. The CPU will begin to service the interrupt exception after wake-up.

[4] The BDM logic is clocked by a separate TCLK clock. Entering halt mode via the BDM port exits any low-power mode. Upon exit from halt mode, the previous low-power mode will be re-entered and changes made in halt mode will remain in effect.

Chapter 8
Chip Configuration Module (CCM)

8.1　Introduction

This chapter describes the various operating configurations of the device. It provides a description of signals used by the CCM and a programming model.

8.1.1　Block Diagram

The Chip Configuration Module (CCM) controls the chip configuration and mode of operation for the MCF5213
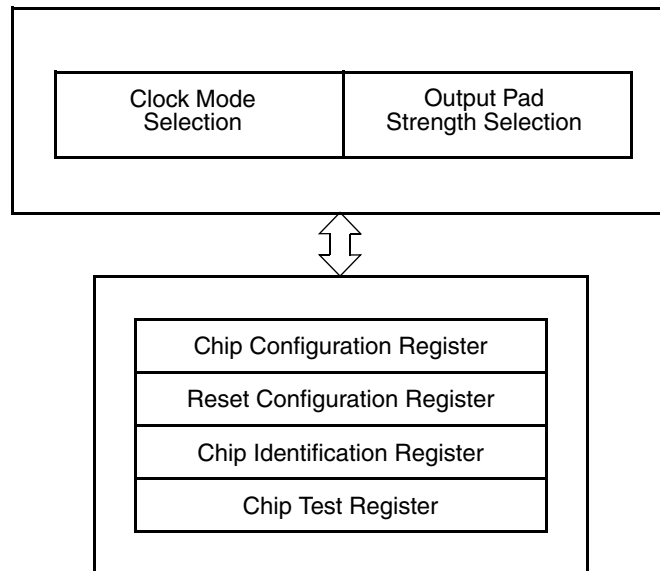
Figure 8-1. Chip Configuration Module Block Diagram

8.1.2　Features

- The CCM performs these operations.Selects external clock or phase-lock loop (PLL) mode with internal or external reference
- Selects output pad drive strength
- Selects low-power configuration
- Selects processor status (PSTAT) and processor debug data (DDATA) functions
- Selects BDM or JTAG mode

8.2 External Signal Descriptions

Table 8-1 provides an overview of the CCM signals.

Table 8-1. Signal Properties

Name	Function	Reset State
$\overline{\text{RCON}}$	Reset configuration select	Internal weak pull-up device
CLKMOD[1:0]	Clock mode select[1]	—

[1] Refer to Chapter 7, "Clock Module" for more information.

8.2.1 $\overline{\text{RCON}}$

If the external $\overline{\text{RCON}}$ pin is asserted during reset, then various chip functions, including the reset configuration pin functions after reset, are configured according to the levels driven onto the external data pins (see Section 8.4, "Functional Description"). The internal configuration signals are driven to reflect the levels on the external configuration pins to allow for module configuration.

8.2.2 CLKMOD[1:0]

The state of the CLKMOD[1:0] pins during reset determines the clock mode after reset. Refer to Chapter 7, "Clock Module" for more information.

8.3 Memory Map/Register Definition

This subsection provides a description of the memory map and registers.

8.3.1 Programming Model

The CCM programming model consists of these registers:

- The chip configuration register (CCR) controls the main chip configuration.
- The reset configuration register (RCON) indicates the default chip configuration.
- The chip identification register (CIR) contains a unique part number.

Some control register bits are implemented as write-once bits. These bits are always readable, but once the bit has been written, additional writes have no effect, except during debug and test operations.

Some write-once bits can be read and written while in debug mode. When debug mode is exited, the chip configuration module resumes operation based on the current register values. If a write to a write-once register bit occurs while in debug mode, the register bit remains writable on exit from debug or test mode. Table 8-2 shows the accessibility of write-once bits.

Table 8-2. Write-Once Bits Read/Write Accessibility

Configuration	Read/Write Access
All configurations	Read-always
Debug operation	Write-always

8.3.2 Memory Map

Table 8-3. Chip Configuration Module Memory Map

IPSBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]	Access[1]
0x11_0004	Chip Configuration Register (CCR)		Low-Power Control Register (LPCR)[2]		S
0x11_0008	Reset Configuration Register (RCON)		Chip Identification Register (CIR)		S
0x11_000C	Reserved[3]				S
0x11_0010	Unimplemented[4]				—

[1] S = CPU supervisor mode access only. User mode accesses to supervisor only addresses have no effect and result in a cycle termination transfer error.

[2] See Chapter 8, "Power Management," for a description of the LPCR. It is shown here only to warn against accidental writes to this register.

[3] Writing to reserved addresses with values other than 0 could put the device in a test mode; reading returns 0s.

[4] Accessing an unimplemented address has no effect and causes a cycle termination transfer error.

NOTE

To safeguard against unintentionally activating test logic, write 0x0000 to the above reserved location during initialization (immediately after reset) to lock out test features. Setting any bits in the CCR may lead to unpredictable results.

8.3.3 Register Descriptions

The following subsection describes the CCM registers.

8.3.3.1 Chip Configuration Register (CCR)

	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	SZEN	PSTEN	0	BME		BMT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0		0	0	1
Address	IPSBAR + 0x11_0004															

Figure 8-2. Chip Configuration Register (RCON)

Table 8-4. CCR Field Descriptions

Bits	Name	Description
15–7	—	Reserved, should be cleared.
6	SZEN	TSIZ[1:0] enable. This read/write bit enables the TSIZ[1:0] function of the external pins. 0 TSIZ[1:0] function disabled. 1 TSIZ[1:0] function enabled.
5	PSTEN	PST[3:0]/DDATA[3:0] enable. This read/write bit enables the Processor Status (PST) and Debug Data (DDATA)n functions of the external pins. 0 PST/DDATA function disabled. 1 PST/DDATA function enabled.
4	—	Reserved, should be cleared.
3	BME	Bus monitor enable. This read/write bit enables the bus monitor to operate during external bus cycles. 0 Bus monitor disabled for external bus cycles. 1 Bus monitor enabled for external bus cycles. Table 8-2 shows the read/write accessibility of this write-once bit.
2–0	BMT	Bus monitor timing. This field selects the timeout period (in system clocks) for the bus monitor. 000 65536 001 32768 010 16384 011 8192 100 4096 101 2048 110 1024 111 512 Table 8-2 shows the read/write accessibility of this write-once bit.

8.3.3.2   Reset Configuration Register (RCON)

At reset, RCON determines the default operation of certain chip functions. All default functions defined by the RCON values can only be overridden during reset configuration (see Section 8.4.1, "Reset Configuration") if the external $\overline{\text{RCON}}$ pin is asserted. RCON is a read-only register.

	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	RLOAD	0	0	0	0	MODE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
Address	IPSBAR + 0x11_0008															

Figure 8-3. Reset Configuration Register (RCON)

Table 8-5. RCON Field Descriptions

Bits	Name	Description
15–6	—	Reserved, should be cleared.
5	RLOAD	Pad driver load. Reflects the default pad driver strength configuration. 0 2mA (partial) drive strength (This is the default value used for the MCF5213.) 1 10mA (full) drive strength
4–1	—	Reserved, should be cleared.
0	MODE	Chip configuration mode. Reflects the default chip configuration mode. 0 Single-chip mode (This is the value used for the MCF5213.) 1 Reserved.) The default mode cannot be overridden during reset configuration.

8.3.3.3   Chip Identification Register (CIR)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R						PIN								PRN		
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Address							IPSBAR + 0x11_000A									
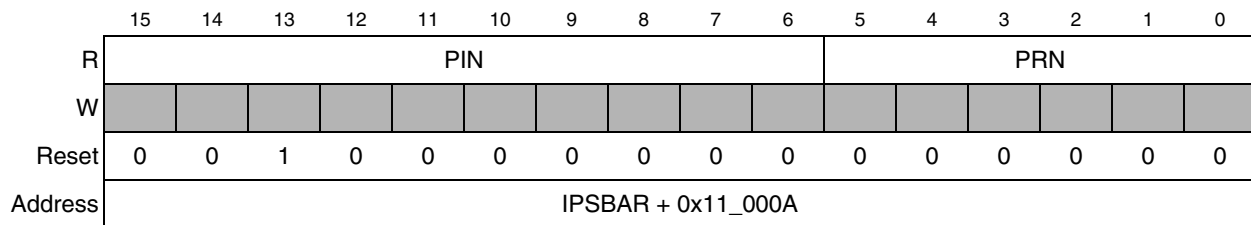
Figure 8-4. Chip Identification Register (CIR)

Table 8-6. CIR Field Description

Bits	Name	Description
15–6	PIN	Part identification number. Contains a unique identification number for the device. MCF5211 = 0x3C MCF5212 = 0x42 MCF5213 = 0x43
5–0	PRN	Part revision number. This number is increased by one for each new full-layer mask set of this part. The revision numbers are assigned in chronological order, beginning with zero.

8.4   Functional Description

Three functions are defined within the chip configuration module:

1. Reset configuration
2. Output pad strength configuration
3. Clock mode selections

These functions are described in the following sections.

8.4.1 Reset Configuration

During reset, the pins for the reset override functions are immediately configured to known states. Table 8-7 shows the states of the external pins while in reset.

Table 8-7. Reset Configuration Pin States During Reset

Pin	Pin Function[1]	I/O	Output State	Input State
RCON	$\overline{\text{RCON}}$ function for all modes[2]	Input	—	Internal weak pull-up device
CLKMOD1, CLKMOD0	Not affected	Input	—	Must be driven by external logic

[1] If the external $\overline{\text{RCON}}$ pin is not asserted during reset, pin functions are determined by the default operation mode defined in the RCON register. If the external $\overline{\text{RCON}}$ pin is asserted, pin functions are determined by the override values driven on the external data bus pins.

[2] During reset, the external $\overline{\text{RCON}}$ pin assumes its $\overline{\text{RCON}}$ pin function, but this pin changes to the function defined by the chip operation mode immediately after reset. See Table 8-8.

If the $\overline{\text{RCON}}$ pin is not asserted during reset, the chip configuration and the reset configuration pin functions after reset are determined by RCON or fixed defaults, regardless of the states of the external data pins. The internal configuration signals are driven to levels specified by the RCON register's reset state for default module configuration.

If the $\overline{\text{RCON}}$ pin is asserted during reset, then various chip functions, including the reset configuration pin functions after reset, are configured according to the levels driven onto the external data pins. (See Table 8-8) The internal configuration signals are driven to reflect the levels on the external configuration pins to allow for module configuration.

Table 8-8. Configuration During Reset[1]

Pin(s) Affected	Default Configuration	Override Pins in Reset[2,]	Function
All output pins	RCON[5] = 1	0	Partial strength
		1	Full strength[4]
Clock mode	No default[3]	**CLKMOD1, CLKMOD0**	**Clock Mode**
		00	External clock mode (PLL disabled)
		01	1:1 PLL mode
		10	Normal PLL mode with external clock reference
		11	Normal PLL mode w/crystal reference

[1] Modifying the default configurations is possible only if the external $\overline{\text{RCON}}$ pin is asserted.

2   The external reset override circuitry drives the data bus pins with the override values while $\overline{\text{RSTOUT}}$ is asserted. It must stop driving the data bus pins within one CLKOUT cycle after $\overline{\text{RSTOUT}}$ is negated. To prevent contention with the external reset override circuitry, the reset override pins are forced to inputs during reset and do not become outputs until at least one CLKOUT cycle after $\overline{\text{RSTOUT}}$ is negated. $\overline{\text{RCON}}$ must also be negated within one cycle after $\overline{\text{RSTOUT}}$ is negated.

3   There is no default configuration for clock mode selection. The actual values for the CLKMOD pins must always be driven during reset. Once out of reset, the CLKMOD pins have no effect on the clock mode selection.

8.4.2   Output Pad Strength Configuration

Output pad strength is determined during reset configuration. Once reset is exited, the output pad strength configuration can be changed by programming the Pin Drive Strength Register as described in Section 11.6.6.2.

8.4.3   Clock Mode Selection

The clock mode is selected during reset and reflected in the PLLMODE, PLLSEL, and PLLREF bits of SYNSR. Once reset is exited, the clock mode cannot be changed. Table 8-9 summarizes clock mode selection during reset configuration.

Table 8-9. Clock Mode Selection[1]

Clock Mode	CLKMOD[1]	CLKMOD[0]	PLL SYNSR Bits		
			PLLMODE	PLLSEL	PLLREF
External clock mode; PLL disabled	0	0	0	0	0
1:1 PLL mode	0	1	1	0	0
Normal PLL mode; external clock reference	1	0	1	1	0
Normal PLL mode; crystal oscillator reference	1	0	1	1	1

1   There is no default configuration for clock mode selection. The actual values for the CLKMOD pins must always be driven during reset. Once out of reset, the CLKMOD pins have no effect on the clock mode selection.

8.5   Reset

Reset initializes CCM registers to a known startup state as described in Section 8.3, "Memory Map/Register Definition." The CCM controls chip configuration at reset as described in Section 8.4, "Functional Description."

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 9
Reset Controller Module

The reset controller is provided to determine the cause of reset, assert the appropriate reset signals to the system, and then to keep a history of what caused the reset. The Low Voltage Detection module, which generates low-voltage detect (LVD) interrupts and resets, is implemented within the reset controller module.

9.1 Features

Module features include:
- Seven sources of reset:
  — External
  — Power-on reset (POR)
  — Phase locked-loop (PLL) loss of lock
  — PLL loss of clock
  — Software
  — LVD reset
- Software-assertable $\overline{\text{RSTO}}$ pin independent of chip reset state
- Software-readable status flags indicating the cause of the last reset
- LVD control and status bits for setup and use of LVD reset or interrupt

9.2 Block Diagram

Figure 9-1 illustrates the reset controller and is explained in the following sections.
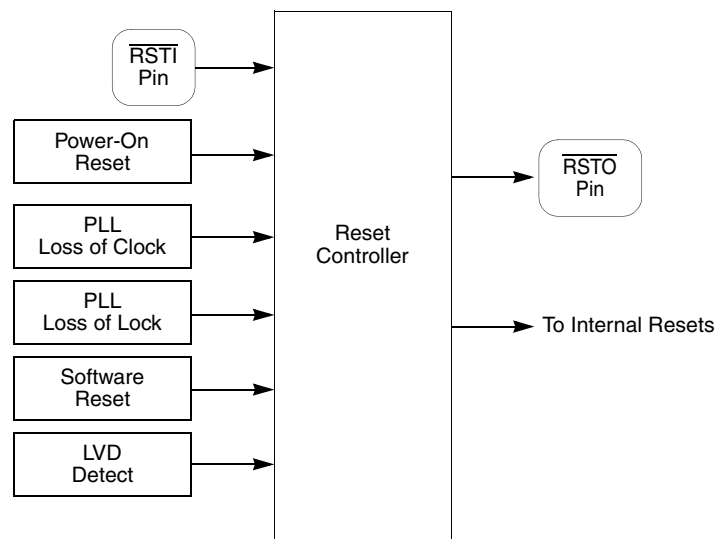
Figure 9-1. Reset Controller Block Diagram

MCF5213 Reference Manual, Rev. 1.1

Preliminary

9.3 Signals

Table 9-1 provides a summary of the reset controller signal properties. The signals are described in the following paragraphs.

Table 9-1. Reset Controller Signal Properties

Name	Direction	Input Hysteresis	Input Synchronization
$\overline{\text{RSTI}}$	I	Y	Y[1]
$\overline{\text{RSTO}}$	O	—	—

[1] $\overline{\text{RSTI}}$ is always synchronized except when in low-power stop mode.

9.3.1 $\overline{\text{RSTI}}$

Asserting the external $\overline{\text{RSTI}}$ for at least four rising CLKOUT edges causes the external reset request to be recognized and latched.

9.3.2 $\overline{\text{RSTO}}$

This active-low output signal is driven low when the internal reset controller module resets the chip. When $\overline{\text{RSTO}}$ is active, the user can drive override options on the data bus.

9.4 Memory Map and Registers

The reset controller programming model consists of these registers:

- Reset control register (RCR), which selects reset controller functions
- Reset status register (RSR), which reflects the state of the last reset source

See Table 9-2 for the memory map and the following paragraphs for a description of the registers.

Table 9-2. Reset Controller Memory Map

IPSBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]	Access[1]
0x11_0000	RCR	RSR	Reserved[2]	Reserved[2]	S/U

[1] S/U = supervisor or user mode access.

[2] Writes to reserved address locations have no effect and reads return 0s.

9.4.1 Reset Control Register (RCR)

The RCR allows software control for requesting a reset, for independently asserting the external $\overline{\text{RSTO}}$ pin, and for controlling low-voltage detect (LVD) functions.

	7	6	5	4			0
Field	SOFTRST	FRCRSTOUT	—	LVDF	LVDIE	LVDRE	—	LVDE
Reset	0000_0101							
R/W	R/W							
Address	IPSBAR + 0x11_0000							

Figure 9-2. Reset Control Register (RCR)

Table 9-3. RCR Field Descriptions

Bit(s)	Name	Description
7	SOFTRST	Allows software to request a reset. The reset caused by setting this bit clears this bit. 1 Software reset request 0 No software reset request
6	FRCRSTOUT	Allows software to assert or negate the external $\overline{RSTO}$ pin. 1 Assert $\overline{RSTO}$ pin 0 Negate $\overline{RSTO}$ pin CAUTION: External logic driving reset configuration data during reset needs to be considered when asserting the $\overline{RSTO}$ pin when setting FRCRSTOUT.
5	—	Reserved, should be cleared.
4	LVDF	LVD flag. Indicates the low-voltage detect status if LVDE is set. Write a 1 to clear the LVDF bit. 1 Low voltage has been detected 0 Low voltage has not been detected NOTE: The setting of this flag causes an LVD interrupt if LVDE and LVDIE bits are set and LVDRE is cleared when the supply voltage V_{DD} drops below V_{DD} (minimum). The vector for this interrupt is shared with INT0 of the EPORT module. Interrupt arbitration in the interrupt service routine is necessary if both of these interrupts are enabled. Also, LVDF is not cleared at reset, however it will always initialize to a zero since the part will not come out of reset while in a low-power state (LVDE/LVDRE bits are enabled out of reset).
3	LVDIE	LVD interrupt enable. Controls the LVD interrupt if LVDE is set. This bit has no effect if the LVDE bit is a logic 0. 1 LVD interrupt enabled 0 LVD interrupt disabled
2	LVDRE	LVD reset enable. Controls the LVD reset if LVDE is set. This bit has no effect if the LVDE bit is a logic 0. LVD reset has priority over LVD interrupt, if both are enabled. 1 LVD reset enabled 0 LVD reset disabled
1	—	Reserved, should be cleared.
0	LVDE	Controls whether the LVD is enabled. 1 LVD is enabled 0 LVD is disabled

9.4.2 Reset Status Register (RSR)

The RSR contains a status bit for every reset source. When reset is entered, the cause of the reset condition is latched along with a value of 0 for the other reset sources that were not pending at the time of the reset

condition. These values are then reflected in RSR. One or more status bits may be set at the same time. The cause of any subsequent reset is also recorded in the register, overwriting status from the previous reset condition.

RSR can be read at any time. Writing to RSR has no effect.

	7	6	5	4	3	2	1	0
Field	—	LVD	SOFT	—	POR	EXT	LOC	LOL
Reset	Reset Dependent							
R/W	R							
Address	IPSBAR + 0x11_0001							

Figure 9-3. Reset Status Register (RSR)

Table 9-4. RSR Field Descriptions

Bit(s)	Name	Description
7	—	Reserved, should be cleared.
6	LVD	Low voltage detect. Indicates that the last reset state was caused by an LVD reset. 1 Last reset state was caused by an LVD reset 0 Last reset state was not caused by an LVD reset
5	SOFT	Software reset flag. Indicates that the last reset was caused by software. 1 Last reset caused by software 0 Last reset not caused by software
4	—	Reserved, should be cleared.
3	POR	Power-on reset flag. Indicates that the last reset was caused by a power-on reset. 1 Last reset caused by power-on reset 0 Last reset not caused by power-on reset
2	EXT	External reset flag. Indicates that the last reset was caused by an external device asserting the external $\overline{RSTI}$ pin. 1 Last reset state caused by external reset 0 Last reset not caused by external reset
1	LOC	Loss-of-clock reset flag. Indicates that the last reset state was caused by a PLL loss of clock. 1 Last reset caused by loss of clock 0 Last reset not caused by loss of clock
0	LOL	Loss-of-lock reset flag. Indicates that the last reset state was caused by a PLL loss of lock. 1 Last reset caused by a loss of lock 0 Last reset not caused by loss of lock

9.5 Functional Description

9.5.1 Reset Sources

Table 9-5 defines the sources of reset and the signals driven by the reset controller.

Table 9-5. Reset Source Summary

Source	Type
Power on	Asynchronous
External $\overline{RSTI}$ pin (not stop mode)	Synchronous
External $\overline{RSTI}$ pin (during stop mode)	Asynchronous
Loss of clock	Asynchronous
Loss of lock	Asynchronous
Software	Synchronous
LVD reset	Asynchronous

To protect data integrity, a synchronous reset source is not acted upon by the reset control logic until the end of the current bus cycle. Reset is then asserted on the next rising edge of the system clock after the cycle is terminated. Whenever the reset control logic must synchronize reset to the end of the bus cycle, the internal bus monitor is automatically enabled regardless of the BME bit state in the chip configuration register (CCR). Then, if the current bus cycle is not terminated normally the bus monitor terminates the cycle based on the length of time programmed in the BMT field of the CCR.

Internal byte, word, or longword writes are guaranteed to complete without data corruption when a synchronous reset occurs. External writes, including longword writes to 16-bit ports, are also guaranteed to complete.

Asynchronous reset sources usually indicate a catastrophic failure. Therefore, the reset control logic does not wait for the current bus cycle to complete. Reset is asserted immediately to the system.

9.5.1.1   Power-On Reset

At power up, the reset controller asserts $\overline{RSTO}$. $\overline{RSTO}$ continues to be asserted until V_{DD} has reached a minimum acceptable level and, if PLL clock mode is selected, until the PLL achieves phase lock. Then after approximately another 512 cycles, RSTO is negated and the part begins operation.

9.5.1.2   External Reset

Asserting the external $\overline{RSTI}$ for at least four rising CLKOUT edges causes the external reset request to be recognized and latched. The bus monitor is enabled and the current bus cycle is completed. The reset controller asserts RSTO for approximately 512 cycles after RSTI is negated and the PLL has acquired lock. The part then exits reset and begins operation.

In low-power stop mode, the system clocks are stopped. Asserting the external $\overline{RSTI}$ in stop mode causes an external reset to be recognized.

9.5.1.3   Loss-of-Clock Reset

This reset condition occurs in PLL clock mode when the LOCRE bit in the SYNCR is set and either the PLL reference or the PLL itself fails. The reset controller asserts RSTO for approximately 512 cycles after the PLL has acquired lock. The part then exits reset and begins operation.

9.5.1.4 Loss-of-Lock Reset

This reset condition occurs in PLL clock mode when the LOLRE bit in the SYNCR is set and the PLL loses lock. The reset controller asserts $\overline{\text{RSTO}}$ for approximately 512 cycles after the PLL has acquired lock. The part then exits reset and resumes operation.

9.5.1.5 Software Reset

A software reset occurs when the SOFTRST bit is set. If the $\overline{\text{RSTI}}$ is negated and the PLL has acquired lock, the reset controller asserts $\overline{\text{RSTO}}$ for approximately 512 cycles. Then the part exits reset and resumes operation.

9.5.1.6 LVD Reset

The LVD reset will occur when the supply input voltage, $V_{DD,}$ drops below V_{LVD} (minimum).

9.5.2 Reset Control Flow

The reset logic control flow is shown in Figure 9-4. In this figure, the control state boxes have been numbered, and these numbers are referred to (within parentheses) in the flow description that follows. All cycle counts given are approximate.
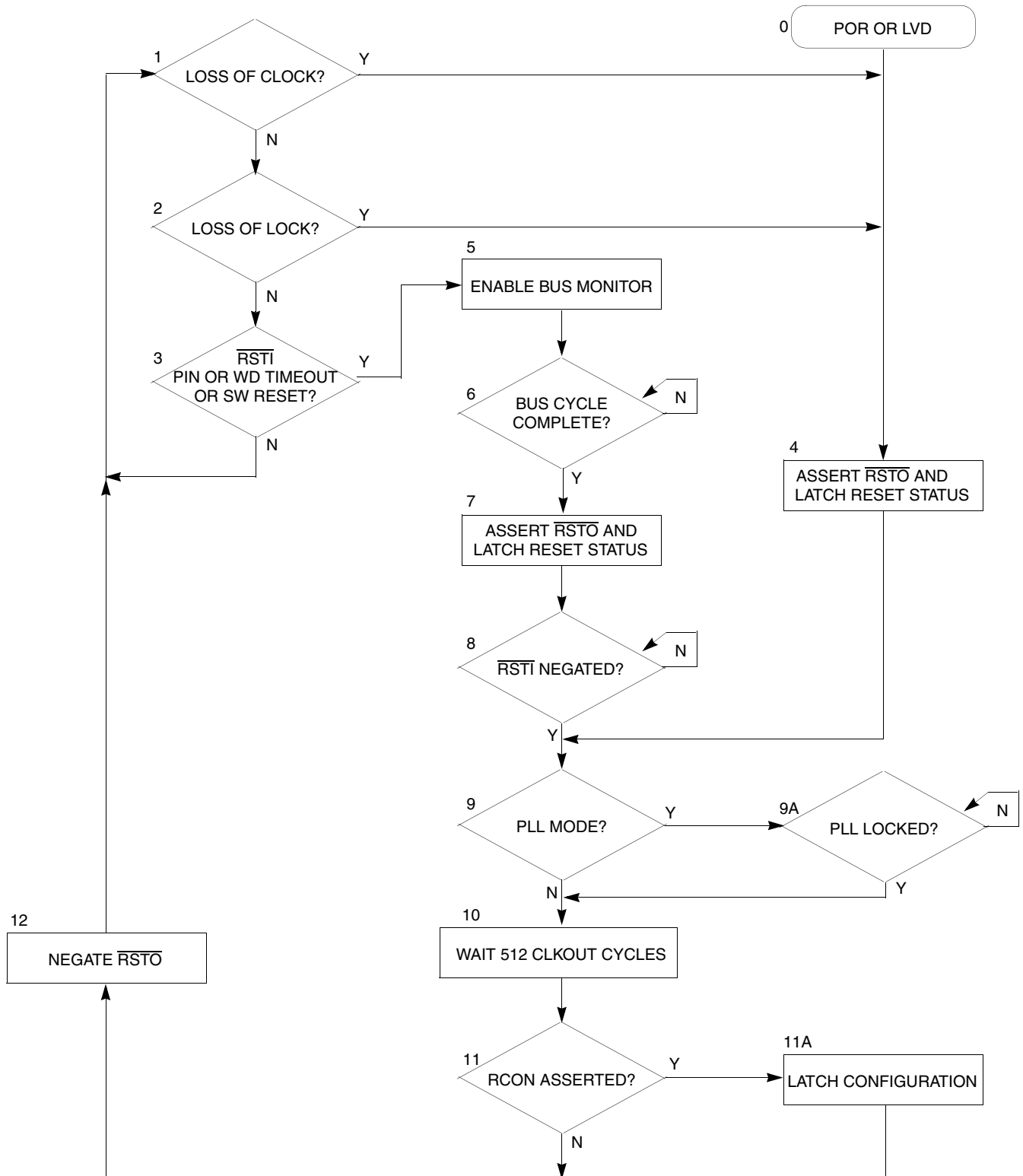
Figure 9-4. Reset Control Flow

9.5.2.1 Synchronous Reset Requests

In this discussion, the reference in parentheses refer to the state numbers in Figure 9-4. All cycle counts given are approximate.

If the external $\overline{\text{RSTI}}$ signal is asserted by an external device for at least four rising CLKOUT edges (3), if software requests a reset, the reset control logic latches the reset request internally and enables the bus monitor (5). When the current bus cycle is completed (6), $\overline{\text{RSTO}}$ is asserted (7). The reset control logic waits until the $\overline{\text{RSTI}}$ signal is negated (8) and for the PLL to attain lock (9, 9A) before waiting 512 CLKOUT cycles (1). The reset control logic may latch the configuration according to the $\overline{\text{RCON}}$ signal level (11, 11A) before negating $\overline{\text{RSTO}}$ (12).

If the external $\overline{\text{RSTI}}$ signal is asserted by an external device for at least four rising CLKOUT edges during the 512 count (10) or during the wait for PLL lock (9A), the reset flow switches to (8) and waits for the $\overline{\text{RSTI}}$ signal to be negated before continuing.

9.5.2.2 Internal Reset Request

If reset is asserted by an asynchronous internal reset source, such as loss of clock (1) or loss of lock (2), the reset control logic asserts $\overline{\text{RSTO}}$ (4). The reset control logic waits for the PLL to attain lock (9, 9A) before waiting 512 CLKOUT cycles (1). Then the reset control logic may latch the configuration according to the $\overline{\text{RCON}}$ pin level (11, 11A) before negating $\overline{\text{RSTO}}$ (12).

If loss of lock occurs during the 512 count (10), the reset flow switches to (9A) and waits for the PLL to lock before continuing.

9.5.2.3 Power-On Reset/Low-Voltage Detect Reset

When the reset sequence is initiated by power-on reset (0), the same reset sequence is followed as for the other asynchronous reset sources.

9.5.3 Concurrent Resets

This section describes the concurrent resets. As in the previous discussion references in parentheses refer to the state numbers in Figure 9-4.

9.5.3.1 Reset Flow

If a power-on reset or low-voltage detect condition is detected during any reset sequence, the reset sequence starts immediately (0).

If the external $\overline{\text{RSTI}}$ pin is asserted for at least four rising CLKOUT edges while waiting for PLL lock or the 512 cycles, the external reset is recognized. Reset processing switches to wait for the external $\overline{\text{RSTI}}$ pin to negate (8).

If a loss-of-clock or loss-of-lock condition is detected while waiting for the current bus cycle to complete (5, 6) for an external reset request, the cycle is terminated. The reset status bits are latched (7) and reset processing waits for the external $\overline{\text{RSTI}}$ pin to negate (8).

If a loss-of-clock or loss-of-lock condition is detected during the 512 cycle wait, the reset sequence continues after a PLL lock (9, 9A).

9.5.3.2 Reset Status Flags

For a POR reset, the POR and LVD bits in the RSR are set, and the SOFT, WDR, EXT, LOC, and LOL bits are cleared even if another type of reset condition is detected during the reset sequence for the POR.

If a loss-of-clock or loss-of-lock condition is detected while waiting for the current bus cycle to complete (5, 6) for an external reset request, the EXT, SOFT, and/or WDR bits along with the LOC and/or LOL bits are set.

If the RSR bits are latched (7) during the EXT, SOFT, and/or WDR reset sequence with no other reset conditions detected, only the EXT, SOFT, and/or WDR bits are set.

If the RSR bits are latched (4) during the internal reset sequence with the $\overline{RSTI}$ pin not asserted and no SOFT or WDR event, then the LOC and/or LOL bits are the only bits set.

For a LVD reset, the LVD bit in the RSR is set, and the SOFT, WDR, EXT, LOC, and LOL bits are cleared to 0 even if another type of reset condition is detected during the reset sequence for LVD.

Chapter 10
System Control Module (SCM)

This section details the functionality of the System Control Module (SCM) which provides the programming model for the System Access Control Unit (SACU), the system bus arbiter, a 32-bit core watchdog timer (CWT), and the system control registers and logic. Specifically, the system control includes the internal peripheral system (IPS) base address register (IPSBAR), the processor's dual-port RAM base address register (RAMBAR), and system control registers that include the core watchdog timer control.

10.1   Overview

The SCM provides the control and status for a variety of functions including base addressing and address space masking for both the IPS peripherals and resources (IPSBAR) and the ColdFire core memory spaces (RAMBAR). The CPU core supports two memory banks, one for the internal SRAM and the other for the internal Flash.

The SACU provides the mechanism needed to implement secure bus transactions to the system address space.

The programming model for the system bus arbitration resides in the SCM. The SCM sources the necessary control signals to the arbiter for bus master management.

The CWT provides a means of preventing system lockup due to uncontrolled software loops via a special software service sequence. If periodic software servicing action does not occur, the CWT times out with a programmed response (system reset or interrupt) to allow recovery or corrective action to be taken.

10.2   Features

The SCM includes these distinctive features:

- IPS base address register (IPSBAR)
    - Base address location for 1-Gbyte peripheral space
    - User control bits
- Processor-local memory base address register (RAMBAR)
- System control registers
    - Core reset status register (CRSR) indicates type of last reset
    - Core watchdog control register (CWCR) for watchdog timer control
    - Core watchdog service register (CWSR) to service watchdog timer
- System bus master arbitration programming model (MPARK)
- System access control unit (SACU) programming model

— Master privilege register (MPR)

— Peripheral access control registers (PACRs)

— Grouped peripheral access control registers (GPACR0, GPACR1)

10.3 Memory Map and Register Definition

The memory map for the SCM registers is shown in Table 10-1. All the registers in the SCM are memory-mapped as offsets within the 1 Gbyte IPS address space and accesses are controlled to these registers by the control definitions programmed into the SACU.

Table 10-1. SCM Register Map

IPSBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x00_0000	IPSBAR			
0x00_0004	—			
0x00_0008	RAMBAR			
0x00_000C	PPMRH			
0x00_0010	CRSR	CWCR	LPICR[1]	CWSR
0x00_0014	DMAREQC[2]			
0x00_0018	PPMRL			
0x00_001C	MPARK			
0x00_0020	MPR	PPMRS	PPMRC	IPSBMT
0x00_0024	PACR0	PACR1	PACR2	PACR3
0x00_0028	PACR4	PACR5	PACR6	PACR7
0x00_002c	PACR8	—	—	—
0x00_0030	GPACR0	GPACR1	—	—
0x00_0034	—	—	—	—
0x00_0038	—	—	—	—
0x00_003C	—	—	—	—

[1] The LPICR is described in Chapter 7, "Power Management."

[2] The DMAREQC register is described in Chapter 14, "DMA Controller Module"

10.4 Register Descriptions

10.4.1 Internal Peripheral System Base Address Register (IPSBAR)

The IPSBAR specifies the base address for the 1 Gbyte memory space associated with the on-chip peripherals. At reset, the base address is loaded with a default location of 0x4000_0000 and marked as valid (IPSBAR[V]=1). If desired, the address space associated with the internal modules can be moved by loading a different value into the IPSBAR at a later time.

Preliminary

NOTE

Accessing reserved IPSBAR memory space could result in an unterminated bus cycle that causes the core to hang. Only a hard reset will allow the core to recover from this state. Therefore, all bus accesses to IPSBAR space should fall within a module's memory map space.

If an address "hits" in overlapping memory regions, the following priority is used to determine what memory is accessed:

1. IPSBAR
2. RAMBAR
3. Chip Selects

NOTE

This is the list of memory access priorities when viewed from the processor core.

See Figure 10-1 and Table 10-2 for descriptions of the bits in IPSBAR.

	31	30	29		16
Field	BA31	BA30	—		
Reset	0	1	—		
R/W	R/W				

	15		1	0
Field	—			V
Reset	—			
R/W	R/W			
Address	IPSBAR + 0x000			

Figure 10-1. IPS Base Address Register (IPSBAR)

Table 10-2. IPSBAR Field Description

Bits	Name	Description
31–30	BA	Base address. Defines the base address of the 1-Gbyte internal peripheral space. This is the starting address for the IPS registers when the valid bit is set.
29–1	—	Reserved, should be cleared.
0	V	Valid. Enables/disables the IPS Base address region. V is set at reset. 0 IPS Base address is not valid. 1 IPS Base address is valid.

10.4.2   Memory Base Address Register (RAMBAR)

The device supports dual-ported local SRAM memory. This processor-local memory can be accessed directly by the core and/or other system bus masters. Since this memory provides single-cycle accesses at processor speed, it is ideal for applications where double-buffer schemes can be used to maximize

system-level performance. For example, a DMA channel in a typical double-buffer (also known as a ping-pong scheme) application may load data into one portion of the dual-ported SRAM while the processor is manipulating data in another portion of the SRAM. Once the processor completes the data calculations, it begins processing the just-loaded buffer while the DMA moves out the just-calculated data from the other buffer, and reloads the next data block into the just-freed memory region. The process repeats with the processor and the DMA "ping-ponging" between alternate regions of the dual-ported SRAM.

The device design implements the dual-ported SRAM in the memory space defined by the RAMBAR register. There are two physical copies of the RAMBAR register: one located in the processor core and accessible only via the privileged MOVEC instruction at CPU space address 0xC05 and another located in the SCM at IPSBAR + 0x008. ColdFire core accesses to this memory are controlled by the processor-local copy of the RAMBAR, while module accesses are enabled by the SCM's RAMBAR.

The physical base address programmed in both copies of the RAMBAR is typically the same value; however, they can be programmed to different values. By definition, the base address must be a 0-modulo-size value.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	BA31	BA30	BA29	BA28	BA27	BA26	BA25	BA24	BA23	BA22	BA21	BA20	BA19	BA18	BA17	BA16
Reset	0000_0000_0000_0000															
R/W	R/W															

	15					10	9	8								0
Field	—						BDE	—								
Reset	0000_0000_0000_0000															
R/W	R/W															
Address	IPSBAR + 0x008															
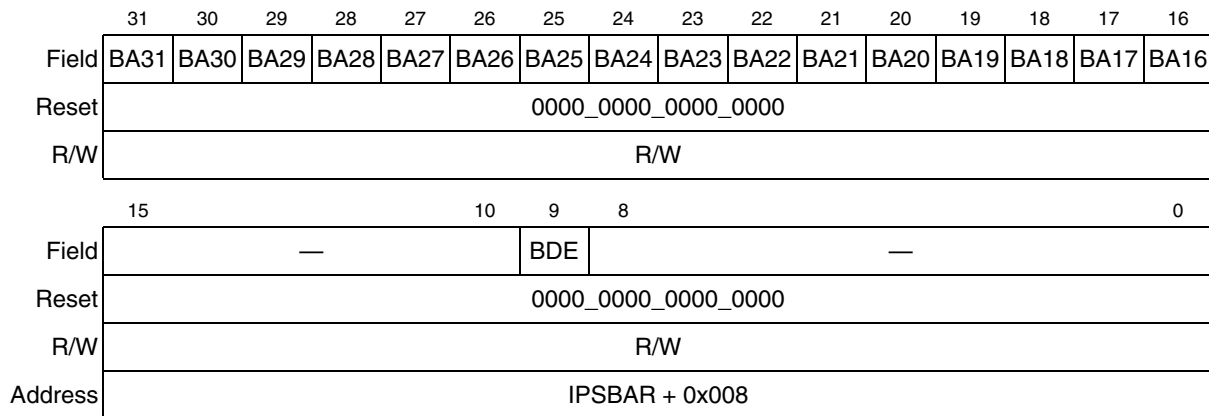
Figure 10-2. Memory Base Address Register (RAMBAR)

Table 10-3. RAMBAR Field Description

Bits	Name	Description
31–16	BA	Base address. Defines the memory module's base address on a 64-Kbyte boundary corresponding to the physical array location within the 4 Gbyte address space supported by ColdFire.
15–10	—	Reserved, should be cleared.
9	BDE	Back door enable. Qualifies the module accesses to the memory. 0 Disables module accesses to the module. 1 Enables module accesses to the module. NOTE: The SPV bit in the CPU's RAMBAR must also be set to allow dual port access to the SRAM. For more information, see Section 6.2.1, "SRAM Base Address Register (RAMBAR)."
8–0	—	Reserved, should be cleared.

The SRAM modules are configured through the RAMBAR shown in Figure 10-2.

- RAMBAR specifies the base address of the SRAM.

- All undefined bits are reserved. These bits are ignored during writes to the RAMBAR and return zeros when read.
- The back door enable bit, RAMBAR[BDE], is cleared at reset, disabling the module access to the SRAM.

NOTE

The RAMBAR default value of 0x0000_0000 is invalid. The RAMBAR located in the processor's CPU space must be initialized with the valid bit set before the CPU (or modules) can access the on-chip SRAM (see Chapter 6, "Static RAM (SRAM)" for more information.

For details on the processor's view of the local SRAM memories, see Section 6.2.1, "SRAM Base Address Register (RAMBAR)."

10.4.3 Core Reset Status Register (CRSR)

The CRSR contains a bit for two of the reset sources to the CPU. A bit set to 1 indicates the last type of reset that occurred. The CRSR is updated by the control logic when the reset is complete. Only one bit is set at any one time in the CRSR. The register reflects the cause of the most recent reset. To clear a bit, a logic 1 must be written to the bit location; writing a zero has no effect.

NOTE

The reset status register (RSR) in the reset controller module (see Chapter 29, "Reset Controller Module") provides indication of all reset sources except the core watchdog timer.

	7	6	5	4			0
Field	EXT	—	CWDR		—		
Reset				See Note			
R/W				R/W			
Address				IPSBAR + 0x010			

Note: The reset value of EXT and CWDR depend on the last reset source. All other bits are initialized to zero.

Figure 10-3. Core Reset Status Register (CRSR)

Table 10-4. CRSR Field Descriptions

Bits	Name	Description
7	EXT	External reset. 1 An external device driving RSTI caused the last reset. Assertion of reset by an external device causes the processor core to initiate reset exception processing. All registers are forced to their initial state.
6	—	Reserved, should be cleared.

Table 10-4. CRSR Field Descriptions

Bits	Name	Description
5	CWDR	Core watchdog timer reset. 1 The last reset was caused by the core watchdog timer. If CWRI in the CWCR is set and the core watchdog timer times out, a hard reset occurs.
4–0	—	Reserved, should be cleared.

10.4.4   Core Watchdog Control Register (CWCR)

The core watchdog timer prevents system lockup if the software becomes trapped in a loop with no controlled exit. The core watchdog timer can be enabled or disabled through CWCR[CWE]. By default it is disabled. If enabled, the watchdog timer requires the periodic execution of a core watchdog servicing sequence. If this periodic servicing action does not occur, the timer times out, resulting in a watchdog timer interrupt or a hardware reset, as programmed, by CWCR[CWRI]. If the timer times out and the core watchdog transfer acknowledge enable bit (CWCR[CWTA]) is set, a watchdog timer interrupt is asserted. If a core watchdog timer interrupt acknowledge cycle has not occurred after another timeout, CWT TA is asserted in an attempt to allow the interrupt acknowledge cycle to proceed by terminating the bus cycle. The setting of CWCR[CWTAVAL] indicates that the watchdog timer TA was asserted.

When the core watchdog timer times out and CWCR[CWRI] is programmed for a software reset, an internal reset is asserted and CRSR[CWDR] is set. To prevent the core watchdog timer from interrupting or resetting, the CWSR must be serviced by performing the following sequence:

1. Write 0x55 to CWSR.
2. Write 0xAA to the CWSR.

Both writes must occur in order before the time-out, but any number of instructions can be executed between the two writes. This order allows interrupts and exceptions to occur, if necessary, between the two writes. Caution should be exercised when changing CWCR values after the software watchdog timer has been enabled with the setting of CWCR[CWE], because it is difficult to determine the state of the core watchdog timer while it is running. The countdown value is constantly compared with the time-out period specified by CWCR[CWT]. The following steps must be taken to change CWT:

1. Disable the core watchdog timer by clearing CWCR[CWE].
2. Reset the counter by writing 0x55 and then 0xAA to CWSR.
3. Update CWCR[CWT].
4. Re-enable the core watchdog timer by setting CWCR[CWE]. This step can be performed in step 3.

The CWCR controls the software watchdog timer, time-out periods, and software watchdog timer transfer acknowledge. The register can be read at any time, but can be written only if the CWT is not pending. At system reset, the software watchdog timer is disabled.

	7	6	5	3	2	1	0
Field	CWE	CWRI	CWT[2:0]		CWTA	CWTAVAL	CWTIC
Reset	0000_0000						
R/W	R/W						
Address	IPSBAR + 0x011						

Figure 10-4. Core Watchdog Control Register (CWCR)

Table 10-5. CWCR Field Description

Bits	Name	Description
7	CWE	Core watchdog enable. 0 SWT disabled. 1 SWT enabled.
6	CWRI	Core watchdog reset/interrupt select. 0 If a time-out occurs, the CWT generates an interrupt to the processor core. The interrupt level for the CWT is programmed in the interrupt control register 7 (ICR7) of INTC0. 1 A CWT time-out generates a soft reset to the entire device.
5–3	CWT[2:0]	Core watchdog timing delay. These bits select the timeout period for the CWT as shown in Table 8-6. At system reset, the CWT field is cleared signaling the minimum time-out period but the watchdog is disabled (cwcr[cwe] = 0). the following table shows the core watchdog timer delay. | CWT [2:0]	CWT Time-Out Period
000	2^9 Bus clock frequency
001	2^{11} Bus clock frequency
010	2^{13} Bus clock frequency
011	2^{15} Bus clock frequency
100	2^{19} Bus clock frequency
101	2^{23} Bus clock frequency
110	2^{27} Bus clock frequency
111	2^{31} Bus clock frequency	
2	CWTA	Core watchdog transfer acknowledge enable. 0 CWTA Transfer acknowledge disabled. 1 CWTA Transfer Acknowledge enabled. After one CWT time-out period of the unacknowledged assertion of the CWT interrupt, the transfer acknowledge asserts, which allows CWT to terminate a bus cycle and allow the interrupt acknowledge to occur.
1	CWTAVAL	Core watchdog transfer acknowledge valid. 0 CWTA Transfer Acknowledge has not occurred. 1 CWTA Transfer Acknowledge has occurred. Write a 1 to clear this flag bit.
0	CWTIF	Core watchdog timer interrupt flag. 0 CWT interrupt has not occurred 1 CWT interrupt has occurred. Write a 1 to clear the interrupt request.

MCF5213 Reference Manual, Rev. 1.1

Preliminary

10.4.5 Core Watchdog Service Register (CWSR)

The software watchdog service sequence must be performed using the CWSR as a data register to prevent a CWT time-out. The service sequence requires two writes to this data register: first a write of 0x55 followed by a write of 0xAA. Both writes must be performed in this order prior to the CWT time-out, but any number of instructions or accesses to the CWSR can be executed between the two writes. If the CWT has already timed out, writing to this register has no effect in negating the CWT interrupt. Figure 10-5 illustrates the CWSR. At system reset, the contents of CWSR are uninitialized.

	7	0
Field	CWSR[7:0]	
Reset	Uninitialized	
R/W	R/W	
Address	IPSBAR + 0x013	

Figure 10-5. Core Watchdog Service Register (CWSR)

10.4.6 Overview

The basic functionality is that of a 4-port, pipelined internal bus arbitration module with the following attributes:

- The master pointed to by the current arbitration pointer may get on the bus with zero latency if the address phase is available. All other requesters face at least a one cycle arbitration pipeline delay in order to meet bus timing constraints on address phase hold.
- If a requester will get an immediate address phase (that is, it is pointed to by the current arbitration pointer and the bus address phase is available), it will be the current bus master and is ignored by arbitration. All remaining requesting ports are evaluated by the arbitration algorithm to determine the next-state arbitration pointer.
- There are two arbitration algorithms, fixed and round-robin. Fixed arbitration sets the next-state arbitration pointer to the highest priority requester. Round-robin arbitration sets the next-state arbitration pointer to the highest priority requester (calculated by adding a requester's fixed priority to the current bus master's fixed priority and then taking this sum modulo the number of possible bus masters).
- The default priority is DMA (M2) > CPU (M0), where M2 is the highest and M0 the lowest priority.
- There are two actions for an idle arbitration cycle, either leave the current arbitration pointer as is or set it to the lowest priority requester.
- The anti-lock-out logic for the fixed priority scheme forces the arbitration algorithm to round-robin if any requester has been held for longer than a specified cycle count.

10.4.7 Arbitration Algorithms

There are two modes of arbitration: fixed and round-robin. This section discusses the differences between them.

10.4.7.1 Round-Robin Mode

Round-robin arbitration is the default mode after reset. This scheme cycles through the sequence of masters as specified by MPARK[M*n*_PRTY] bits. Upon completion of a transfer, the master is given the lowest priority and the priority for all other masters is increased by one.

If no masters are requesting, the arbitration unit must "park", pointing at one of the masters. There are two possibilities, park the arbitration unit on the last active master, or park pointing to the highest priority master. Setting MPARK[PRK_LAST] causes the arbitration pointer to be parked on the highest priority master. In round-robin mode, programming the timeout enable and lockout bits MPARK[13,11:8] will have no effect on the arbitration.

10.4.7.2 Fixed Mode

In fixed arbitration the master with highest priority (as specified by the MPARK[M*n*_PRTY] bits) will win the bus. That master will relinquish the bus when all transfers to that master are complete.

If MPARK[TIMEOUT] is set, a counter will increment for each master for every cycle it is denied access. When a counter reaches the limit set by MPARK[LCKOUT_TIME], the arbitration algorithm will be changed to round-robin arbitration mode until all locks are cleared. The arbitration will then return to fixed mode and the highest priority master will be granted the bus.

As in round-robin mode, if no masters are requesting, the arbitration pointer will park on the highest priority master if MPARK[PRK_LAST] is set, or will park on the master which last requested the bus if cleared.

10.4.8 Bus Master Park Register (MPARK)

The MPARK controls the operation of the system bus arbitration module. The platform bus master connections are defined as:

- Master 2 (M2): 4-channel DMA
- Master 0 (M0): V2 ColdFire Core

31						26	25	24	23	22	21	20	19	18	17	16
Field	—						M2_P_EN	BCR24BIT	—		M2_PRTY		M0_PRTY			
Reset	0011_0000_1110_0001															
R/W	R/W															

15	14	13	12	11			8	7							0
Field	—	FIXED	TIMEOUT	PRKLAST	LCKOUT_TIME				—						
Reset	0000_0000_0000_0000														
R/W	R/W														
Address	IPSBAR + 0x01C														

Figure 10-6. Default Bus Master Park Register (MPARK)

Table 10-6. MPARK Field Description

Bits	Name	Description
31–26	—	Reserved, should be cleared.
25	M2_P_EN	DMA bandwith control enable 0 disable the use of the DMA's bandwidth control to elevate the priority of its bus requests. 1 enable the use of the DMA's bandwidth control to elevate the priority of its bus requests.
24	BCR24BIT	Enables the use of 24 bit byte count registers in the DMA module 0 DMA BCRs function as 16 bit counters. 1 DMA BCRs function as 24 bit counters.
23–22	—	Reserved, should be cleared.
21–20	M2_PRTY	Master priority level for master 2 (DMA Controller) 00 fourth (lowest) priority 01 third priority 10 second priority 11 first (highest) priority
19–18	M0_PRTY	Master priority level for master 0 (ColdFire Core) 00 fourth (lowest) priority 01 third priority 10 second priority 11 first (highest) priority
17–16	—	Reserved, should be cleared.
15	—	Reserved, should be cleared.
14	FIXED	Fixed or round robin arbitration 0 round robin arbitration 1 fixed arbitration
13	TIMEOUT	Timeout Enable 0 disable count for when a master is locked out by other masters. 1 enable count for when a master is locked out by other masters and allow access when LCKOUT_TIME is reached.
12	PRKLAST	Park on the last active master or highest priority master if no masters are active 0 park on last active master 1 park on highest priority master
11–8	LCKOUT_TIME	Lock-out Time. Lock-out time for a master being denied the bus. The lock out time is defined as $2^\wedge$ LCKOUT_TIME[3:0].
7–0	—	Reserved, should be cleared.

The initial state of the master priorities is M2 > M0. System software should guarantee that the programmed Mn_PRTY fields are unique, otherwise the hardware defaults to the initial-state priorities.

10.5   System Access Control Unit (SACU)

This section details the functionality of the System Access Control Unit (SACU) which provides the mechanism needed to implement secure bus transactions to the address space mapped to the internal modules.

10.5.1 Overview

The SACU supports the traditional model of two privilege levels: supervisor and user. Typically, memory references with the supervisor attribute have total accessibility to all the resources in the system, while user mode references cannot access system control and configuration registers. In many systems, the operating system executes in supervisor mode, while application software executes in user mode.

The SACU further partitions the access control functions into two parts: one control register defines the privilege level associated with each bus master, and another set of control registers define the access levels associated with the peripheral modules and the memory space.

The SACU's programming model is physically implemented as part of the System Control Module (SCM) with the actual access control logic included as part of the arbitration controller. Each bus transaction targeted for the IPS space is first checked to see if its privilege rights allow access to the given memory space. If the privilege rights are correct, the access proceeds on the bus. If the privilege rights are insufficient for the targeted memory space, the transfer is immediately aborted and terminated with an exception, and the targeted module not accessed.

10.5.2 Features

Each bus transfer can be classified by its privilege level and the reference type. The complete set of access types includes:

- Supervisor instruction fetch
- Supervisor operand read
- Supervisor operand write
- User instruction fetch
- User operand read
- User operand write

Instruction fetch accesses are associated with the execute attribute.

It should be noted that while the bus does not implement the concept of reference type (code versus data) and only supports the user/supervisor privilege level, the reference type attribute is supported by the system bus. Accordingly, the access checking associated with both privilege level and reference type is performed in the IPS controller using the attributes associated with the reference from the system bus.

The SACU partitions the access control mechanisms into three distinct functions:

- Master privilege register (MPR)
  — Allows each bus master to be assigned a privilege level:
    – Disable the master's user/supervisor attribute and force to user mode access
    – Enable the master's user/supervisor attribute
  — The reset state provides supervisor privilege to the processor core (bus master 0).
  — Input signals allow the non-core bus masters to have their user/supervisor attribute enabled at reset. This is intended to support the concept of a trusted bus master, and also controls the ability of a bus master to modify the register state of any of the SACU control registers; that is, only trusted masters can modify the control registers.

- Peripheral access control registers (PACRs)
  - Nine 8-bit registers control access to 17 of the on-chip peripheral modules.
  - Provides read/write access rights, supervisor/user privilege levels
  - Reset state provides supervisor-only read/write access to these modules
- Grouped peripheral access control registers (GPACR0, GPACR1)
  - One single register (GPACR0) controls access to 14 of the on-chip peripheral modules
  - One register (GPACR1) controls access for IPS reads and writes to the Flash module

— Provide read/write/execute access rights, supervisor/user privilege levels

— Reset state provides supervisor-only read/write access to each of these peripheral spaces

10.5.3   Memory Map/Register Definition

The memory map for the SACU program-visible registers within the System Control Module (SCM) is shown in Figure 10-7. The MPR, PACR, and GPACRs are 8 bits in width.

Table 10-7. SACU Register Memory Map

IPSBAR Offset	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x020	MPR		PPMRS		PPMRC		IPSBMT	
0x024	PACR0		PACR1		PACR2		PACR3	
0x028	PACR4		PACR5		PACR6		PACR7	
0x02c	PACR8		—		—		—	
0x030	GPACR0		GPACR1		—		—	
0x034	—		—		—		—	
0x038	—		—		—		—	
0x03C	—		—		—		—	

10.5.3.1   Master Privilege Register (MPR)

The MPR specifies the access privilege level associated with each bus master in the platform. The register provides one bit per bus master, where bit 3 corresponds to master 3 (Fast Ethernet Controller), bit 2 to master 2 (DMA Controller), bit 1 to master 1 (internal bus master), and bit 0 to master 0 (ColdFire core).

	7	0
Field	—	MPR[3:0]
Reset	0000_0011	
R/W	R/W	
Address	IPSBAR + 0x020	
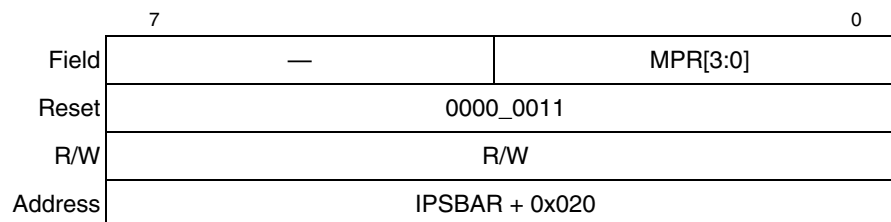
Figure 10-7. Master Privilege Register (MPR)

Table 10-8. MPR[n] Field Descriptions

Bits	Name	Description
7–4	—	Reserved. Should be cleared.
3–0	MPR	Each 1-bit field defines the access privilege level of the given bus master *n*. 0 All bus master accesses are in user mode. 1 All bus master accesses use the sourced user/supervisor attribute.

Only trusted bus masters can modify the access control registers. If a non-trusted bus master attempts to write any of the SACU control registers, the access is aborted with an error termination and the registers remain unaffected.

The processor core is connected to bus master 0 and is always treated as a trusted bus master. Accordingly, MPR[0] is forced to 1 at reset.

10.5.3.2 Peripheral Access Control Registers (PACR0–PACR8)

Access to several on-chip peripherals is controlled by shared peripheral access control registers. A single PACR defines the access level for each of the two modules. These modules only support operand reads and writes. Each PACR follows the format illustrated in Figure 10-8. For a list of PACRs and the modules that they control, refer to Table 10-11.

	7	6	4	3	2	0
Field	LOCK1	ACCESS_CTRL1		LOCK0	ACCESS_CTRL0	
Reset	0000_0000					
R/W	R/W					
Address	IPSBAR + 0x24 + Offset					

Figure 10-8. Peripheral Access Control Register (PACR*n*)

Table 10-9. PACR Field Descriptions

Bits	Name	Description
7	LOCK1	This bit, when set, prevents subsequent writes to ACCESSCTRL1. Any attempted write to the PACR generates an error termination and the contents of the register are not affected. Only a system reset clears this flag.
6–4	ACCESS_CTRL1	This 3-bit field defines the access control for the given platform peripheral. The encodings for this field are shown in Table 10-10.
3	LOCK0	This bit, when set, prevents subsequent writes to ACCESSCTRL0. Any attempted write to the PACR generates an error termination and the contents of the register are not affected. Only a system reset clears this flag.
2–0	ACCESS_CTRL0	This 3-bit field defines the access control for the given platform peripheral. The encodings for this field are shown in Table 10-10.

Table 10-10. PACR ACCESSCTRL Bit Encodings

Bits	Supervisor Mode	User Mode
000	Read/Write	No Access
001	Read	No Access
010	Read	Read
011	Read	No Access
100	Read/Write	Read/Write
101	Read/Write	Read

MCF5213 Reference Manual, Rev. 1.1

Table 10-10. PACR ACCESSCTRL Bit Encodings (continued)

Bits	Supervisor Mode	User Mode
110	Read/Write	Read/Write
111	No Access	No Access

Table 10-11. Peripheral Access Control Registers (PACRs)

IPSBAR Offset	Name	Modules Controlled	
		ACCESS_CTRL1	ACCESS_CTRL0
0x024	PACR0	SCM	—
0x025	PACR1	—	DMA
0x026	PACR2	UART0	UART1
0x027	PACR3	UART2	—
0x028	PACR4	I^{2}C	QSPI
0x029	—	—	—
0x02a	PACR6	DTIM0	DTIM1
0x02b	PACR7	DTIM2	DTIM3
0x02c	PACR8	INTC0	—
0x02d	—	—	—
0x02e	—	—	—

At reset, these on-chip modules are configured to have only supervisor read/write access capabilities. If an instruction fetch access to any of these peripheral modules is attempted, the IPS bus cycle is immediately terminated with an error.

10.5.3.3 Grouped Peripheral Access Control Registers (GPACR0 & GPACR1)

The on-chip peripheral space starting at IPSBAR is subdivided into sixteen 64-Mbyte regions. Each of the first two regions has a unique access control register associated with it. The other fourteen regions are in reserved space; the access control registers for these regions are not implemented. Bits [29:26] of the address select the specific GPACRn to be used for a given reference within the IPS address space. These access control registers are 8 bits in width so that read, write, and execute attributes may be assigned to the given IPS region.

NOTE

The access control for modules with memory space protected by PACR0–PACR8 are determined by the PACR0–PACR8 settings. The access control is not affected by GPACR0, even though the modules are mapped in its 64-Mbyte address space.

	7	6–4	3	0
Field	LOCK	—	ACCESS_CTRL	
Reset	0000_0000			
Read/Write	R/W	R	R/W	
Address	IPSBAR + 0x030, IPSBAR + 0x31			
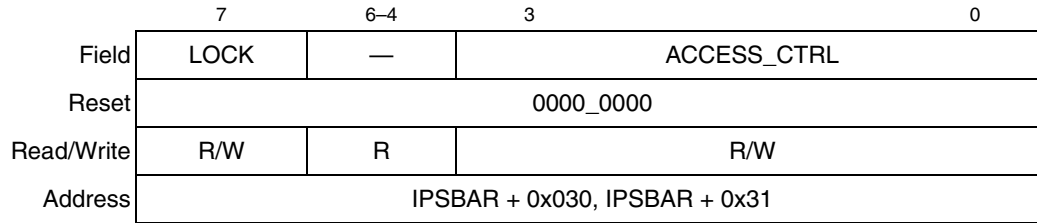
Figure 10-9. GPACR Register

Table 10-12. Grouped Peripheral Access Control Register (GPACR) Field Descriptions

Bits	Name	Description
7	LOCK	This bit, once set, prevents subsequent writes to the GPACR. Any attempted write to the GPACR generates an error termination and the contents of the register are not affected. Only a system reset clears this flag.
6–4	—	Reserved, should be cleared.
3–0	ACCESS_CTRL	This 4-bit field defines the access control for the given memory region. The encodings for this field are shown in Table 10-13.

At reset, these on-chip modules are configured to have only supervisor read/write access capabilities. Bit encodings for the ACCESS_CTRL field in the GPACR are shown in Table 10-13. Table 10-14 shows the memory space protected by the GPACRs and the modules mapped to these spaces.

Table 10-13. GPACR ACCESS_CTRL Bit Encodings

Bits	Supervisor Mode	User Mode
0000	Read / Write	No Access
0001	Read	No Access
0010	Read	Read
0011	Read	No Access
0100	Read / Write	Read / Write
0101	Read / Write	Read
0110	Read / Write	Read / Write
0111	No Access	No Access
1000	Read / Write / Execute	No Access
1001	Read / Execute	No Access
1010	Read / Execute	Read / Execute
1011	Execute	No Access
1100	Read / Write / Execute	Read / Write / Execute
1101	Read / Write / Execute	Read / Execute
1110	Read / Write	Read
1111	Read / Write / Execute	Execute

Table 10-14. GPACR Address Space

Register	Space Protected (IPSBAR Offset)	Modules Protected
GPACR0	0x0000_0000–0x03FF_FFFF	Ports, CCM, PMM, Reset controller, Clock, EPORT, WDOG, PIT0–PIT3, QADC, GPTA, GPTB, FlexCAN, CFM (Control)
GPACR1	0x0400_0000–0x07FF_FFFF	CFM (Flash module's backdoor access for programming or access by a bus master other than the core)

Chapter 11
General Purpose I/O Module

11.1   Introduction

Many of the pins associated with the external interface may be used for several different functions. Their primary function is to provide an external memory interface 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 pins. In some cases, the pin function is set by the operating mode, and the alternate pin functions are not supported.

The digital I/O pins are grouped into 8-bit ports. Some ports do not use all eight bits. Each port has registers that configure, monitor, and control the port pins. Figure 11-1 is a block diagram of the MCF5213 ports.

```
PORT DD      DDATA[3:0] / PDD[7:4]
             PST[3:0] / PDD[3:0]

PORT AS      SDA / PAS[1] / CANRX / RXD2
             SCL / PAS[0] / CANTX/TXD2

PORT QS      QSPI_PCS3 / PQS[6] / SYNCA / SYNCB
             QSPI_PCS2 / PQS[5]
             QSPI_PCS1 / PQS[4]
             QSPI_PCS0 / PQS[3] / SDA / CTS1
             QSPI_SCK / PQS[2] / SCL / RTS1
             QSPI_DIN / PQS[1] / CANRX / RXD1
             QSPI_DOUT / PQS[0] / CANTX RXD0

PORT TA      GPT[3] / PTA[3] / PWM7
             GPT[2] / PTA[2] / PWM5
             GPT[1] / PTA[1] / PWM3
             GPT[0] / PTA[0] / PWM1

PORT TD      PWM7 / PTD[3]
             PWM5 / PTD[2]
             PWM3 / PTD[1]
             PWM1 / PTD[0]

PORT TC      DTIN3 / PTC[3] / DTOUT3 / PWM6
             DTIN2 / PTC[2] / DTOUT2 / PWM4
             DTIN1 / PTC[1] / DTOUT1 / PWM2
             DTIN0 / PTC[0] / DTOUT0 / PWM0

PORT UC      CTS2 / PUC[3]
             RTS2 / PUC[2]
             RXD2 / PUC[1]
             TXD2 / PUC[0]

PORT UB      CTS1 / PUB[3] / SYNCA / RXD2
             RTS1 / PUB[2] / SYNCB / TXD2
             RXD1 / PUB[1]
             TXD1 / PUB[0]

PORT UA      CTS0 / PUA[3] / CANRX
             RTS0 / PUA[2] / CANTX
             RXD0 / PUA[1]
             TXD0 / PUA[0]

PORT AN      AN0 / PAN[0]
             AN1 / PAN[1]
             AN2 / PAN[2]
             AN3 / PAN[3]
             AN4 / PAN[4]
             AN5 / PAN[5]
             AN6 / PAN[6]
             AN7 / PAN[7]

PORT NQ      IRQ1 / PNQ[1] / SYNCA / PWM1
             IRQ2 / PNQ[2]
             IRQ3 / PNQ[3]
             IRQ4 / PNQ[4]
             IRQ5 / PNQ[5]
             IRQ6 / PNQ[6]
             IRQ7 / PNQ[7]
```

Figure 11-1. General Purpose I/O Module Block Diagram

11.2   Overview

The MCF5213 ports module controls the configuration for various external pins, including those used for:

- External bus accesses
- Chip selects
- Debug data
- Processor status
- FlexCAN transmit/receive data
- I^2C serial control
- QSPI
- UART transmit/receive
- 32-bit DMA timers

11.3   Features

The MCF5213 ports includes these distinctive features:

- Control of primary function use on all ports
- Digital I/O support for all ports
  - Registers for storing output pin data
  - Registers for controlling pin data direction
  - Registers for reading current pin state
  - Registers for setting and clearing output pin data registers

11.4   Signal Descriptions

Refer to Chapter 2, "Signal Descriptions," for more detailed information on the different signals and pins.

11.5   Memory Map/Register Definition

11.5.1   Ports Memory Map

Table 11-1 summarizes all the registers in the MCF5213 ports address space.

Table 11-1. MCF5213 Ports Module Memory Map

Offset[1]	31–24	23–16	15–8	7–0	Access[2]
Port Output Data Registers					
$0000	Reserved				S/U
$0004	Reserved				S/U
$0008	PORTNQ	PORTDD	PORTAN	PORTAS	S/U
$000C	Reserved	PORTQS	PORTTA	PORTTC	S/U
$0010	PORTTD	PORTUA	PORTUB	PORTUC	S/U
Port Data Direction Registers					
$0014	Reserved				S/U
$0018	Reserved				S/U
$001C	DDRNQ	DDRDD	DDRAN	DDRAS	S/U
$0020	Reserved	DDRQS	DDRTA	DDRTC	S/U
$0024	DDRTD	DDRUA	DDRUB	DDRUC	S/U
Port Pin Data/Set Data Registers					
$0028	Reserved				S/U
$002C	Reserved				S/U
$0030	PORTNQP/SETNQ	PORTDDP/SETDD	PORTANP/SETAN	PORTASP/SETAS	S/U
$0034	Reserved	PORTQSP/SETQS	PORTTAP/SETTA	PORTTCP/SETTC	S/U
$0038	PORTTDP/SETTD	PORTUAP/SETUA	PORTUBP/SETUB	PORTUCP/SETUC	S/U
Port Clear Output Data Registers					
$003C	Reserved				S/U
$0040	Reserved				S/U
$0044	CLRNQ	CLRDD	CLRAN	CLRAS	S/U
$0048	Reserved	CLRQS	CLRTA	CLRTC	S/U
$004C	CLRTD	CLRUA	CLRUB	CLRUC	S/U
Port Pin Assignment Registers					
$0050	PNQPAR	PDDPAR	PANPAR	PASPAR	S/U
$0054	PQSPAR		PTAPAR	PTCPAR	S/U
$0058	PTDPAR	PUAPAR	PUBPAR	PUCPAR	S/U
$005C–$0074	Reserved				S/U
Port Pad Control Registers					
$0078	PSRR[31:0]				S/U
$007C	PDSR[31:0]				S/U

[1]The register address is the sum of the IPSBAR address and the base address offset.

[2]S/U = supervisor or user mode access. User mode accesses to supervisor-only addresses have no effect and cause a cycle termination transfer error.

11.6 Register Descriptions

11.6.1 Port Output Data Registers (PORT*n*)

The PORT*n* registers store the data to be driven on the corresponding port *n* pins when the pins are configured for digital output.

The PORT*n* registers with a full 8-bit implementation, are shown in Figure 11-2. The remaining PORT*n* registers use fewer than eight bits. Their bit definitions are shown in Figure 11-3, Figure 11-4, Figure 11-5, and Figure 11-6.

The PORT*n* registers are read/write. At reset, all bits in the PORT*n* registers are set.

Reading a PORT*n* register returns the current values in the register, not the port *n* pin values.

PORT*n* bits can be set by setting the PORT*n* register, or by setting the corresponding bits in the PORT*n*P/SET*n* register. They can be cleared by clearing the PORT*n* register, or by clearing the corresponding bits in the CLR*n* register.

Base + $0009 (PORTDD)
Base + $000A (PORTAN)

	7	6	5	4	3	2	1	0
Field	PORT*n*7	PORT*n*6	PORT*n*5	PORT*n*4	PORT*n*3	PORT*n*2	PORT*n*1	PORT*n*0
Reset				1111_1111				
R/W				R/W				

Figure 11-2. PORT*n* — Port Output Data Registers [7:0]

Base + $000E (PORTTA)
Base + $000F (PORTTC)
Base + $0010 (PORTTD)
Base + $0011 (PORTUA)
Base + $0012 (PORTUB)
Base + $0013 (PORTUC)

	7	6	5	4	3	2	1	0
Field	—	—	—	—	PORT*n*3	PORT*n*2	PORT*n*1	PORT*n*0
Reset				0000_1111				
R/W		R				R/W		

Figure 11-3. PORT*n* — Port Output Data Registers [3:0]

Base + $000D (PORTQS)

	7	6	5	4	3	2	1	0
Field	—	PORT*n*6	PORT*n*5	PORT*n*4	PORT*n*3	PORT*n*2	PORT*n*1	PORT*n*0
Reset				0111_1111				
R/W	R				R/W			

Figure 11-4. PORT*n* — Port Output Data Registers [6:0]

Base + $0008 (PORTNQ)

	7	6	5	4	3	2	1	0
Field	PORT*n*7	PORT*n*6	PORT*n*5	PORT*n*4	PORT*n*3	PORT*n*2	PORT*n*1	—
Reset				1111_1110				
R/W				R/W				R

Figure 11-5. PORT*n* — Port Output Data Registers [7:1]

Base + $000B (PORTAS)

	7	6	5	4	3	2	1	0
Field	—	—	—	—	—	—	PORT*n*1	PORT*n*0
Reset				0000_0011				
R/W			R				R/W	

Figure 11-6. PORT*n* — Port Output Data Registers [1:0]

PORTn

> Port n Output Data Bits.
>
> Drive 1 when port n pin is digital output
>
> Drive 0 when port n pin is digital output

11.6.2   Port Data Direction Registers (DDR*n)*

The DDR*n* registers control the direction of the port *n* pin drivers when the pins are configured for digital I/O.

The DDR*n* registers are read/write. At reset, all bits in the DDR*n* registers are cleared to 0s.

Setting any bit in a DDR*n* register configures the corresponding port *n* pin as an output. Clearing any bit in a DDR*n* register configures the corresponding pin as an input.

Base + $001D (DDRDD)
Base + $001E (DDRAN)

	7	6	5	4	3	2	1	0
Field	DDR*n*7	DDR*n*6	DDR*n*5	DDR*n*4	DDR*n*3	DDR*n*2	DDR*n*1	DDR*n*0
Reset				0000_0000				
R/W				R/W				

Figure 11-7. DDR*n* — Port Data Direction Registers [7:0]

Base + $0022 (DDRTA)
Base + $0023 (DDRTC)
Base + $0024 (DDRTD)
Base + $0025 (DDRUA)
Base + $0026 (DDRUB)
Base + $0027 (DDRUC)

	7	6	5	4	3	2	1	0
Field	—	—	—	—	DDR*n*3	DDR*n*2	DDR*n*1	DDR*n*0
Reset				0000_0000				
R/W			R				R/W	

Figure 11-8. DDR*n* — Port Data Direction Registers [3:0]

Base + $0021 (DDRQS)

	7	6	5	4	3	2	1	0
Field	—	DDRn6	DDR*n*5	DDR*n*4	DDR*n*3	DDR*n*2	DDR*n*1	DDR*n*0
Reset				0000_0000				
R/W	R				R/W			

Figure 11-9. DDR*n* — Port Data Direction Registers [6:0]

Base + $001C (DDRNQ)

	7	6	5	4	3	2	1	0
Field	DDR*n*7	DDR*n*6	DDR*n*5	DDR*n*4	DDR*n*3	DDR*n*2	DDR*n*1	—
Reset				0000_0000				
R/W				R/W				R

Figure 11-10. DDR*n* — Port Data Direction Registers [7:1]

Base + $001F (DDRAS)

	7	6	5	4	3	2	1	0
Field	—	—	—	—	—	—	DDR*n*1	DDR*n*0
Reset				0000_0000				
R/W				R			R/W	

Figure 11-11. DDR*n* — Port Data Direction Registers [1:0]

DDR*n*

Port *n* Data Direction Bits.
1 = Port *n* pin configured as output
0 = Port *n* pin configured as input

MCF5213 Reference Manual, Rev. 1.1

Preliminary

11.6.3   Port Pin Data/Set Data Registers (PORT*n*P/SET*n*)

The PORT*n*P/SET*n* registers reflect the current pin states and control the setting of output pins when the pin is configured for digital I/O.

The PORT*n*P/SET*n* registers are read/write. At reset, the bits in the PORT*n*P/SET*n* registers are set to the current pin states.

Reading a PORT*n*P/SET*n* register returns the current state of the port *n* pins.

Writing 1s to a PORT*n*P/SET*n* register sets the corresponding bits in the PORT*n* register. Writing 0s has no effect.

Base + $0031 (PORTDD/SETDD)
Base + $0032 (PORTAN/SETAN)

	7	6	5	4	3	2	1	0
Field	PORT*n*P7	PORT*n*P6	PORT*n*P5	PORT*n*P4	PORT*n*P3	PORT*n*P2	PORT*n*P1	PORT*n*P0
Reset				1111_1111				
R/W				R/W				

Figure 11-12. PORT*n*P/SET*n* — Port Pin Data/Set Data Registers [7:0]

Base + $0036 (PORTTA/SETTA)
Base + $0037 (PORTTC/SETTC)
Base + $0038 (PORTTD/SETTD)
Base + $0039 (PORTUA/SETUA)
Base + $003A (PORTUB/SETUB)
Base + $003B (PORTUC/SETUC)

	7	6	5	4	3	2	1	0
Field	—	—	—	—	PORT*n*P3	PORT*n*P2	PORT*n*P1	PORT*n*P0
Reset				0000_1111				
R/W		R				R/W		

Figure 11-13. PORT*n*P/SET*n* — Port Pin Data/Set Data Registers [3:0]

Base + $0035 (PORTQS/SETQS)

	7	6	5	4	3	2	1	0
Field	—	PORT*n*P6	PORT*n*P5	PORT*n*P4	PORT*n*P3	PORT*n*P2	PORT*n*P1	PORT*n*P0
Reset				0111_1111				
R/W	R				R/W			

Figure 11-14. PORT*n*P/SET*n* — Port Pin Data/Set Data Registers [6:0]

Base + $0030 (PORTNQ/SETNQ)

	7	6	5	4	3	2	1	0
Field	PORT*n*P7	PORT*n*P6	PORT*n*P5	PORT*n*P4	PORT*n*P3	PORT*n*P2	PORT*n*P1	—
Reset				1111_1110				
R/W				R/W				R

Figure 11-15. PORT*n*P/SET*n* — Port Pin Data/Set Data Registers [7:1]

Base + $0033 (PORTAS/SETAS)

	7	6	5	4	3	2	1	0
Field	—	—	—	—	—	—	PORT*n*P1	PORT*n*P0
Reset				0000_0011				
R/W			R				R/W	

Figure 11-16. PORT*n*P/SET*n* — Port Pin Data/Set Data Registers [1:0]

PORT*n*P/SET*n*

   Port *n* Pin Data/Set Data Bits.
      1 = Port *n* pin state is 1 (read); set corresponding PORT*n* bit (write)
      0 = Port *n* pin state is 0 (read)

11.6.4   Port Clear Output Data Registers (CLR*n*)

Writing 0s to a CLR*n* register clears the corresponding bits in the PORT*n* register. Writing 1s has no effect. Reading the CLR*n* register returns 0s.

The CLR*n* registers are read/write.

Base + $0045 (CLRDD)
Base + $0046 (CLRAN)

	7	6	5	4	3	2	1	0
Field	CLR*n*7	CLR*n*6	CLR*n*5	CLR*n*4	CLR*n*3	CLR*n*2	CLR*n*1	CLR*n*0
Reset				0000_0000				
R/W				R/W				

Figure 11-17. CLRn — Port Clear Output Data Registers [7:0]

Base + $004A (CLRTA)
Base + $004B (CLRTC)
Base + $004C (CLRTD)
Base + $004D (CLRUA)
Base + $004E (CLRUB)
Base + $004F (CLRUC)

	7	6	5	4	3	2	1	0
Field	—	—	—	—	CLR*n*3	CLR*n*2	CLR*n*1	CLR*n*0
Reset				0000_0000				
R/W		R				R/W		

Figure 11-18. CLR*n* — Port Clear Output Data Registers [3:0]

Base + $0049 (CLRQS)

	7	6	5	4	3	2	1	0
Field	—	CLR*n*6	CLR*n*5	CLR*n*4	CLR*n*3	CLR*n*2	CLR*n*1	CLR*n*0
Reset				0000_0000				
R/W	R				R/W			

Figure 11-19. CLR*n* — Port Clear Output Data Registers [6:0]

Base + $0044 (CLRNQ)

	7	6	5	4	3	2	1	0
Field	CLR*n*7	CLR*n*6	CLR*n*5	CLR*n*4	CLR*n*3	CLR*n*2	CLR*n*1	—
Reset				0000_0000				
R/W				R/W				R

Figure 11-20. CLR*n* — Port Clear Output Data Registers [7:1]

Base + $0047 (CLRAS)

	7	6	5	4	3	2	1	0
Field	—	—	—	—	—	—	CLR*n*1	CLR*n*0
Reset				0000_0000				
R/W			R				R/W	

Figure 11-21. CLR*n* — Port Clear Output Data Registers [1:0]

CLR*n*

Port *n* Clear Output Data Register Bits.
1 = Never returned for reads; no effect for writes
0 = Always returned for reads; clears corresponding PORT*n* bit for writes

11.6.5 Pin Assignment Registers

All Pin Assignment Registers are read/write.

If multiple pins are configured for the one function then the result is undefined.

11.6.5.1 Dual Function Pin Assignment Registers

The dual function pin assignment registers allow each pin controlled by each register bit to be configured between the GPIO function and the Primary function.

Base + $0051 (PDDPAR)
Base + $0052 (PANPAR)

	7	6	5	4	3	2	1	0
Field	P*n*PAR7	P*n*PAR6	P*n*PAR5	P*n*PAR4	P*n*PAR3	P*n*PAR2	P*n*PAR1	P*n*PAR0
Reset	0000_0000							
R/W	R/W							

Figure 11-22. P*n*PAR — Port *n* Pin Assignment Registers Dual [7:0]

Base + $0058 (PTDPAR)
Base + $005B (PUCPAR)

	7	6	5	4	3	2	1	0
Field	—	—	—	—	P*n*PAR3	P*n*PAR2	P*n*PAR1	P*n*PAR0
Reset	0000_0000							
R/W	R				R/W			

Figure 11-23. P*n*PAR — Port *n* Pin Assignment Registers Dual [3:0]

P*n*PAR

Port *n* Pin Assignment Register Bits.
1 = Pin assumes the Primary function for that pin
0 = Pin assumes the GPIO function for that pin

11.6.5.2 Quad Function Pin Assignment Registers

The quad function pin assignment registers allow each pin controlled by each register bit to be configured between the GPIO function, Primary function, Alternate 1 function and the Alternate 2 function.

Base + $0054 (PQSPAR)

	15	14	13	12	11	10	9	8
Field	—	—	P*n*PAR6		P**n**PAR5		P*n*PAR4	
Reset	0000_0000							
R/W	R		R/W					

	7	6	5	4	3	2	1	0
Field	P*n*PAR3		P*n*PAR2		P*n*PAR1		P*n*PAR0	
Reset	0000_0000							
R/W	R/W							

Figure 11-24. P*n*PAR — Port *n* Pin Assignment Registers Quad [13:0]

Base + $0053 (PASPAR)

	7	6	5	4	3	2	1	0
Field	—	—	—	—	P*n*PAR1		P*n*PAR0	
Reset	0000_0000							
R/W	R				R/W			

Figure 11-25. P*n*PAR — Port *n* Pin Assignment Registers Quad [3:0]

Base + $0056 (PTAPAR)
Base + $0057 (PTCPAR)
Base + $0059 (PUAPAR)
Base + $005A (PUBPAR)

	7	6	5	4	3	2	1	0
Field	P*n*PAR3		P*n*PAR2		P*n*PAR1		P*n*PAR0	
Reset	0000_0000							
R/W	R/W							

Figure 11-26. P*n*PAR — Port *n* Pin Assignment Registers Quad [7:0]

P*n*PAR

Port *n* Pin Assignment Register Bits.

Table 11-2. Double Bit Pin Assignment Register Bit

Bit Value	Pin Assignment
00	GPIO function
01	Primary function
10	Alternate 1 function
11	Alternate 2 function

11.6.5.3 Port NQ Pin Assignment Register

The Port NQ Pin Assignment Register contains both quad function (for $\overline{\text{IRQ1}}$) and dual function pin assignment controls. Refer to the previous two sections for the encodings for the different fields. Note that the reset value of the PNQPAR register defaults to the primary function ($\overline{\text{IRQ}}$) instead of GPIO.

Base + $0050 (PNQPAR)

	7	6	5	4	3	2	1	0
Field	PNQPAR7	PNQPAR6	PNQPAR5	PNQPAR4	PNQPAR3	PNQPAR2	PNQPAR1	
Reset	1111_1101							
R/W	R/W							

Figure 11-27. PNQPAR — Port NQ Pin Assignment Register

11.6.6 Pad Control Registers

11.6.6.1 Pin Slew Rate Register

The Pin Slew Rate register is read/write and each bit resets to logic 0.

Base + $0078 (PSRR)

	31	30	29	28	27	26	25	24
Field	PSRR31	PSSR30	PSSR29	PSSR28	PSSR27	PSSR26	PSSR25	PSSR24
Reset					1			
R/W	R/W							
	23	22	21	20	19	18	17	16
Field	PSSR23	PSSR22	PSSR21	PSSR20	PSSR19	PSSR18	PSSR17	PSSR16
Reset					1			
R/W	R/W							
	15	14	13	12	11	10	9	8
Field	PSSR15	PSSR14	PSSR13	PSSR12	PSSR11	PSSR10	PSSR9	PSSR8
Reset					1			
R/W	R/W							
	7	6	5	4	3	2	1	0
Field	PSSR7	PSSR6	PSSR5	PSSR4	PSSR3	PSSR2	PSSR1	PSSR0
Reset					1			
R/W	R/W							

[1]Each bit resets to logic 0 in Single Chip mode and logic 1 in EzPort/FAST mode.

Figure 11-28. PSSR — Pin Slew Rate Register [31:0]

PSSR

    Pin Slew Rate Register control bits.
        1 = Pin is configured for slow slew rate (delay is approximately 10 times slower).
        0 = Pin is configured for fast slew rate.

11.6.6.2 Pin Drive Strength Register

The Pin Drive Strength register is read/write and each bit resets to logic 0 in Single Chip mode (MCF5213 default) and logic 1 in EzPort and FAST mode.

Base + $007C (PDSR)

	31	30	29	28	27	26	25	24
Field	PDSR31	PDSR30	PDSR29	PDSR28	PDSR27	PDSR26	PDSR25	PDSR24
Reset								
R/W				R/W				

	23	22	21	20	19	18	17	16
Field	PDSR23	PDSR22	PDSR21	PDSR20	PDSR19	PDSR18	PDSR17	PDSR16
Reset								
R/W				R/W				

	15	14	13	12	11	10	9	8
Field	PDSR15	PDSR14	PDSR13	PDSR12	PDSR11	PDSR10	PDSR9	PDSR8
Reset								
R/W				R/W				

	7	6	5	4	3	2	1	0
Field	PDSR7	PDSR6	PDSR5	PDSR4	PDSR3	PDSR2	PDSR1	PDSR0
Reset								
R/W				R/W				

[1]Each bit resets to logic 0 in Single Chip mode and logic 1 in EzPort/FAST mode.

Figure 11-29. PDSR — Pin Drive Strength Register [31:0]

PDSR

   Pin Drive Strength Register control bits.
      1 = Pin is configured for high drive strength (10mA).
      0 = Pin is configured for low drive strength (2mA).

11.7 Ports Interrupts

The ports module does not generate interrupt requests.

Chapter 12
Interrupt Controller Module

This section details the functionality for the interrupt controller. The general features of the interrupt controller include:

- 57 interrupt sources, organized as:
  - 50 fully-programmable interrupt sources
  - 7 fixed-level interrupt sources
- Each of the 57 sources has a unique interrupt control register (ICR*nx*) to define the software-assigned levels and priorities within the level
- Unique vector number for each interrupt source
- Ability to mask any individual interrupt source, plus global mask-all capability
- Supports both hardware and software interrupt acknowledge cycles
- "Wake-up" signal from low-power stop modes

The 57 fully-programmable and seven fixed-level interrupt sources for the interrupt controller handle the complete set of interrupt sources from all of the modules on the device. This section describes how the interrupt sources are mapped to the interrupt controller logic and how interrupts are serviced.

12.1   68K/ColdFire Interrupt Architecture Overview

Before continuing with the specifics of the interrupt controller, a brief review of the interrupt architecture of the 68K/ColdFire family is appropriate.

The interrupt architecture of ColdFire is exactly the same as the M68000 family, where there is 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 machine'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 require that, once asserted, the interrupt source remain asserted until explicitly disabled by the interrupt service routine.

During the interrupt exception processing, the CPU enters supervisor mode, disables trace mode and then fetches an 8-bit vector from the interrupt controller. This byte-sized operand fetch is known as the interrupt acknowledge (IACK) cycle with the ColdFire implementation using a special encoding of the transfer type and transfer modifier attributes to distinguish this data fetch from a "normal" memory access. The fetched data provides an index into the exception vector table which contains 256 addresses, each pointing to the beginning of a specific exception service routine. In particular, vectors 64 - 255 of the exception vector table are reserved for user interrupt service routines. The first 64 exception vectors are reserved for the processor to handle reset, error conditions (access, address), arithmetic faults, system calls, etc. 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 2 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 (see

Section 2.6, "Exception Stack Frame Definition" for more information on the stack frame format). 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. For many peripheral devices, the processing of the IACK cycle directly negates the interrupt request, while other devices require that request to be explicitly negated during the processing of the service routine.

For this device, , the processing of the interrupt acknowledge cycle is fundamentally different than previous 68K/ColdFire cores. In the new approach, all IACK cycles are directly handled by the interrupt controller, so the requesting peripheral device is not accessed during the IACK. As a result, the interrupt request must be explicitly cleared in the peripheral during the interrupt service routine. For more information, see Section 12.1.1.3, "Interrupt Vector Determination."

Unlike the M68000 family, all ColdFire processors guarantee that the first instruction of the service routine is executed before sampling for interrupts is resumed. By making this initial instruction a load of the SR, interrupts can be safely disabled, if required.

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* at http://www.freescale.com/coldfire.

12.1.1   Interrupt Controller Theory of Operation

To support the interrupt architecture of the 68K/ColdFire programming model, the combined 63 interrupt sources are organized as 7 levels, with each level supporting up to 9 prioritized requests. Consider the priority structure within a single interrupt level (from highest to lowest priority) as shown in Table 12-1.

Table 12-1. Interrupt Priority Within a Level

ICR[2:0]	Priority	Interrupt Sources
111	7 (Highest)	8-63
110	6	8-63
101	5	8-63
100	4	8-63
—	Fixed Midpoint Priority	1-7
011	3	8-63
010	2	8-63
001	1	8-63
000	0 (Lowest)	8-63

The level and priority is fully programmable for all sources except interrupt sources 1–7. Interrupt source 1–7 (from the Edgeport module) are fixed at the corresponding level's midpoint priority. Thus, a maximum of 8 fully-programmable interrupt sources are mapped into a single interrupt level. The "fixed" interrupt source is hardwired to the given level, and represents the mid-point of the priority within the level. For the

fully-programmable interrupt sources, the 3-bit level and the 3-bit priority within the level are defined in the 8-bit interrupt control register (ICR*nx*).

The operation of the interrupt controller can be broadly partitioned into three activities:

- Recognition
- Prioritization
- Vector Determination during IACK

12.1.1.1 Interrupt Recognition

The interrupt controller continuously examines the request sources and the interrupt mask register to determine if there are active requests. This is the recognition phase.

12.1.1.2 Interrupt Prioritization

As an active request is detected, it is translated into the programmed interrupt level, and the resulting 7-bit decoded priority level (IRQ[7:1]) is driven out of the interrupt controller.

12.1.1.3 Interrupt Vector Determination

Once the core has sampled for pending interrupts and begun interrupt exception processing, it generates an interrupt acknowledge cycle (IACK). The IACK transfer is treated as a memory-mapped byte read by the processor, and routed to the appropriate interrupt controller. Next, the interrupt controller extracts the level being acknowledged from address bits[4:2], and then determines the highest priority interrupt request active for that level, and returns the 8-bit interrupt vector for that request to complete the cycle. The 8-bit interrupt vector is formed using the following algorithm:

```
For INTC0,              vector_number = 64 + interrupt source number
```

Recall vector_numbers 0 - 63 are reserved for the ColdFire processor and its internal exceptions. Thus, the following mapping of bit positions to vector numbers applies for the INTC0:

```
if interrupt source 1 is active and acknowledged,      then vector_number =  65

if interrupt source 2 is active and acknowledged,      then vector_number =  66

...

if interrupt source 8 is active and acknowledged,      then vector_number =  72

if interrupt source 9 is active and acknowledged,      then vector_number =  73

...

if interrupt source 62 is active and acknowledged,     then vector_number = 126
```

The net effect is a fixed mapping between the bit position within the source to the actual interrupt vector number.

If there is no active interrupt source for the given level, a special "spurious interrupt" vector (vector_number = 24) is returned and it is the responsibility of the service routine to handle this error situation.

Note this protocol implies the interrupting peripheral is not accessed during the acknowledge cycle since the interrupt controller completely services the acknowledge. This means the interrupt source must be explicitly disabled in the interrupt service routine. This design provides unique vector capability for all interrupt requests, regardless of the "complexity" of the peripheral device.

12.2   Memory Map

The register programming model for the interrupt controllers is memory-mapped to a 256-byte space. In the following discussion, there are a number of program-visible registers greater than 32 bits in size. For these control fields, the physical register is partitioned into two 32-bit values: a register "high" (the upper longword) and a register "low" (the lower longword). The nomenclature <reg_name>H and <reg_name>L is used to reference these values.

The registers and their locations are defined in Table 12-3. The offsets listed start from the base address for each interrupt controller. The base addresses for the interrupt controllers are listed below:

Table 12-2. Interrupt Controller Base Addresses

Interrupt Controller Number	Base Address
INTC	IPSBAR + 0xC00

Table 12-3. Interrupt Controller Memory Map

Module Offset	Bits[31:24]	Bits[23:16]	Bits[15:8]	Bits[7:0]
IPSBAR + 0x0C00	Interrupt Pending Register High (IPRH), [63:32]			
IPSBAR + 0x0C04	Interrupt Pending Register Low (IPRL), [31:1]			
IPSBAR + 0x0C08	Interrupt Mask Register High (IMRH), [63:32]			
IPSBAR + 0x0C0c	Interrupt Mask Register Low (IMRL), [31:0]			
IPSBAR + 0x0C10	Interrupt Force Register High (INTFRCH), [63:32]			
IPSBAR + 0x0C14	Interrupt Force Register Low (INTFRCL), [31:1]			
IPSBAR + 0x0C18	IRLR[7:1]	IACKLPR[7:0]	Reserved	
IPSBAR + 0x0C1C - IPSBAR + 0x0C3C	Reserved			
IPSBAR + 0x0C40	Reserved	ICR01	ICR02	ICR03
IPSBAR + 0x0C44	ICR04	ICR05	ICR06	ICR07
IPSBAR + 0x0C48	ICR08	ICR09	ICR10	ICR11
IPSBAR + 0x0Cx4C	ICR12	ICR13	ICR14	ICR15
IPSBAR + 0x0C50	ICR16	ICR17	ICR18	ICR19
IPSBAR + 0x0C54	ICR20	ICR21	ICR22	ICR23
IPSBAR + 0x0C58	ICR24	ICR25	ICR26	ICR27
IPSBAR + 0x0C5C	ICR28	ICR29	ICR30	ICR31
IPSBAR + 0x0C60	ICR32	ICR33	ICR34	ICR35
IPSBAR + 0x0C64	ICR36	ICR37	ICR38	ICR39
IPSBAR + 0x0C68	ICR40	ICR41	ICR42	ICR43
IPSBAR + 0x0C6C	ICR44	ICR45	ICR46	ICR47
IPSBAR + 0x0C70	ICR48	ICR49	ICR50	ICR51
IPSBAR + 0x0C74	ICR52	ICR53	ICR54	ICR55

MCF5213 Reference Manual, Rev. 1.1

Table 12-3. Interrupt Controller Memory Map (continued)

Module Offset	Bits[31:24]	Bits[23:16]	Bits[15:8]	Bits[7:0]
IPSBAR + 0x0C78	ICR56	ICR57	ICR58	ICR59
IPSBAR + 0x0C7C	ICR60	ICR61	ICR62	ICR63
IPSBAR + 0x0C80 IPSBAR + 0x0CDC	Reserved			
IPSBAR + 0x0CE0	SWIACK	Reserved		
IPSBAR + 0x0CE4	L1IACK	Reserved		
IPSBAR + 0x0CE8	L2IACK	Reserved		
IPSBAR + 0x0CEC	L3IACK	Reserved		
IPSBAR + 0x0CF0	L4IACK	Reserved		
IPSBAR + 0x0CF4	L5IACK	Reserved		
IPSBAR + 0x0CF8	L6IACK	Reserved		
IPSBAR + 0x0CFC	L7IACK	Reserved		

12.3   Register Descriptions

12.3.1   Interrupt Pending Registers (IPRH*n*, IPRL*n*)

The IPRH*n* and IPRL*n* registers, Figure 12-1 and Figure 12-2, are each 32 bits in size, and provide a bit map for each interrupt request to indicate if there is an active request (1 = active request, 0 = no request) for the given source. The state of the interrupt mask register does not affect the IPR*n*. The IPR*n* is cleared by reset. The IPR*n* is a read-only register, so any attempted write to this register is ignored. Bit 0 is not implemented and reads as a zero.

	31	16
Field	INT[63:48]	
Reset	0000_0000_0000_0000	
R/W	R	

	15	0
Field	INT[47:32]	
Reset	0000_0000_0000_0000	
R/W	R	
	IPSBAR + 0xC00	
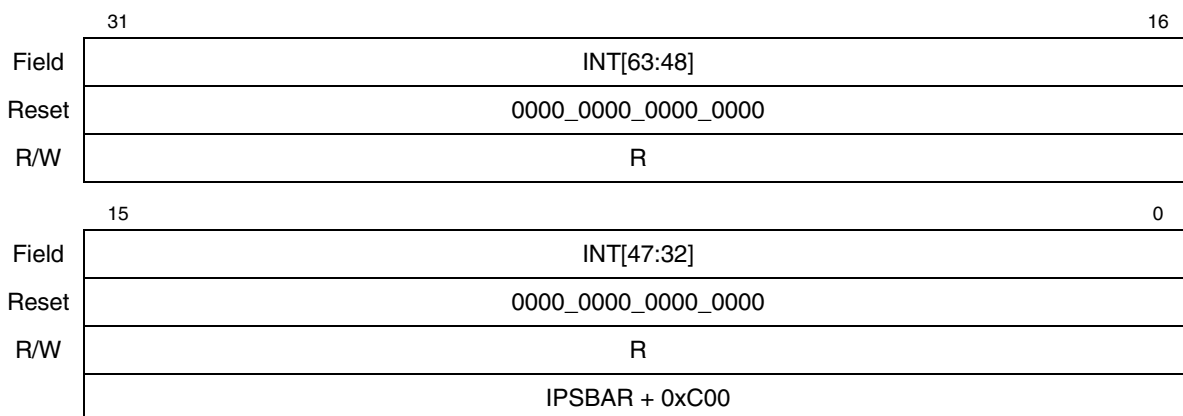
Figure 12-1. Interrupt Pending Register High (IPRH*n*)

MCF5213 Reference Manual, Rev. 1.1

Preliminary

Table 12-4. IPRH*n* Field Descriptions

Bits	Name	Description
31–0	INT	Interrupt pending. Each bit corresponds to an interrupt source. The corresponding IMRH*n* bit determines whether an interrupt condition can generate an interrupt. At every system clock, the IPRH*n* samples the signal generated by the interrupting source. The corresponding IPRH*n* bit reflects the state of the interrupt signal even if the corresponding IMRH*n* bit is set. 0 The corresponding interrupt source does not have an interrupt pending 1 The corresponding interrupt source has an interrupt pending

	31		16
Field		INT[31:16]	
Reset		0000_0000_0000_0000	
R/W		R	

	15		1	0
Field		INT[16:1]		—
Reset		0000_0000_0000_0000		
R/W		R		
		IPSBAR + 0xC04		
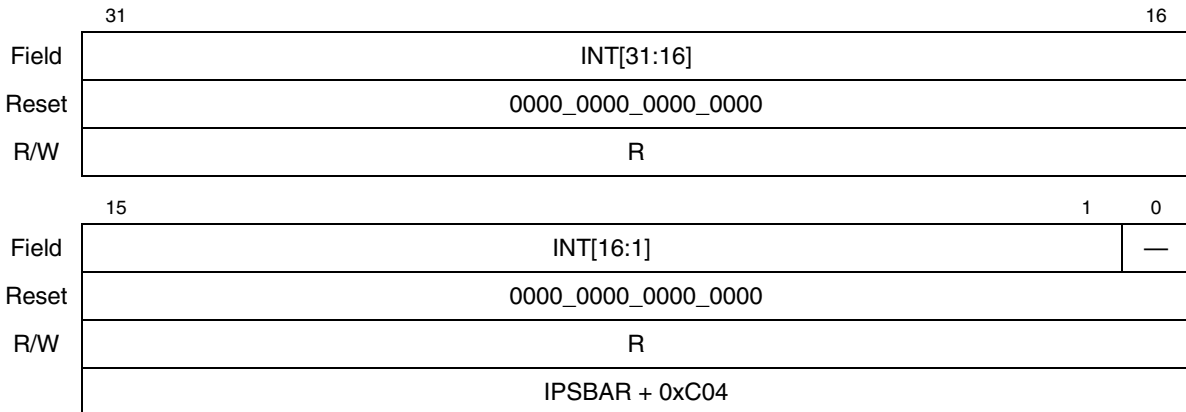
Figure 12-2. Interrupt Pending Register Low (IPRL*n*)

Table 12-5. IPRL*n* Field Descriptions

Bits	Name	Description
31–1	INT	Interrupt Pending. Each bit corresponds to an interrupt source. The corresponding IMRL*n* bit determines whether an interrupt condition can generate an interrupt. At every system clock, the IPRL*n* samples the signal generated by the interrupting source. The corresponding IPRL*n* bit reflects the state of the interrupt signal even if the corresponding IMRL*n* bit is set. 0 The corresponding interrupt source does not have an interrupt pending 1 The corresponding interrupt source has an interrupt pending
0	—	Reserved, should be cleared.

12.3.2 Interrupt Mask Register (IMRH*n*, IMRL*n*)

The IMRH*n* and IMRL*n* registers are each 32 bits in size and provide a bit map for each interrupt to allow the request to be disabled (1 = disable the request, 0 = enable the request). The IMR*n* is set to all ones by reset, disabling all interrupt requests. The IMR*n* can be read and written. A write that sets bit 0 of the IMR forces the other 63 bits to be set, disabling all interrupt sources, and providing a global mask-all capability.

	31			16
Field		INT_MASK[63:48]		
Reset		1111_1111_1111_1111		
R/W		R/W		

	15			0
Field		INT_MASK[47:32]		
Reset		1111_1111_1111_1111		
R/W		R/W		
		IPSBAR + 0xC08		
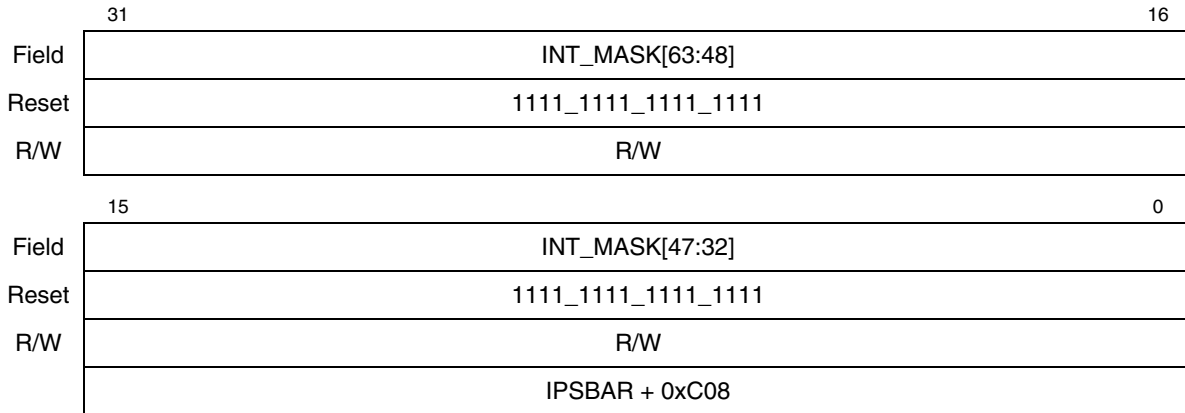
Figure 12-3. Interrupt Mask Register High (IMRH*n*)

Table 12-6. IMRH*n* Field Descriptions

Bits	Name	Description
31–0	INT_MASK	Interrupt mask. Each bit corresponds to an interrupt source. The corresponding IMRH*n* bit determines whether an interrupt condition can generate an interrupt. The corresponding IPRH*n* bit reflects the state of the interrupt signal even if the corresponding IMRH*n* bit is set. 0 The corresponding interrupt source is not masked 1 The corresponding interrupt source is masked

	31			16
Field		INT_MASK[31:16]		
Reset		1111_1111_1111_1111		
R/W		R/W		

	15		1	0
Field		INT_MASK[16:1]		MASKALL
Reset		1111_1111_1111_1111		
R/W		R/W		
		IPSBAR + 0xC0C		

Figure 12-4. Interrupt Mask Register Low (IMRL*n*)

Table 12-7. IMRL*n* Field Descriptions

Bits	Name	Description
31–1	INT_MASK	Interrupt mask. Each bit corresponds to an interrupt source. The corresponding IMRL*n* bit determines whether an interrupt condition can generate an interrupt. The corresponding IPRL*n* bit reflects the state of the interrupt signal even if the corresponding IMRL*n* bit is set. 0 The corresponding interrupt source is not masked 1 The corresponding interrupt source is masked
0	MASKALL	Mask all interrupts. Setting this bit will force the other 63 bits of the IMRH*n* and IMRL*n* to ones, disabling all interrupt sources, and providing a global mask-all capability.

NOTE

If an interrupt source is being masked in the interrupt controller mask register (IMR) or a module's interrupt mask register while the interrupt mask in the status register (SR[I]) is set to a value lower than the interrupt's level, a spurious interrupt may occur. This is because by the time the status register acknowledges this interrupt, the interrupt has been masked. A spurious interrupt is generated because the CPU cannot determine the interrupt source. To avoid this situation for interrupts sources with levels 1-6, first write a higher level interrupt mask to the status register, before setting the mask in the IMR or the module's interrupt mask register. After the mask is set, return the interrupt mask in the status register to its previous value. Since level seven interrupts cannot be disabled in the status register prior to masking, use of the IMR or module interrupt mask registers to disable level seven interrupts is not recommended.

12.3.3   Interrupt Force Registers (INTFRCH*n*, INTFRCL*n*)

The INTFRCH*n* and INTFRCL*n* registers are each 32 bits in size and provide a mechanism to allow software generation of interrupts for each possible source for functional or debug purposes. The system design may reserve one or more sources to allow software to self-schedule interrupts by forcing one or more of these bits (1 = force request, 0 = negate request) in the appropriate INTFRC*n* register. The assertion of an interrupt request via the INTFRC*n* register is not affected by the interrupt mask register. The INTFRC*n* register is cleared by reset.

	31															16
Field	INTFRCH[63:48]															
Reset	0000_0000_0000_0000															
R/W	R/W															

	15															0
Field	INTFRCH[47:32]															
Reset	0000_0000_0000_0000															
R/W	R/W															
	IPSBAR + 0xC10															
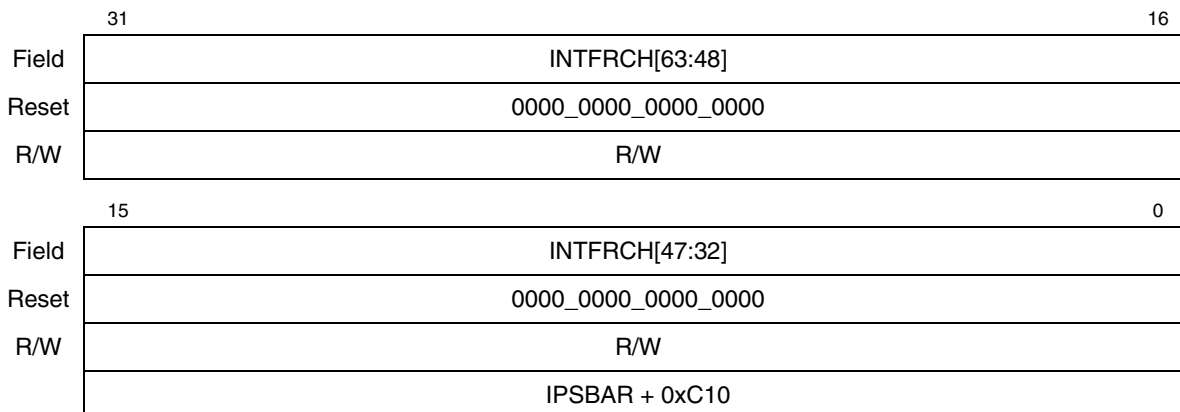
Figure 12-5. Interrupt Force Register High (INTFRCH*n*)

Table 12-8. INTFRCH*n* Field Descriptions

Bits	Name	Description
31–0	INTFRC	Interrupt force. Allows software generation of interrupts for each possible source for functional or debug purposes. 0 No interrupt forced on corresponding interrupt source 1 Force an interrupt on the corresponding source

	31															16
Field	INTFRCL[31:16]															
Reset	0000_0000_0000_0000															
R/W	R/W															

	15														1	0
Field	INTFRCL[16:1]															—
Reset	0000_0000_0000_0000															
R/W	R/W															
	IPSBAR + 0xC14															

Figure 12-6. Interrupt Force Register Low (INTFRCL*n*)

Table 12-9. INTFRCL*n* Field Descriptions

Bits	Name	Description
31–1	INTFRC	Interrupt force. Allows software generation of interrupts for each possible source for functional or debug purposes. 0 No interrupt forced on corresponding interrupt source 1 Force an interrupt on the corresponding source
0	—	Reserved, should be cleared.

12.3.4 Interrupt Request Level Register (IRLR*n*)

This 7-bit register is updated each machine cycle and represents the current interrupt requests for each interrupt level, where bit 7 corresponds to level 7, bit 6 to level 6, etc.

	7	2	1	0
Field	IRQ[7:1]			—
Reset	0000_0000			
R/W	R			
Address	IPSBAR + 0xC18			
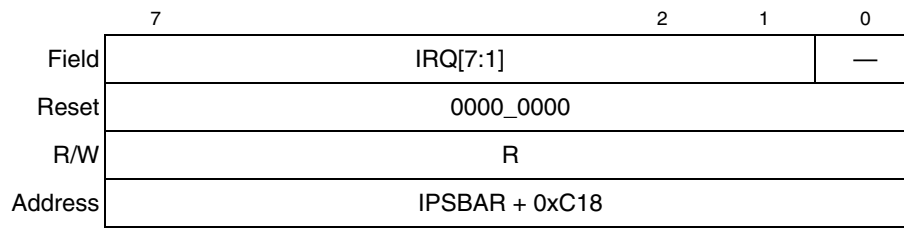
Figure 12-7. Interrupt Request (IR*n*)

Table 12-10. IRQ*n* Field Descriptions

Bits	Name	Description
7–1	IRQ	Interrupt requests. Represents the prioritized active interrupts for each level. 0 There are no active interrupts at this level 1 There is an active interrupt at this level
0	—	Reserved

12.3.5 Interrupt Acknowledge Level and Priority Register (IACKLPR*n*)

Each time an IACK is performed, the interrupt controller responds with the vector number of the highest priority source within the level being acknowledged. In addition to providing the vector number directly for the byte-sized IACK read, this 8-bit register is also loaded with information about the interrupt level and priority being acknowledged. This register provides the association between the acknowledged "physical" interrupt request number and the programmed interrupt level/priority. The contents of this read-only register are described in Figure 12-8 and Table 12-11.

	7	6	4	3	0
Field	—	LEVEL		PRI	
Reset	0000_0000				
R/W	R				
Address	IPSBAR + 0xC19				
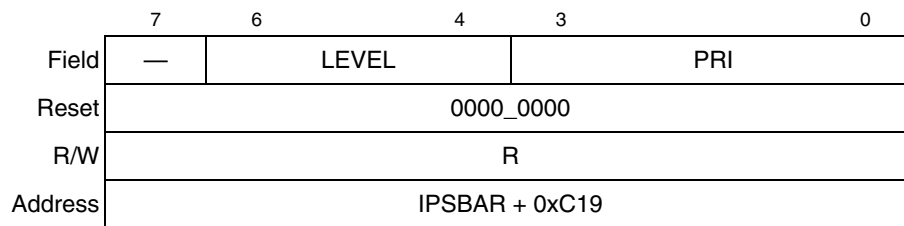
Figure 12-8. IACK Level and Priority Register (IACKLPR*n*)

Table 12-11. IACKLPR*n* Field Descriptions

Bits	Name	Description
7	—	Reserved

Table 12-11. IACKLPR*n* Field Descriptions (continued)

Bits	Name	Description
6–4	LEVEL	Interrupt level. Represents the interrupt level currently being acknowledged.
3–0	PRI	Interrupt Priority. Represents the priority within the interrupt level of the interrupt currently being acknowledged. 0 Priority 0 1 Priority 1 2 Priority 2 3 Priority 3 4 Priority 4 5 Priority 5 6 Priority 6 7 Priority 7 8 Mid-Point Priority associated with the fixed level interrupts only

12.3.6   Interrupt Control Register (ICR*nx*, (*x* = 1, 2,..., 63))

Each ICR*nx* specifies the interrupt level (1-7) and the priority within the level (0-7). All ICR*nx* registers can be read, but only ICR*n*8 to ICR*n*63 can be written. It is software's responsibility to program the ICR*nx* registers with unique and non-overlapping level and priority definitions. Failure to program the ICR*nx* registers in this manner can result in undefined behavior. If a specific interrupt request is completely unused, the ICR*nx* value can remain in its reset (and disabled) state.

	7	6	5	3	2	0
Field	—		IL		IP	
Reset	0000_0000					
R/W	R/W (Read only for ICR*n*1-ICR*n*7)					
Address	See Table 12-2 and Table 12-3 for register offsets					

Figure 12-9. Interrupt Control Register (ICR*nx*)

Table 12-12. ICR*nx* Field Descriptions

Bits	Name	Description
7–6	—	Reserved, should be cleared.
5–3	IL	Interrupt level. Indicates the interrupt level assigned to each interrupt input.
2–0	IP	Interrupt priority. Indicates the interrupt priority for internal modules within the interrupt-level assignment. 000b represents the lowest priority and 111b represents the highest. For the fixed level interrupt sources, the priority is fixed at the midpoint for the level, and the IP field will always read as 000b.

12.3.6.1   Interrupt Sources

Table 10-13 lists the interrupt sources for each interrupt request line

Table 12-13. Interrupt Source Assignment For Interrupt Controller 0

Source	Module	Flag	Source Description	Flag Clearing Mechanism
1	EPORT	EPF1	Edge port flag 1	Write EPF1 = 1
2		EPF2	Edge port flag 2	Write EPF2 = 1
3		EPF3	Edge port flag 3	Write EPF3 = 1
4		EPF4	Edge port flag 4	Write EPF4 = 1
5		EPF5	Edge port flag 5	Write EPF5 = 1
6		EPF6	Edge port flag 6	Write EPF6 = 1
7		EPF7	Edge port flag 7	Write EPF7 = 1
8	SCM	SWTI	Software watchdog timeout	Cleared when service complete.
9	DMA	DONE	DMA Channel 0 transfer complete	Write DONE = 1
10		DONE	DMA Channel 1 transfer complete	Write DONE = 1
11		DONE	DMA Channel 2 transfer complete	Write DONE = 1
12		DONE	DMA Channel 3 transfer complete	Write DONE = 1
13	UART0	INT	UART0 interrupt	Automatically cleared
14	UART1	INT	UART1 interrupt	Automatically cleared
15	UART2	INT	UART2 interrupt	Automatically cleared
16	Not used (Reserved)			
17	I^2C	IIF	I^2C interrupt	Write IIF = 0
18	QSPI	INT	QSPI interrupt	Write 1 to appropriate QIR bit
19	DTIM0	INT	DTIM0 interrupt	Write 1 to appropriate DTER0 bit
20	DTIM1	INT	DTIM1 interrupt	Write 1 to appropriate DTER1 bit
21	DTIM2	INT	DTIM2 interrupt	Write 1 to appropriate DTER2 bit
22	DTIM3	INT	DTIM3 interrupt	Write 1 to appropriate DTER3 bit

Table 12-13. Interrupt Source Assignment For Interrupt Controller 0 (continued)

Source	Module	Flag	Source Description	Flag Clearing Mechanism
23	FLEXCAN	BUF0I	Message Buffer 0 Interrupt	Write 1 to BUF0I after reading as 1
24		BUF1I	Message Buffer 1 Interrupt	Write 1 to BUF1I after reading as 1
25		BUF2I	Message Buffer 2 Interrupt	Write 1 to BUF2I after reading as 1
26		BUF3I	Message Buffer 3 Interrupt	Write 1 to BUF3I after reading as 1
27		BUF4I	Message Buffer 4 Interrupt	Write 1 to BUF4I after reading as 1
28		BUF5I	Message Buffer 5 Interrupt	Write 1 to BUF5I after reading as 1
29		BUF6I	Message Buffer 6 Interrupt	Write 1 to BUF6I after reading as 1
30		BUF7I	Message Buffer 7 Interrupt	Write 1 to BUF7I after reading as 1
31		BUF8I	Message Buffer 8 Interrupt	Write 1 to BUF8I after reading as 1
32		BUF9I	Message Buffer 9 Interrupt	Write 1 to BUF9I after reading as 1
33		BUF10I	Message Buffer 10 Interrupt	Write 1 to BUF10I after reading as 1
34		BUF11I	Message Buffer 11 Interrupt	Write 1 to BUF11I after reading as 1
35		BUF12I	Message Buffer 12 Interrupt	Write 1 to BUF12I after reading as 1
36		BUF13I	Message Buffer 13 Interrupt	Write 1 to BUF13I after reading as 1
37		BUF14I	Message Buffer 14 Interrupt	Write 1 to BUF14I after reading as 1
38		BUF15I	Message Buffer 15 Interrupt	Write 1 to BUF15I after reading as 1
39		ERR_INT	Error Interrupt	Read reported error bits in ESR or write 0 to ERR_INT
40		BOFF_INT	Bus-Off Interrupt	Write 0 to BOFF_INT
41	GPT	TOF	Timer overflow	Write TOF = 1 or access TIMCNTH/L if TFFCA = 1
42		PAIF	Pulse accumulator input	Write PAIF = 1 or access PAC if TFFCA = 1
43		PAOVF	Pulse accumulator overflow	Write PAOVF = 1 or access PAC if TFFCA = 1
44		C0F	Timer channel 0	Write C0F = 1 or access IC/OC if TFFCA = 1
45		C1F	Timer channel 1	Write 1 to C1F or access IC/OC if TFFCA = 1
46		C2F	Timer channel 2	Write 1 to C2F or access IC/OC if TFFCA = 1
47		C3F	Timer channel 3	Write 1 to C3F or access IC/OC if TFFCA = 1
48	PMM	LVDF	LVD	Write LVDF = 1
49	ADC	ADCA	ADCA conversion complete	Write 1 to EOSI0
50		ADCB	ADCB conversion complete	Write 1 to EOSI1
51		ADCINT	ADC Interrupt	Wrte 1 to ZCI, LLMTI and HLMTI
52	Not used (Reserved)			
53	Not used (Reserved)			

MCF5213 Reference Manual, Rev. 1.1

Preliminary

Table 12-13. Interrupt Source Assignment For Interrupt Controller 0 (continued)

Source	Module	Flag	Source Description	Flag Clearing Mechanism
54	Not used (Reserved)			
55	PIT0	PIF	PIT interrupt flag	Write PIF = 1 or write PMR
56	PIT1	PIF	PIT interrupt flag	Write PIF = 1 or write PMR
57	Not Used (Reserved)			
58	Not Used (Reserved)			
59	CFM	CBEIF	SGFM buffer empty	Write CBEIF = 1
60	CFM	CCIF	SGFM command complete	Cleared automatically
61	CFM	PVIF	Protection violation	Cleared automatically
62	CFM	AEIF	Access error	Cleared automatically
63	PWM	PWM	PWM Interrupt	Write PWMIF = 1

12.3.7 Software and Level n IACK Registers (SWIACKR, L1IACK–L7IACK)

The eight IACK registers can be explicitly addressed via the CPU, or implicitly addressed via a processor-generated interrupt acknowledge cycle during exception processing. In either case, the interrupt controller's actions are very similar.

First, consider an IACK cycle to a specific level: that is, a level-n IACK. When this type of IACK arrives in the interrupt controller, the controller examines all the 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. In addition to providing the vector number, the interrupt controller also loads the level and priority number for the level into the IACKLPR register, where it may be retrieved later.

This interrupt controller design also supports the concept of a software IACK. A software IACK is a useful concept that allows an interrupt service routine to determine if there are other pending interrupts so that the overhead associated with interrupt exception processing (including machine state save/restore functions) can be minimized. In general, the software IACK is performed near the end of an interrupt service routine, and if there are additional active interrupt sources, the current interrupt service routine (ISR) passes control to the appropriate service routine, but without taking another interrupt exception.

When the interrupt controller receives a software IACK read, it returns the vector number associated with the highest level, highest priority unmasked interrupt source for that interrupt controller. The IACKLPR register is also loaded as the software IACK is performed. If there are no active sources, the interrupt controller returns an all-zero vector as the operand. For this situation, the IACKLPR register is also cleared.

In addition to the software IACK registers within each interrupt controller, there are global software IACK registers. A read from the global SWIACK will return the vector number for the highest level and priority unmasked interrupt source from all interrupt controllers. A read from one of the L*n*IACK registers will return the vector for the highest priority unmasked interrupt within a level for all interrupt controllers.

	7	6		4	3		0
Field				VECTOR			
Reset				0000_0000			
R/W				R			
Address				See Table 12-2 and Table 12-3 for register offsets			

Figure 12-10. Software and Level *n* IACK Registers (SWIACKR, L1IACK–L7IACK)

Table 12-14. SWIACK and L1IACK-L7IACK Field Descriptions

Bits	Name	Description
7–0	VECTOR	Vector number. A read from the SWIACK register returns the vector number associated with the highest level, highest priority unmasked interrupt source. A read from one of the L*n*ACK registers returns the highest priority unmasked interrupt source within the level.

12.4   Low-Power Wakeup Operation

The System Control Module (SCM) contains an 8-bit low-power interrupt control register (LPICR) used explicitly for controlling the low-power stop mode. This register must explicitly be programmed by software to enter low-power mode.

The interrupt controller provides a special combinatorial logic path to provide a special wake-up signal to exit from the low-power stop mode. This special mode of operation works as follows:

*   First, LPICR[6:4] is loaded with the mask level that will be specified while the core is in stop mode. LPICR[7] must be set to enable this mode of operation.

NOTE

The wakeup mask level taken from LPICR[6:4] is adjusted by hardware to allow a level 7 IRQ to generate a wakeup. That is, the wakeup mask value used by the interrupt controller must be in the range of 0–6.

*   Second, the processor executes a STOP instruction which places it in stop mode. Once the processor is stopped, each interrupt controller enables a special logic path which evaluates the incoming interrupt sources in a purely combinatorial path; that is, there are no clocked storage elements. If an active interrupt request is asserted and the resulting interrupt level is greater than the mask value contained in LPICR[6:4], then the interrupt controller asserts the wake-up output signal, which is routed to the SCM and then to the PLL module to re-enable the device's clock trees and resume processing.

Chapter 13
Edge Port Module (EPORT)

13.1   Introduction

The edge port module (EPORT) has seven external interrupt pins, $\overline{\text{IRQ7}}$–$\overline{\text{IRQ1}}$. Each pin can be configured individually as a level-sensitive interrupt pin, an edge-detecting interrupt pin (rising edge, falling edge, or both), or a general-purpose input/output (I/O) pin. See Figure 13-1.
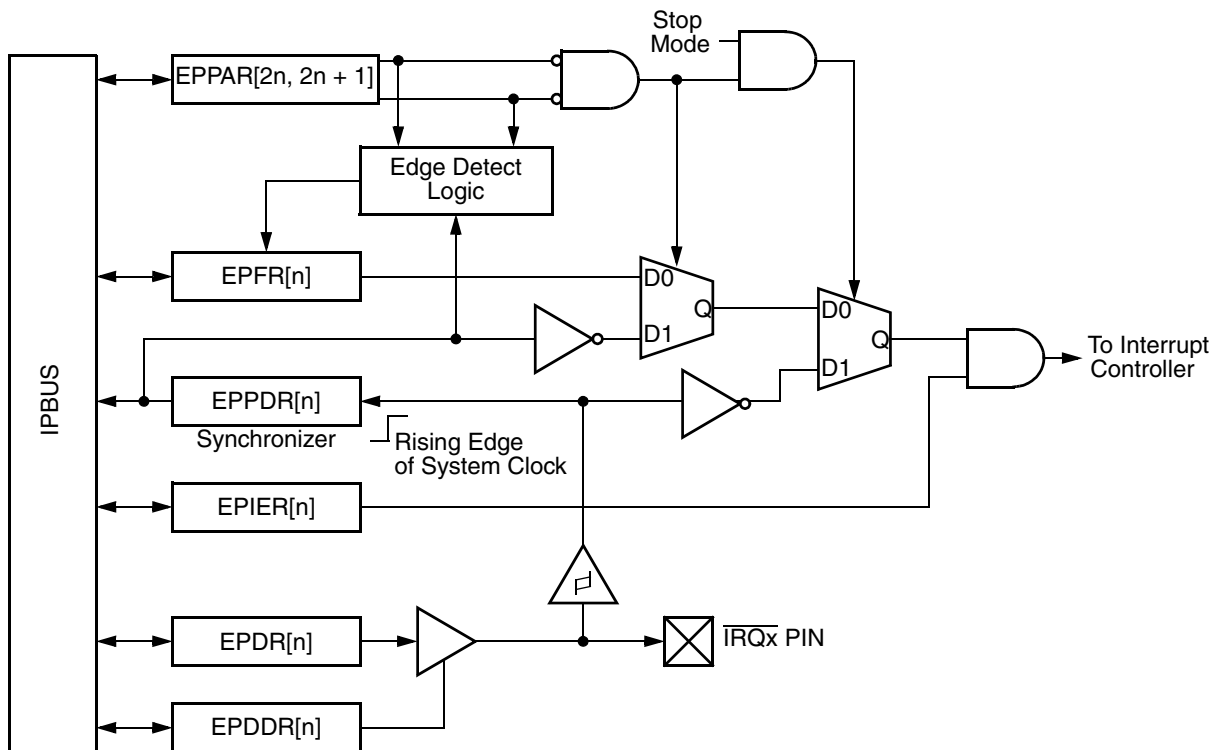
Figure 13-1. EPORT Block Diagram

13.2   Low-Power Mode Operation

This section describes the operation of the EPORT module in low-power modes. For more information on low-power modes, see Chapter 7, "Power Management." Table 13-1 shows EPORT module operation in low-power modes, and describes how this module may exit from each mode.

NOTE

The low-power interrupt control register (LPICR) in the System Control Module specifies the interrupt level at or above which is needed to bring the device out of a low-power mode.

Table 13-1. Edge Port Module Operation in Low-power Modes

Low-power Mode	EPORT Operation	Mode Exit
Wait	Normal	Any $\overline{\text{IRQx}}$ Interrupt at or above level in LPICR
Doze	Normal	Any $\overline{\text{IRQx}}$ Interrupt at or above level in LPICR
Stop	Level-sensing Only	Any $\overline{\text{IRQx}}$ Interrupt set for level-sensing at or above level in LPICR

In wait and doze modes, the EPORT module continues to operate as it does in run mode. It may be configured to exit the low-power modes by generating an interrupt request on either a selected edge or a low level on an external pin. In stop mode, there are no clocks available to perform the edge-detect function. Only the level-detect logic is active (if configured) to allow any low level on the external interrupt pin to generate an interrupt (if enabled) to exit stop mode.

NOTE

The input pin synchronizer is bypassed for the level-detect logic since no clocks are available.

13.3 Interrupt/General-Purpose I/O Pin Descriptions

All pins default to general-purpose input pins at reset. The pin value is synchronized to the rising edge of CLKOUT when read from the EPORT pin data register (EPPDR). The values used in the edge/level detect logic are also synchronized to the rising edge of CLKOUT. These pins use Schmitt triggered input buffers which have built in hysteresis designed to decrease the probability of generating false edge-triggered interrupts for slow rising and falling input signals.

When a pin is configured as an output, it is driven to a state whose level is determined by the corresponding bit in the EPORT data register (EPDR). All bits in the EPDR are high at reset.

13.4   Memory Map and Registers

This subsection describes the memory map and register structure.

13.4.1   Memory Map

Refer to Table 13-2 for a description of the EPORT memory map. The EPORT has an IPSBAR offset for base address of 0x0013_0000.

Table 13-2. Edge Port Module Memory Map

IPSBAR Offset	Bits 15–8	Bits 7–0	Access[1]
0x0013_0000	EPORT Pin Assignment Register (EPPAR)		S
0x0013_0002	EPORT Data Direction Register (EPDDR)	EPORT Interrupt Enable Register (EPIER)	S
0x0013_0004	EPORT Data Register (EPDR)	EPORT Pin Data Register (EPPDR)	S/U
0x0013_0006	EPORT Flag Register (EPFR)	Reserved[2]	S/U

[1] S = CPU supervisor mode access only. S/U = CPU supervisor or user mode access. User mode accesses to supervisor only addresses have no effect and result in a cycle termination transfer error.

[2] Writing to reserved address locations has no effect, and reading returns 0s.

13.4.2   Registers

The EPORT programming model consists of these registers:

- The EPORT pin assignment register (EPPAR) controls the function of each pin individually.
- The EPORT data direction register (EPDDR) controls the direction of each one of the pins individually.
- The EPORT interrupt enable register (EPIER) enables interrupt requests for each pin individually.
- The EPORT data register (EPDR) holds the data to be driven to the pins.
- The EPORT pin data register (EPPDR) reflects the current state of the pins.
- The EPORT flag register (EPFR) individually latches EPORT edge events.

13.4.2.1   EPORT Pin Assignment Register (EPPAR)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	EPPA7		EPPA6		EPPA5		EPPA4		EPPA3		EPPA2		EPPA1		—	
Reset	0000_0000_0000_0000															
R/W	R/W														R	
Address	IPSBAR + 0x0013_0000, 0x0013_0001															

Figure 13-2. EPORT Pin Assignment Register (EPPAR)

Table 13-3. EPPAR Field Descriptions

Bit(s)	Name	Description
15–2	EPPAx	EPORT pin assignment select fields. The read/write EPPAx fields configure EPORT pins for level detection and rising and/or falling edge detection. Pins configured as level-sensitive are inverted so that a logic 0 on the external pin represents a valid interrupt request. Level-sensitive interrupt inputs are not latched. To guarantee that a level-sensitive interrupt request is acknowledged, the interrupt source must keep the signal asserted until acknowledged by software. Level sensitivity must be selected to bring the device out of stop mode with an $\overline{\text{IRQx}}$ interrupt. Pins configured as edge-triggered are latched and need not remain asserted for interrupt generation. A pin configured for edge detection can trigger an interrupt regardless of its configuration as input or output. Interrupt requests generated in the EPORT module can be masked by the interrupt controller module. EPPAR functionality is independent of the selected pin direction. Reset clears the EPPAx fields. 00 Pin $\overline{\text{IRQx}}$ level-sensitive 01 Pin $\overline{\text{IRQx}}$ rising edge triggered 10 Pin $\overline{\text{IRQx}}$ falling edge triggered 11 Pin $\overline{\text{IRQx}}$ both falling edge and rising edge triggered
1–0	—	Reserved, should be cleared.

13.4.2.2 EPORT Data Direction Register (EPDDR)

	7	6	5	4	3	2	1	0
Field	EPDD7	EPDD6	EPDD5	EPDD4	EPDD3	EPDD2	EPDD1	—
Reset	0000_0000							
R/W	R/W							R
Address	IPSBAR + 0x0013_0002							

Figure 13-3. EPORT Data Direction Register (EPDDR)

Table 13-4. EPDD Field Descriptions

Bit(s)	Name	Description
7–1	EPDDx	Setting any bit in the EPDDR configures the corresponding pin as an output. Clearing any bit in EPDDR configures the corresponding pin as an input. Pin direction is independent of the level/edge detection configuration. Reset clears EPDD7-EPDD1. To use an EPORT pin as an external interrupt request source, its corresponding bit in EPDDR must be clear. Software can generate interrupt requests by programming the EPORT data register when the EPDDR selects output. 1 Corresponding EPORT pin configured as output 0 Corresponding EPORT pin configured as input
0	—	Reserved, should be cleared.

13.4.2.3 Edge Port Interrupt Enable Register (EPIER)

	7	6	5	4	3	2	1	0
Field	EPIE7	EPIE6	EPIE5	EPIE4	EPIE3	EPIE2	EPIE1	—
Reset	0000_0000							
R/W	R/W							R
Address	IPSBAR + 0x0013_0003							

Figure 13-4. EPORT Port Interrupt Enable Register (EPIER)

Table 13-5. EPIER Field Descriptions

Bit(s)	Name	Description
7–1	EPIEx	Edge port interrupt enable bits enable EPORT interrupt requests. If a bit in EPIER is set, EPORT generates an interrupt request when: • The corresponding bit in the EPORT flag register (EPFR) is set or later becomes set. • The corresponding pin level is low and the pin is configured for level-sensitive operation. Clearing a bit in EPIER negates any interrupt request from the corresponding EPORT pin. Reset clears EPIE7-EPIE1. 1 Interrupt requests from corresponding EPORT pin enabled 0 Interrupt requests from corresponding EPORT pin disabled
0	—	Reserved, should be cleared.

13.4.2.4 Edge Port Data Register (EPDR)

	7	6	5	4	3	2	1	0
Field	EPD7	EPD6	EPD5	EPD4	EPD3	EPD2	EPD1	—
Reset	1111_1111							
R/W	R/W							R
Address	IPSBAR + 0x0013_0004							

Figure 13-5. EPORT Port Data Register (EPDR)

Table 13-6. EPDR Field Descriptions

Bit(s)	Name	Description
7–1	EPDx	Edge port data bits. Data written to EPDR is stored in an internal register; if any pin of the port is configured as an output, the bit stored for that pin is driven onto the pin. Reading EDPR returns the data stored in the register. Reset sets EPD7-EPD1.
0	—	Reserved, should be cleared.

13.4.2.5 Edge Port Pin Data Register (EPPDR)

	7	6	5	4	3	2	1	0
Field	EPPD7	EPPD6	EPPD5	EPPD4	EPPD3	EPPD2	EPPD1	—
Reset	Current pin state							0
R/W	R							
Address	IPSBAR + 0x0013_0005							

Figure 13-6. EPORT Port Pin Data Register (EPPDR)

Table 13-7. EPPDR Field Descriptions

Bit(s)	Name	Description
7–1	EPPDx	Edge port pin data bits. The read-only EPPDR reflects the current state of the EPORT pins $\overline{IRQ7}$–$\overline{IRQ1}$. Writing to EPPDR has no effect, and the write cycle terminates normally. Reset does not affect EPPDR.
0	—	Reserved, should be cleared.

13.4.2.6 Edge Port Flag Register (EPFR)

	7	6	5	4	3	2	1	0
Field	EPF7	EPF6	EPF5	EPF4	EPF3	EPF2	EPF1	—
Reset	0000_0000							
R/W	R/W							R
Address	IPSBAR + 0x0013_0006							

Figure 13-7. EPORT Port Flag Register (EPFR)

Table 13-8. EPFR Field Descriptions

Bit(s)	Name	Description
7–1	EPFx	Edge port flag bits. When an EPORT pin is configured for edge triggering, its corresponding read/write bit in EPFR indicates that the selected edge has been detected. Reset clears EPF7-EPF1. Bits in this register are set when the selected edge is detected on the corresponding pin. A bit remains set until cleared by writing a 1 to it. Writing 0 has no effect. If a pin is configured as level-sensitive (EPPARx = 00), pin transitions do not affect this register. 1 Selected edge for $\overline{IRQx}$ pin has been detected. 0 Selected edge for $\overline{IRQx}$ pin has not been detected.
0	—	Reserved, should be cleared.

Chapter 14
DMA Controller Module

This chapter describes the Direct Memory Access (DMA) controller module. It provides an overview of the module and describes in detail its signals and registers. The latter sections of this chapter describe operations, features, and supported data transfer modes in detail.

NOTE

The designation "*n*" is used throughout this section to refer to registers or signals associated with one of the four identical DMA channels: DMA0, DMA1, DMA2 or DMA3.

14.1   Overview

The DMA controller module provides an efficient way to move blocks of data with minimal processor interaction. The DMA module, shown in Figure 14-1, provides four channels that allow byte, word, longword, or 16-byte burst data transfers. Each channel has a dedicated source address register (SAR*n*), destination address register (DAR*n*), byte count register (BCR*n*), control register (DCR*n*), and status register (DSR*n*). Transfers are dual address to on-chip devices, such as UARTs, and GPIOs.
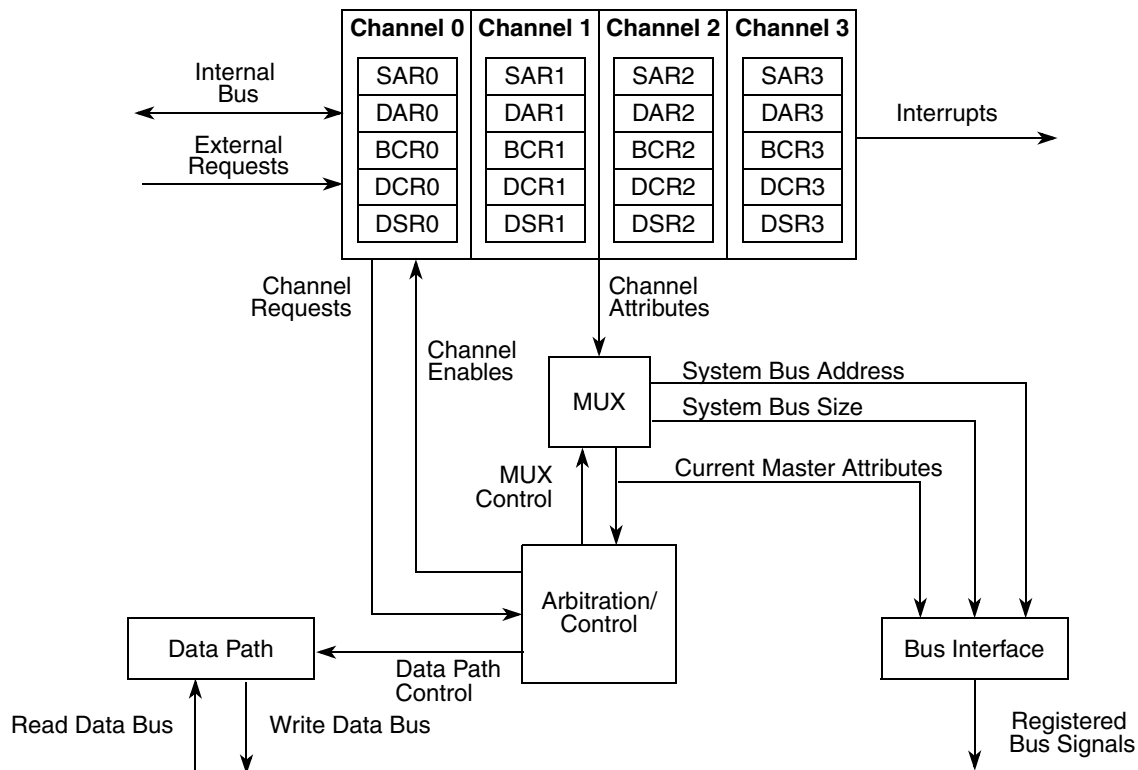
Figure 14-1. DMA Signal Diagram

NOTE

Throughout this chapter "external request" and DREQ are used to refer to a DMA request from one of the on-chip UARTs or DMA timers.

14.1.1 DMA Module Features

The DMA controller module features are as follows:

- Four independently programmable DMA controller channels
- Auto-alignment feature for source or destination accesses
- Dual-address transfers
- Channel arbitration on transfer boundaries
- Data transfers in 8-, 16-, 32-, or 128-bit blocks using a 16-byte buffer
- Continuous-mode or cycle-steal transfers
- Independent transfer widths for source and destination
- Independent source and destination address registers

14.2 DMA Transfer Overview

The DMA module can transfer data faster than the ColdFire core. The term "direct memory access" refers to a fast method of moving data within system memory (including memory and peripheral devices) with minimal processor intervention, greatly improving overall system performance. The DMA module consists of four independent, functionally equivalent channels, so references to DMA in this chapter apply to any of the channels. It is not possible to implicitly address all four channels at once.

The processor generates DMA requests internally by setting DCR[START]; the UART modules and DMA timers can generate a DMA request by asserting internal DREQ signals. The processor can program bus bandwidth for each channel. The channels support cycle-steal and continuous transfer modes; see Section 14.4.1, "Transfer Requests (Cycle-Steal and Continuous Modes)."

The DMA controller supports dual-address transfers. The DMA channels support up to 32 data bits.

- Dual-address transfers—A dual-address transfer consists of a read followed by a write and is initiated by an internal request using the START bit or by asserting DREQ. Two types of transfer can occur: a read from a source device or a write to a destination device. See Figure 14-2 for more information.
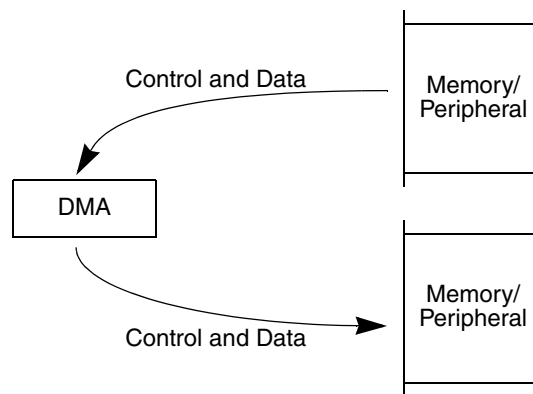
Figure 14-2. Dual-Address Transfer

Any operation involving the DMA module follows the same three steps:

1. Channel initialization—Channel registers are loaded with control information, address pointers, and a byte-transfer count.
2. Data transfer—The DMA accepts requests for operand transfers and provides addressing and bus control for the transfers.
3. Channel termination—Occurs after the operation is finished, either successfully or due to an error. The channel indicates the operation status in the channel's DSR, described in Section 14.3.5, "DMA Status Registers (DSR0–DSR3)."

14.3 DMA Controller Module Programming Model

This section describes each internal register and its bit assignment. Note that modifying DMA control registers during a DMA transfer can result in undefined operation. Table 14-1 shows the mapping of DMA controller registers. Note the differences for the byte count registers depending on the value of MPARK[BCR24BIT]. See Section 8.5.3, "Bus Master Park Register (MPARK)" for further information.

Table 14-1. Memory Map for DMA Controller Module Registers

DMA Channel	IPSBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0	0x100	Source address register 0 (SAR0) [p. 14-4]			
	0x104	Destination address register 0 (DAR0) [p. 14-5]			
	0x108	DMA status register 0 (DSR0) [p. 14-8]	Byte count register 0 (BCR24BIT = 1) (BCR0) [p. 14-5]		
	0x10C	DMA control register 0 (DCR0) [p. 14-5]			
1	0x140	Source address register 1 (SAR1) [p. 14-4]			
	0x144	Destination address register 1 (DAR1) [p. 14-5]			
	0x148	DMA status register 0 (DSR0) [p. 14-8]	Byte count register 1 (BCR24BIT = 1) (BCR0) [p. 14-5]		
	0x14C	DMA control register 1 (DCR0) [p. 14-5]			
2	0x180	Source address register 2 (SAR2) [p. 14-4]			
	0x184	Destination address register 2 (DAR2) [p. 14-5]			
	0x188	DMA status register 0 (DSR0) [p. 14-8]	Byte count register 2 (BCR24BIT = 1) (BCR0) [p. 14-5]		
	0x18C	DMA control register 2 (DCR0) [p. 14-5]			
3	0x1C0	Source address register 3 (SAR3) [p. 14-4]			
	0x1C4	Destination address register 3 (DAR3) [p. 14-5]			
	0x1C8	DMA status register 0 (DSR0) [p. 14-8]	Byte count register 3 (BCR24BIT = 1) (BCR0) [p. 14-5]		
	0x1CC	DMA control register 3 (DCR0) [p. 14-5]			

14.3.1   Source Address Registers (SAR0–SAR3)

SAR*n*, shown in Figure 14-3, contains the address from which the DMA controller requests data.

	31			0
Field		SAR		
Reset		0000_0000_0000_0000_0000_0000_0000_0000		
R/W		R/W		
Address		IPSBAR + 0x100, 0x140, 0x180, 0x1C0		

Figure 14-3. Source Address Registers (SAR*n*)

NOTE

The backdoor enable bit must be set in both the core and SCM in order to enable backdoor accesses from the DMA to SRAM. See Section 8.4.2, "Memory Base Address Register (RAMBAR)" for more details.

14.3.2 Destination Address Registers (DAR0–DAR3)

DAR*n*, shown in Figure 14-4, holds the address to which the DMA controller sends data.

	31	0
Field	DAR	
Reset	0000_0000_0000_0000_0000_0000_0000_0000	
R/W	R/W	
Address	IPSBAR + 0x104, 0x144, 0x184, 0x1C4	

Figure 14-4. Destination Address Registers (DAR*n*)

NOTE

The DMA should not be used to write data to the UART transmit FIFO in cycle steal mode. When the UART interrupt is used as a DMA request it does not negate fast enough to get a single transfer. The UART transmit FIFO only has one entry so the data from the second byte would be lost.

14.3.3 Byte Count Registers (BCR0–BCR3)

BCR*n*, shown in Figure 14-5, holds the number of bytes yet to be transferred for a given block. The offset within the memory map is based on the value of MPARK[BCR24BIT]. BCR*n* decrements on the successful completion of the address transfer of a write transfer. BCR*n* decrements by 1, 2, 4, or 16 for byte, word, longword, or line accesses, respectively.

Figure 14-5 shows BCR*n*.

	31	24	23	0
Field	—		BCR	
Reset	—		0000_0000_0000_0000_0000_0000	
R/W	R/W			
Address	IPSBAR + 0x108, 0x148, 0x188, 0x1C8			

Figure 14-5. Byte Count Registers (BCR*n*)—BCR24BIT = 1

DSR*n*[DONE], shown in Figure 14-7, is set when the block transfer is complete.

When a transfer sequence is initiated and BCR*n*[BCR] is not a multiple of 16, 4, or 2 when the DMA is configured for line, longword, or word transfers, respectively, DSR*n*[CE] is set and no transfer occurs. See Section 14.3.5, "DMA Status Registers (DSR0–DSR3)."

14.3.4 DMA Control Registers (DCR0–DCR3)

DCR*n*, shown in Figure 14-6, is used for configuring the DMA controller module. Note that DCR*n*[AT] is available only if MPARK[BCR24BIT] is set. See Section 8.5.3, "Bus Master Park Register (MPARK)" for more information.
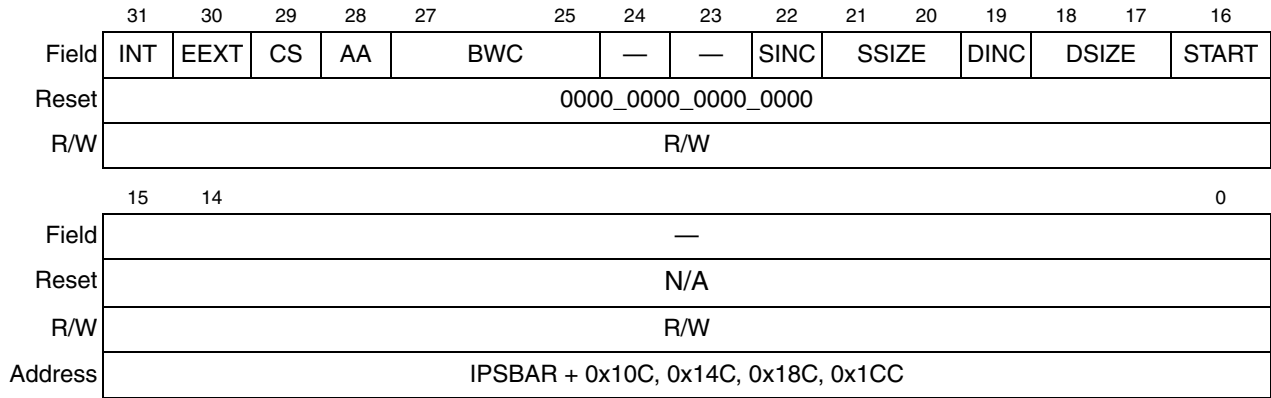
Figure 14-6. DMA Control Registers (DCR*n*)

Table 14-2 describes DCR*n* fields.

Table 14-2. DCR*n* Field Descriptions

Bits	Name	Description
31	INT	Interrupt on completion of transfer. Determines whether an interrupt is generated by completing a transfer or by the occurrence of an error condition. 0 No interrupt is generated. 1 Internal interrupt signal is enabled.
30	EEXT	Enable external request. Care should be taken because a collision can occur between the START bit and DREQ when EEXT = 1. 0 External request is ignored. 1 Enables external request to initiate transfer. The internal request (initiated by setting the START bit) is always enabled.
29	CS	Cycle steal. 0 DMA continuously makes read/write transfers until the BCR decrements to 0. 1 Forces a single read/write transfer per request. The request may be internal by setting the START bit, or external by asserting DREQ.
28	AA	Auto-align. AA and SIZE determine whether the source or destination is auto-aligned, that is, transfers are optimized based on the address and size. See Section 14.4.4.1, "Auto-Alignment." 0 Auto-align disabled 1 If SSIZE indicates a transfer no smaller than DSIZE, source accesses are auto-aligned; otherwise, destination accesses are auto-aligned. Source alignment takes precedence over destination alignment. If auto-alignment is enabled, the appropriate address register increments, regardless of DINC or SINC.

Table 14-2. DCR*n* Field Descriptions (continued)

Bits	Name	Description
27–25	BWC	Bandwidth control. Indicates the number of bytes in a block transfer. When the byte count reaches a multiple of the BWC value, the DMA releases the bus. For example, if BCR24BIT is 0, BWC is 001 (512 bytes or value of 0x0200), and BCR is 0x1000, the bus is relinquished after BCR values of 0x0E00, 0x0C00, 0x0A00, 0x0800, 0x0600, 0x0400, and 0x0200. If BCR24BIT is 0, BWC is 110, and BCR is 33000, the bus is released after 232 bytes because the BCR is at 32768, a multiple of 16384.
24-23	—	Reserved, should be cleared.
22	SINC	Source increment. Controls whether a source address increments after each successful transfer. 0 No change to SAR after a successful transfer. 1 The SAR increments by 1, 2, 4, or 16, as determined by the transfer size.
21–20	SSIZE	Source size. Determines the data size of the source bus cycle for the DMA control module. 00 Longword 01 Byte 10 Word 11 Line (16-byte burst)
19	DINC	Destination increment. Controls whether a destination address increments after each successful transfer. 0 No change to the DAR after a successful transfer. 1 The DAR increments by 1, 2, 4, or 16, depending upon the size of the transfer.
18–17	DSIZE	Destination size. Determines the data size of the destination bus cycle for the DMA controller. 00 Longword 01 Byte 10 Word 11 Line (16-byte burst)
16	START	Start transfer. 0 DMA inactive 1 The DMA begins the transfer in accordance to the values in the control registers. START is cleared automatically after one system clock and is always read as logic 0.
15–0	—	Reserved, should be cleared.

The table embedded in the BWC row:

Encoding	BCR24BIT = 0	BCR24BIT = 1
000	DMA has priority and does not negate its request until transfer completes.	
001	512	16384
010	1024	32768
011	2048	65536
100	4096	131072
101	8192	262144
110	16384	524288
111	32768	1048576

14.3.5 DMA Status Registers (DSR0–DSR3)

In response to an event, the DMA controller writes to the appropriate DSR*n* bit, Figure 14-7. Only a write to DSR*n*[DONE] results in action.

	7	6	5	4	3	2	1	0
Field	—	CE	BES	BED	—	REQ	BSY	DONE
Reset	0000_0000							
R/W	R/W							
Address	IPSBAR + 0x108, 0x148, 0x18, 0x1C8							

Figure 14-7. DMA Status Registers (DSR*n*)

Table 14-3 describes DSR*n* fields.

Table 14-3. DSR*n* Field Descriptions

Bits	Name	Description
7	—	Reserved, should be cleared.
6	CE	Configuration error. Occurs when BCR, SAR, or DAR does not match the requested transfer size, or if BCR = 0 when the DMA receives a start condition. CE is cleared at hardware reset or by writing a 1 to DSR[DONE]. 0　No configuration error exists. 1　A configuration error has occurred.
5	BES	Bus error on source 0　No bus error occurred. 1　The DMA channel terminated with a bus error during the read portion of a transfer.
4	BED	Bus error on destination 0　No bus error occurred. 1　The DMA channel terminated with a bus error during the write portion of a transfer.
3	—	Reserved, should be cleared.
2	REQ	Request 0　No request is pending or the channel is currently active. Cleared when the channel is selected. 1　The DMA channel has a transfer remaining and the channel is not selected.
1	BSY	Busy 0　DMA channel is inactive. Cleared when the DMA has finished the last transaction. 1　BSY is set the first time the channel is enabled after a transfer is initiated.
0	DONE	Transactions done. Set when all DMA controller transactions complete, as determined by transfer count or error conditions. When BCR reaches zero, DONE is set when the final transfer completes successfully. DONE can also be used to abort a transfer by resetting the status bits. When a transfer completes, software must clear DONE before reprogramming the DMA. 0　Writing or reading a 0 has no effect. 1　DMA transfer completed. Writing a 1 to this bit clears all DMA status bits and can be used in an interrupt handler to clear the DMA interrupt and error bits.

14.4 DMA Controller Module Functional Description

In the following discussion, the term "DMA request" implies that DCR*n*[START] or DCR*n*[EEXT] is set, followed by assertion of DREQ*n*. The START bit is cleared when the channel begins an internal access.

Before initiating a dual-address access, the DMA module verifies that DCRn[SSIZE,DSIZE] are consistent with the source and destination addresses. If they are not consistent, the configuration error bit, DSRn[CE], is set. If misalignment is detected, no transfer occurs, DSRn[CE] is set, and, depending on the DCR configuration, an interrupt event is issued. Note that if the auto-align bit, DCRn[AA], is set, error checking is performed on the appropriate registers.

A read/write transfer reads bytes from the source address and writes them to the destination address. The number of bytes is the larger of the sizes specified by DCRn[SSIZE] and DCRn[DSIZE]. See Section 14.3.4, "DMA Control Registers (DCR0–DCR3)."

Source and destination address registers (SARn and DARn) can be programmed in the DCRn to increment at the completion of a successful transfer. BCRn decrements when an address transfer write completes for a single-address access (DCRn[SAA] = 0) or when SAA = 1.

14.4.1 Transfer Requests (Cycle-Steal and Continuous Modes)

The DMA channel supports internal and external requests. A request is issued by setting DCRn[START] or by asserting DREQn. Setting DCRn[EEXT] enables recognition of external DMA requests. Selecting between cycle-steal and continuous modes minimizes bus usage for either internal or external requests.

- Cycle-steal mode (DCRn[CS] = 1)—Only one complete transfer from source to destination occurs for each request. If DCRn[EEXT] is set, a request can be either internal or external. An internal request is selected by setting DCRn[START]. An external request is initiated by asserting DREQn while DCRn[EEXT] is set. Note that multiple transfers will occur if DREQn is continuously asserted.

- Continuous mode (DCRn[CS] = 0)—After an internal or external request, the DMA continuously transfers data until BCRn reaches zero or a multiple of DCRn[BWC] or until DSRn[DONE] is set. If BCRn is a multiple of BWC, the DMA request signal is negated until the bus cycle terminates to allow the internal arbiter to switch masters. DCRn[BWC] = 000 specifies the maximum transfer rate; other values specify a transfer rate limit.

  The DMA performs the specified number of transfers, then relinquishes bus control. The DMA negates its internal bus request on the last transfer before BCRn reaches a multiple of the boundary specified in BWC. On completion, the DMA reasserts its bus request to regain mastership at the earliest opportunity. The DMA loses bus control for a minimum of one bus cycle.

14.4.2 Data Transfer Modes

Each channel supports dual-address transfers, described in the next section.

14.4.2.1 Dual-Address Transfers

Dual-address transfers consist of a source data read and a destination data write. The DMA controller module begins a dual-address transfer sequence during a DMA request. If no error condition exists, DSRn[REQ] is set.

- Dual-address read—The DMA controller drives the SARn value onto the internal address bus. If DCRn[SINC] is set, the SARn increments by the appropriate number of bytes upon a successful read cycle. When the appropriate number of read cycles complete (multiple reads if the destination size is larger than the source), the DMA initiates the write portion of the transfer.

  If a termination error occurs, DSRn[BES,DONE] are set and DMA transactions stop.

- Dual-address write—The DMA controller drives the DARn value onto the address bus. If DCRn[DINC] is set, DARn increments by the appropriate number of bytes at the completion of a

successful write cycle. BCRn decrements by the appropriate number of bytes. DSRn[DONE] is set when BCRn reaches zero. If the BCRn is greater than zero, another read/write transfer is initiated. If the BCRn is a multiple of DCRn[BWC], the DMA request signal is negated until termination of the bus cycle to allow the internal arbiter to switch masters.

If a termination error occurs, DSRn[BES,DONE] are set and DMA transactions stop.

14.4.3 Channel Initialization and Startup

Before a block transfer starts, channel registers must be initialized with information describing configuration, request-generation method, and the data block.

14.4.3.1 Channel Prioritization

The four DMA channels are prioritized in ascending order (channel 0 having highest priority and channel 3 having the lowest) or in an order determined by DCRn[BWC]. If the BWC encoding for a DMA channel is 000, that channel has priority only over the channel immediately preceding it. For example, if DCR3[BWC] = 000, DMA channel 3 has priority over DMA channel 2 (assuming DCR2[BWC] ≠ 000) but not over DMA channel 1.

If DCR0[BWC] = DCR1[BWC] = 000, DMA0 still has priority over DMA1. In this case, DCR1[BWC] = 000 does not affect prioritization.

Simultaneous external requests are prioritized either in ascending order or in an order determined by each channel's DCRn[BWC] bits.

14.4.3.2 Programming the DMA Controller Module

Note the following general guidelines for programming the DMA:

- No mechanism exists within the DMA module itself to prevent writes to control registers during DMA accesses.
- If the DCRn[BWC] value of sequential channels are equal, the channels are prioritized in ascending order.

The SARn is loaded with the source (read) address. If the transfer is from a peripheral device to memory, the source address is the location of the peripheral data register. If the transfer is from memory to either a peripheral device or memory, the source address is the starting address of the data block. This can be any aligned byte address.

The DARn should contain the destination (write) address. If the transfer is from a peripheral device to memory, or from memory to memory, the DARn is loaded with the starting address of the data block to be written. If the transfer is from memory to a peripheral device, DARn is loaded with the address of the peripheral data register. This address can be any aligned byte address.

SARn and DARn change after each cycle depending on DCRn[SSIZE,DSIZE, SINC,DINC] and on the starting address. Increment values can be 1, 2, 4, or 16 for byte, word, longword, or 16-byte line transfers, respectively. If the address register is programmed to remain unchanged (no count), the register is not incremented after the data transfer.

BCRn[BCR] must be loaded with the number of byte transfers to occur. It is decremented by 1, 2, 4, or 16 at the end of each transfer, depending on the transfer size. DSRn[DONE] must be cleared for channel startup.

As soon as the channel has been initialized, it is started by writing a one to DCRn[START] or asserting DREQn, depending on the status of DCRn[EEXT]. Programming the channel for internal requests causes

the channel to request the bus and start transferring data immediately. If the channel is programmed for external request, DREQ*n* must be asserted before the channel requests the bus.

Changes to DCR*n* are effective immediately while the channel is active. To avoid problems with changing a DMA channel setup, write a one to DSR*n*[DONE] to stop the DMA channel.

14.4.4 Data Transfer

This section describes auto-alignment and bandwidth control for DMA transfers.

14.4.4.1 Auto-Alignment

Auto-alignment allows block transfers to occur at the optimal size based on the address, byte count, and programmed size. To use this feature, DCR*n*[AA] must be set. The source is auto-aligned if DCR*n*[SSIZE] indicates a transfer size larger than DCR*n*[DSIZE]. Source alignment takes precedence over the destination when the source and destination sizes are equal. Otherwise, the destination is auto-aligned. The address register chosen for alignment increments regardless of the increment value. Configuration error checking is performed on registers not chosen for alignment.

If BCR*n* is greater than 16, the address determines transfer size. Bytes, words, or longwords are transferred until the address is aligned to the programmed size boundary, at which time accesses begin using the programmed size.

If BCR*n* is less than 16 at the start of a transfer, the number of bytes remaining dictates transfer size. For example, AA = 1, SAR*n* = 0x0001, BCR*n* = 0x00F0, SSIZE = 00 (longword), and DSIZE = 01 (byte). Because SSIZE > DSIZE, the source is auto-aligned. Error checking is performed on destination registers. The access sequence is as follows:

1. Read byte from 0x0001—write 1 byte, increment SAR*n*.
2. Read word from 0x0002—write 2 bytes, increment SAR*n*.
3. Read longword from 0x0004—write 4 bytes, increment SAR*n*.
4. Repeat longwords until SAR*n* = 0x00F0.
5. Read byte from 0x00F0—write byte, increment SAR*n*.

If DSIZE is another size, data writes are optimized to write the largest size allowed based on the address, but not exceeding the configured size.

14.4.4.2 Bandwidth Control

Bandwidth control makes it possible to force the DMA off the bus to allow access to another device. DCR*n*[BWC] provides seven levels of block transfer sizes. If the BCR*n* decrements to a multiple of the decode of the BWC, the DMA bus request negates until the bus cycle terminates. If a request is pending, the arbiter may then pass bus mastership to another device. If auto-alignment is enabled, DCR*n*[AA] = 1, the BCR*n* may skip over the programmed boundary, in which case, the DMA bus request is not negated.

If BWC = 000, the request signal remains asserted until BCR*n* reaches zero. DMA has priority over the core. Note that in this scheme, the arbiter can always force the DMA to relinquish the bus. See

14.4.5 Termination

An unsuccessful transfer can terminate for one of the following reasons:

- Error conditions—When the MCF5213 encounters a read or write cycle that terminates with an error condition, DSRn[BES] is set for a read and DSRn[BED] is set for a write before the transfer is halted. If the error occurred in a write cycle, data in the internal holding register is lost.
- Interrupts—If DCRn[INT] is set, the DMA drives the appropriate internal interrupt signal. The processor can read DSRn to determine whether the transfer terminated successfully or with an error. DSRn[DONE] is then written with a one to clear the interrupt and the DONE and error bits.

Chapter 15
ColdFire Flash Module (CFM)

The microcontroller incorporates SuperFlash® technology licensed from SST. The ColdFire Flash Module (CFM) is constructed with four banks of 32K x 16-bit Flash to generate a 256-Kbyte, 32-bit wide electrically erasable and programmable read-only memory array. The CFM is ideal for program and data storage for single-chip applications and allows for field reprogramming without external high-voltage sources.

The voltage required to program and erase the Flash is generated internally by on-chip charge pumps. Program and erase operations are performed under CPU control through a command-driven interface to an internal state machine. All Flash physical blocks can be programmed or erased at the same time; however, it is not possible to read from a Flash physical block while the same block is being programmed or erased. The partitioning of array blocks makes it possible to program or erase one pair of Flash physical blocks under the control of software routines executing out of another pair.

NOTE
Some implementations of this microcontroller include only 128 Kbytes of Flash; half that of the full feature set device.

15.1   Features

Features of the CFM include:

- 256-Kbytes of Flash memory on the full featured device, 128-Kbytes of Flash memory on subset parts.
- Basic Flash access time of 2 clock cycles. Optimized processor Flash interface reduces basic Flash access time through interleaving and speculative reads.
- Automated program and erase operation
- Concurrent verify, program, and erase of all array blocks
- Read-while-write capability
- Optional interrupt on command completion
- Flexible scheme for protection against accidental program or erase operations
- Access restriction controls for both supervisor/user and data/program space operations
- Security for single-chip applications
- Single power supply (system V_{DD}) used for all module operations
- Auto-sense amplifier timeout for low-power, low-frequency read operations

NOTE
Enabling Flash security will disable BDM communications.

NOTE
When Flash security is enabled, the chip will boot in single-chip mode regardless of the external reset configuration.

15.2   Block Diagram

The CFM module shown in Figure 15-1 contains the Flash physical blocks, the ColdFire Flash bus and IP bus interfaces, Flash interface, register blocks, and the BIST engine.

Each 128-Kbyte Flash physical block is arranged as two 32,768-word (16 bits) memory arrays. Each of these memory arrays is designated as xH or xL, where x represents one of the four Flash physical blocks (0–3) and H/L represents the high or low 16 bits of each longword of logical memory. Each of these words may be read as either individual bytes or aligned words. Aligned longword access is provided by concatenating the outputs of the each of the two memory arrays within the Flash physical block. Simple reads of bytes, aligned words, and aligned longwords require two 66-MHz clock cycles, although the processor's Flash interface includes logic that reduces the effective access time through two-way longword interleaving and speculative reads.

Flash physical blocks are interleaved on longword (4-byte) boundaries. Therefore, all Flash program, erase, and verify commands operate on adjacent Flash physical blocks and are initiated with a single aligned 32-bit write to the appropriate array location. Any other write operation will cause a cycle termination transfer error. Page erase operates simultaneously on two interleaving erase pages in adjacent Flash physical blocks. Each Flash physical block is organized as 1024 rows of 128 bytes with a single erase page consisting of 8 rows (1024 bytes). Since page erase operates simultaneously on two interleaving and adjacent physical Flash blocks, each erase row is comprised of four 16-bit entries in each of two memory arrays within each of two Flash physical blocks. The first row of Flash is made up of 0H_0L_1H_1L [0] through 0H_0L_1H_1L [31], where each [n] represents four 16-bit words from each memory array in each of two physical blocks, for a total of 256 bytes. Since a single erase page consists of 8 rows of 256 bytes, or 2048 bytes, the first erase page is physically located at 0H_0L_1H_1L [0] through 0H_0L_1H_1L [255]. Mass erase operates simultaneously on two adjacent Flash physical blocks in their entirety and erases a total of 256 Kbytes of Flash space.

An erased Flash bit reads 1 and a programmed Flash bit reads 0. The CFM features a sense amplifier timeout (SATO) block that automatically reduces current consumption during reads at low system clock frequencies.
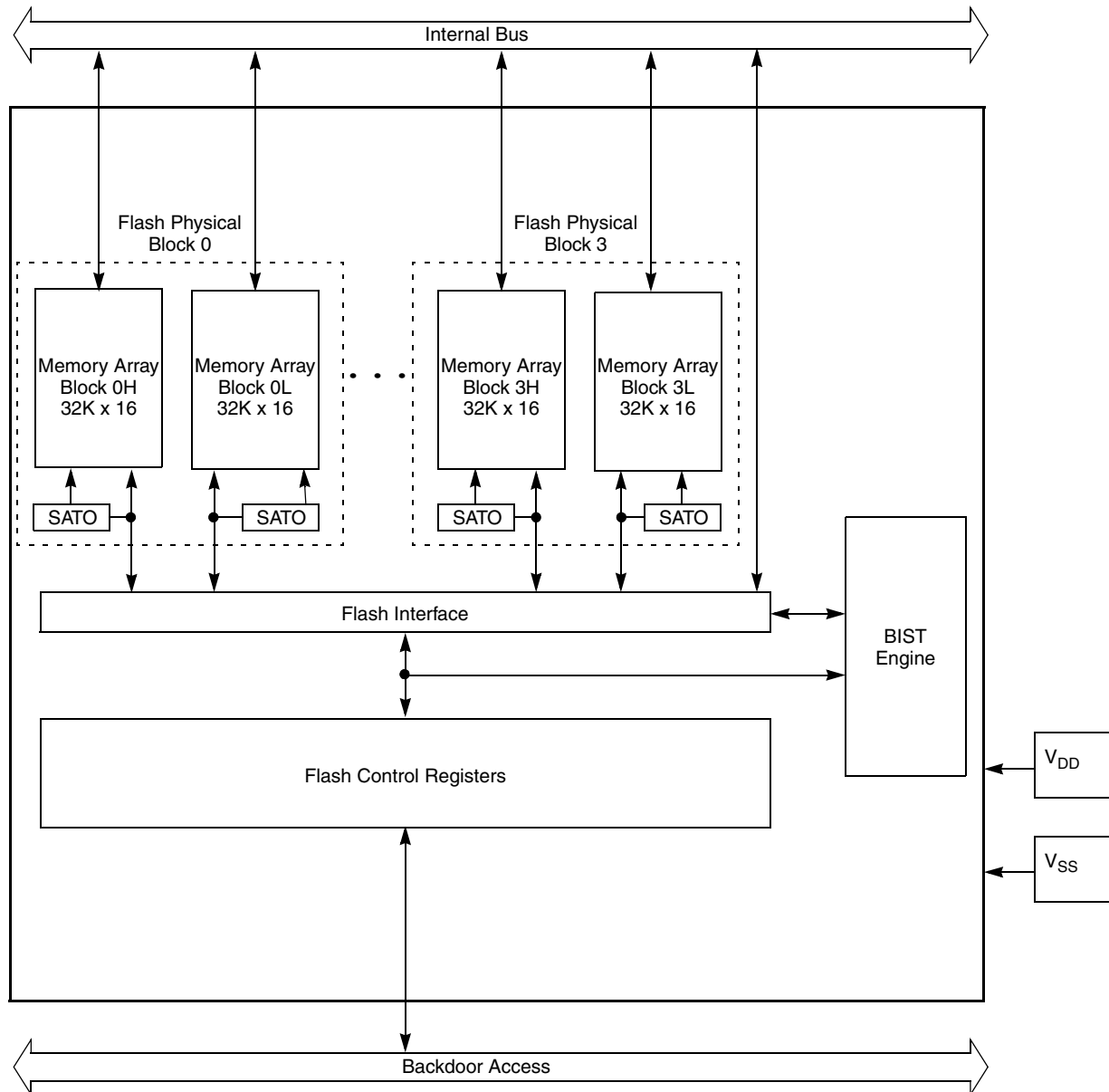
Note:

Mass Erase Block 0 (256 Kbytes) = Flash Physical Block 0 and Flash Physical Block 1.
Mass Erase Block 1 (256 Kbytes) = Flash Physical Block 2 and Flash Physical Block 3.

Figure 15-1. CFM Block Diagram

15.3 Memory Map

Figure 15-2 shows the memory map for the CFM array. The CFM array can reside anywhere in the memory space of the MCU. The starting address of the array is determined by the CFM array base address which must reside on a natural size boundary; that is, the CFM array base address must be an integer multiple of the array size. The CFM register space must reside on a 64 byte boundary as determined by the CFM register base address. Figure 15-2 shows how multiple 32,768 by 16-bit Flash physical blocks interleave to form a contiguous non-volatile memory space. Each pair of 32-bit blocks (even and odd) interleave every 4 bytes to form a 256-Kbyte section of memory.

NOTE

The CFM on the smaller arrays is constructed with two banks of 32K x 16-bit Flash arrays to generate 128 Kbytes of 32-bit Flash memory.
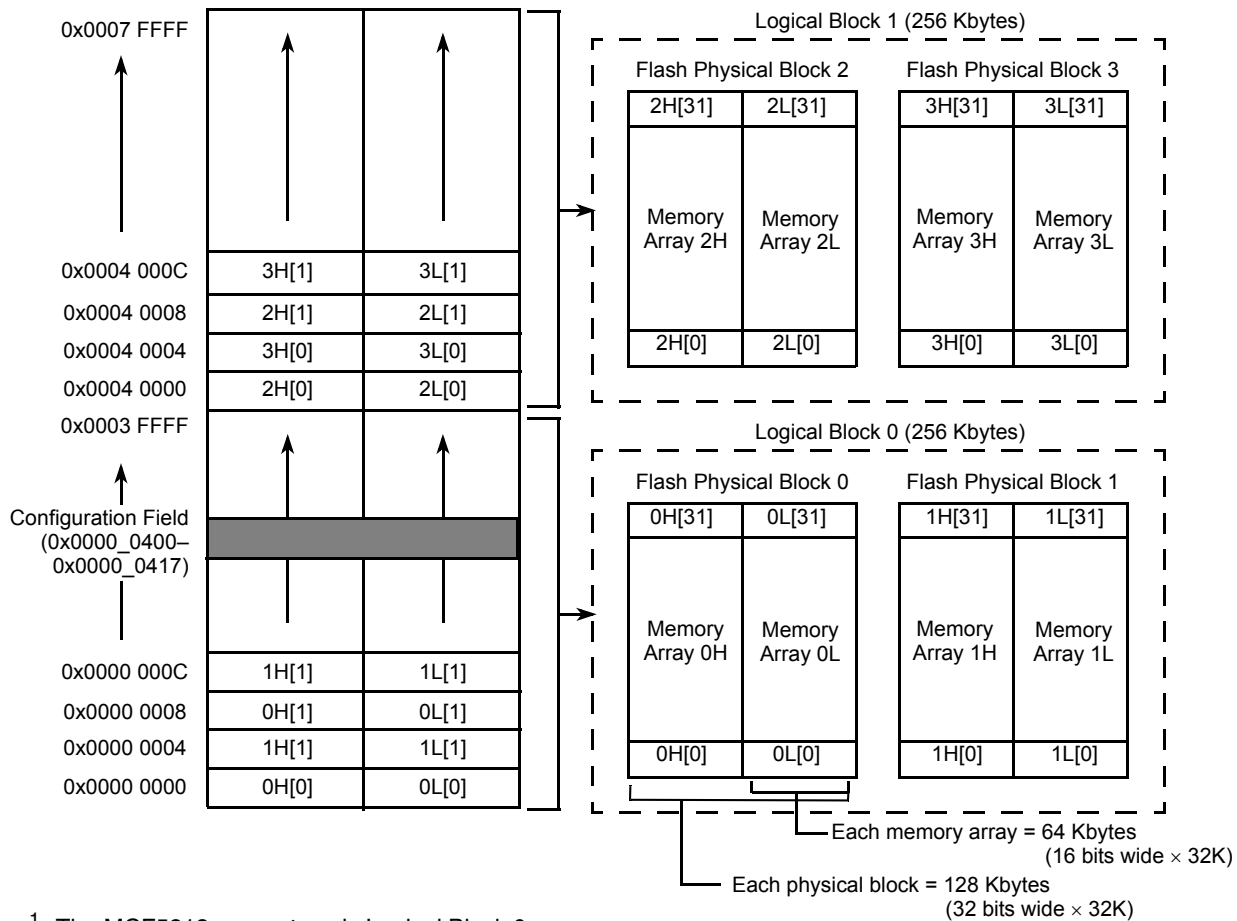
[1] The MCF5213 supports only Logical Block 0.

Figure 15-2. CFM Array Memory Map

The CFM module has hardware interlocks to protect data from accidental corruption. The CFM memory array is logically divided into 16-Kbyte sectors for the purpose of data protection and access control. A flexible scheme allows the protection of any combination of logical sectors (see Section 15.3.4.4, "CFM Protection Register (CFMPROT)"). A similar mechanism is available to control supervisor/user and program/data space access to these sectors.

15.3.1 CFM Configuration Field

The CFM configuration field comprises 24 bytes of reserved array memory space that determines the module protection and access restrictions out of reset. Data to secure the Flash from unauthorized access is also stored in the CFM configuration field. Table 15-1 describes each byte used in this field.

Table 15-1. CFM Configuration Field

Address Offset (from array base address)	Size in Bytes	Description
0x0000_0400–0x0000_0407	8	Back door comparison key
0x0000_0408–0x0000_040B	4	Flash program/erase sector protection Blocks 0H/0L (see Section 15.3.4.4, "CFM Protection Register (CFMPROT)")
0x0000_040C–0x0000_040F	4	Flash supervisor/user space restrictions Blocks 0H/0L (see Section 15.3.4.5, "CFM Supervisor Access Register (CFMSACC)")
0x0000_0410–0x0000_0413	4	Flash program/data space restrictions Blocks 0H/0L (see Section 15.3.4.6, "CFM Data Access Register (CFMDACC)")
0x0000_0414–0x0000_0417	4	Flash security longword (see Section 15.3.4.3, "CFM Security Register (CFMSEC)")

15.3.2 Flash Base Address Register (FLASHBAR)

The configuration information in the Flash base address register (FLASHBAR) controls the operation of the Flash module.

- The FLASHBAR holds the base address of the Flash. The MOVEC instruction provides write-only access to this register.
- The FLASHBAR can be read or written from the debug module in a similar manner.
- All undefined bits in the register are reserved. These bits are ignored during writes to the FLASHBAR, and return zeroes when read from the debug module.
- The back door enable bit, FLASHBAR[BDE], is cleared at reset, disabling back door access to the Flash.
- The FLASHBAR valid bit is programmed according to the chip mode selected at reset (see Chapter 27, "Chip Configuration Module (CCM)" for more details). All other bits are unaffected.

The FLASHBAR register contains several control fields. These fields are shown in Figure 15-3

NOTE

The default value of the FLASHBAR is determined by the chip configuration selected at reset (see Chapter 27, "Chip Configuration Module (CCM)" for more information). If external boot mode is used, then the FLASHBAR located in the processor's CPU space will be invalid and it must be initialized with the valid bit set before the CPU (or modules) can access the on-chip Flash.

NOTE

Flash accesses (reads/writes) by a bus master other than the core, (DMA controller), or writes to Flash by the core during programming must use the backdoor Flash address of IPSBAR plus an offset of 0x0400_0000. For example, for a DMA transfer from the first location of Flash when IPSBAR is still at its default location of 0x4000_0000, the source register would be loaded with 0x4400_0000. Backdoor access to Flash for reads can be made by the bus master, but it takes 2 cycles longer than a direct read of the Flash if using its FLASHBAR address.

NOTE

The Flash is marked as valid on reset based on the RCON (reset configuration) pin state. Flash space is valid on reset when booting in single chip mode (RCON pin asserted and D[26]/D[17]/D[16] set to 110), or when booting internally in master mode (RCON asserted and D[26]/D[17]/D[16] are set to 111 and D[18] and D[19] are set to 00). See Chapter 27, "Chip Configuration Module (CCM)" for more details. When the default reset configuration is not overriden, the MCF5213 will (by default) boot up in single chip mode and the Flash space will be marked as valid at address 0x0. The Flash configuration field is checked during the reset sequence to see if the Flash is secured. If it is the part will always boot from internal Flash, since it will be marked as valid, regardless of what is done for chip configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18		16
Field BA31	BA30	BA29	BA28	BA27	BA26	BA25	BA24	BA23	BA22	BA21	BA20	BA19		—	
Reset						0000_0000_0000_0000									
R/W						R/W									

15			9	8	7	6	5	4	3	2	1	0
Field	—			WP	—		C/I	SC	SD	UC	UD	V
Reset		0000_0001_0010_000										See Note
R/W R/W		R		W	R				R/W			
Address				CPU + 0xC04								

Note: The reset value for the valid bit is determined by the chip mode selected at reset (see Chapter 27, "Chip Configuration Module (CCM)").

Figure 15-3. Flash Base Address Register (FLASHBAR)

Table 15-2. FLASHBAR Field Descriptions

Bits	Name	Description
31–19	BA[31:18]	Base address field. Defines the 0-modulo-512K base address of the Flash module. By programming this field, the Flash may be located on any 512Kbyte boundary within the processor's four gigabyte address space.
18–9	—	Reserved, should be cleared.
8	WP	Write protect. Write only. Allows only read accesses to the Flash. When this bit is set, any attempted write access will generate an access error exception to the ColdFire processor core. 0 Allows read and write accesses to the Flash module 1 Allows only read accesses to the Flash module
7–6	—	Reserved, should be cleared.
5–1	C/I, SC, SD, UC, UD	Address space masks (ASn). These five bit fields allow certain types of accesses to be "masked," or inhibited from accessing the Flash module. The address space mask bits are: C/I CPU space/interrupt acknowledge cycle mask SC Supervisor code address space mask SD Supervisor data address space mask UC User code address space mask UD User data address space mask For each address space bit: 0 An access to the Flash module can occur for this address space 1 Disable this address space from the Flash module. If a reference using this address space is made, it is inhibited from accessing the Flash module, and is processed like any other non-Flash reference. These bits are useful for power management as detailed in Chapter 7, "Power Management."
0	V	Valid. When set, this bit enables the Flash module; otherwise, the module is disabled. 0 Contents of FLASHBAR are not valid 1 Contents of FLASHBAR are valid

15.3.3   CFM Registers

The CFM module also contains a set of control and status registers. The memory map for these registers and their accessibility in supervisor and user modes is shown in Table 15-3.

Table 15-3. CFM Register Address Map

IPSBAR Offset	Bits 31–24	Bits 23–16	Bits 15–8	Bits 7–0	Access[1]
0x1D_0000	CFMMCR		CFMCLKD	Reserved[2]	S
0x1D_0004	Reserved[2]				S
0x1D_0008	CFMSEC				S
0x1D_000C	Reserved[2]				S
0x1D_0010	CFMPROT				S

Table 15-3. CFM Register Address Map

IPSBAR Offset	Bits 31–24	Bits 23–16	Bits 15–8	Bits 7–0	Access[1]
0x1D_0014	CFMSACC				S
0x1D_0018	CFMDACC				S
0x1D_001C	Reserved[2]				S
0x1D_0020	CFMUSTAT	Reserved[2]			S
0x1D_0024	CFMCMD	Reserved[2]			S

[1] S = Supervisor access only. User mode accesses to supervisor only addresses have no effect and result in a cycle termination transfer error.

[2] Addresses not assigned to a register and undefined register bits are reserved for expansion. Write accesses to these reserved address spaces and reserved register bits have no effect.

15.3.4 Register Descriptions

The Flash registers are described in this subsection.

15.3.4.1 CFM Configuration Register (CFMCR)

The CFMCR is used to configure and control the operation of the CFM array.

	15		11	10	9	8	7	6	5	4		0
Field	—			LOCK	PVIE	AEIE	CBEIE	CCIE	KEYACC	—		
Reset	0000_0000_0000_0000											
R/W	R/W											
Address	IPSBAR + 0x1D_0000											

Figure 15-4. CFM Module Configuration Register (CFMCR)

Bits 10 -5 in the CFMCR register are readable and writable with restrictions.

Table 15-4. CFMCR Field Descriptions

Bits	Name	Description
15–11	—	Reserved, should be cleared.
10	LOCK	Write lock control. The LOCK bit is always readable and is set once. 1 CFMPROT, CMFSACC, and CFMDACC register are write-locked. 0 CFMPROT, CMFSACC, and CFMDACC register are writable.
9	PVIE	Protection violation interrupt enable. The PVIE bit is readable and writable. The PVIE bit enables an interrupt in case the protection violation flag, PVIOL, is set. 1 An interrupt will be requested whenever the PVIOL flag is set. 0 PVIOL interrupts disabled.
8	AEIE	Access error interrupt enable. The AEIE bit is readable and writable. The AEIE bit enables an interrupt in case the access error flag, ACCERR, is set. 1 An interrupt will be requested whenever the ACCERR flag is set. 0 ACCERR interrupts disabled.

Table 15-4. CFMCR Field Descriptions

Bits	Name	Description
7	CBEIE	Command buffer empty interrupt enable. The CBEIE bit is readable and writable. CBEIE enables an interrupt request when the command buffer for the Flash physical blocks is empty. 1 Request an interrupt whenever the CBEIF flag is set. 0 Command buffer empty interrupts disabled
6	CCIE	Command complete interrupt enable. The CCIE bit is readable and writable. CCIE enables an interrupt when the command executing for the Flash is complete. 1 Request an interrupt whenever the CCIF flag is set. 0 Command complete interrupts disabled
5	KEYACC	Enable security key writing. The KEYACC bit is readable and only writable if the KEYEN bit in the CFMSEC register is set. 1 Writes to the Flash array are interpreted as keys to open the back door. 0 Writes to the Flash array are interpreted as the start of a program, erase, or verify sequence.
4–0	—	Reserved, should be cleared.

15.3.4.2 CFM Clock Divider Register (CFMCLKD)

The CFMCLKD is used to set the frequency of the clock used for timed events in program and erase algorithms.

	7	6	5				0
Field	DIVLD	PRDIV8			DIV		
Reset				0000_0000			
R/W	R				R/W		
Address				IPSBAR + 0x1D_0002			

Figure 15-5. CFM Clock Divider Register (CFMCLKD)

All bits in CFMCLKD are readable. Bit 7 is a read-only status bit, while bits 6–0 can only be written once.

Table 15-5. CFMCLKD Field Descriptions

Bits	Name	Description
7	DIVLD	Clock divider loaded 1 CFMCLKD has been written since the last reset. 0 CFMCLKD has not been written.
6	PRDIV8	Enable prescaler divide by 8 1 Enables a prescaler that divides the CFM clock by 8 before it enters the CFMCLKD divider. 0 The CFM clock is fed directly into the CFMCLKD divider.
5–0	DIV	Clock divider field. The combination of PRDIV8 and DIV[5:0] effectively divides the CFM input clock down to a frequency between 150 kHz and 200 kHz. The frequency range of the CFM clock is 150 kHz to 102.4 MHz.

NOTE

CFMCLKD must be written with an appropriate value before programming or erasing the Flash array. Refer to Section 15.4.3.1, "Setting the CFMCLKD Register."

15.3.4.3   CFM Security Register (CFMSEC)

The CFMSEC controls the Flash security features.

NOTE

Enabling Flash security will disable BDM communications.

NOTE

When Flash security is enabled, the chip will boot in single-chip mode regardless of the external reset configuration.

	31	30	29			16
Field	KEYEN	SECSTAT		—		
Reset			See Note			
R/W			R			

	15			0
Field		SEC		
Reset		See Note		
R/W		R		
Address		IPSBAR + 0x1D_0008		

Note: The SECSTAT bit reset value is determined by the security state of the Flash. All other bits in the register are loaded at reset from the Flash Security longword stored at the array base address + 0x0000_0414.

Figure 15-6. CFM Security Register (CFMSEC)

Table 15-6. CFMSEC Field Descriptions

Bits	Name	Description
31	KEYEN	Enable back door key to security 1 Back door to Flash is enabled. 0 Back door to Flash is disabled.
30	SECSTAT	Flash security status 1 Flash security is enabled 0 Flash security is disabled

Table 15-6. CFMSEC Field Descriptions

Bits	Name	Description
29–16	—	Reserved. Should be cleared.
15–0	SEC[15:0]	Security field. The SEC bits define the security state of the device; see below.

SEC[15:0]	Description
0x4AC8	Flash secured[1]
All other combinations	Flash unsecured

[1] The 0x4AC8 value was chosen because it represents the ColdFire Halt instruction, making it unlikely that compiled code accidentally programmed at the security longword in the Flash configuration field location would unintentionally secure the device.

The security features of the CFM are described in Section 15.5, "Flash Security Operation."

15.3.4.4 CFM Protection Register (CFMPROT)

The CFMPROT specifies which Flash logical sectors are protected from program and erase operations.

	31		16
Field		PROT	
Reset		See Note	
R/W		R/W	

	15		0
Field		PROT	
Reset		See Note	
R/W		R/W	
Address		IPSBAR + 0x1D_0010	

Note: The CFMPROT register is loaded at reset from the Flash Program/Erase Sector Protection longword stored at the array base address + 0x0000_0400.

Figure 15-7. CFM Protection Register (CFMPROT)

The CFMPROT register is always readable and only writeable when LOCK = 0. To change which logical sectors are protected on a temporary basis, write CFMPROT with a new value after the LOCK bit in CFMCR has been cleared. To change the value of CFMPROT that will be loaded on reset, the protection byte in the Flash configuration field must first be temporarily unprotected using the method just described before reprogramming the protection bytes. Then the Flash Protection longword at offset 0x1D_0400 must be written with the desired value.

Table 15-7. CFMPROT Field Descriptions

Bits	Name	Description
31–0	PROT[31:0]	Sector protection. Each Flash logical sector can be protected from program and erase operations by setting its corresponding PROT bit. 1 Logical sector is protected. 0 Logical sector is not protected.

The CFMPROT controls the protection of thirty-two 16-Kbyte Flash logical sectors in the 512-Kbyte Flash array. Figure 15-8 shows the association between each bit in the CFMPROT and its corresponding logical sector.
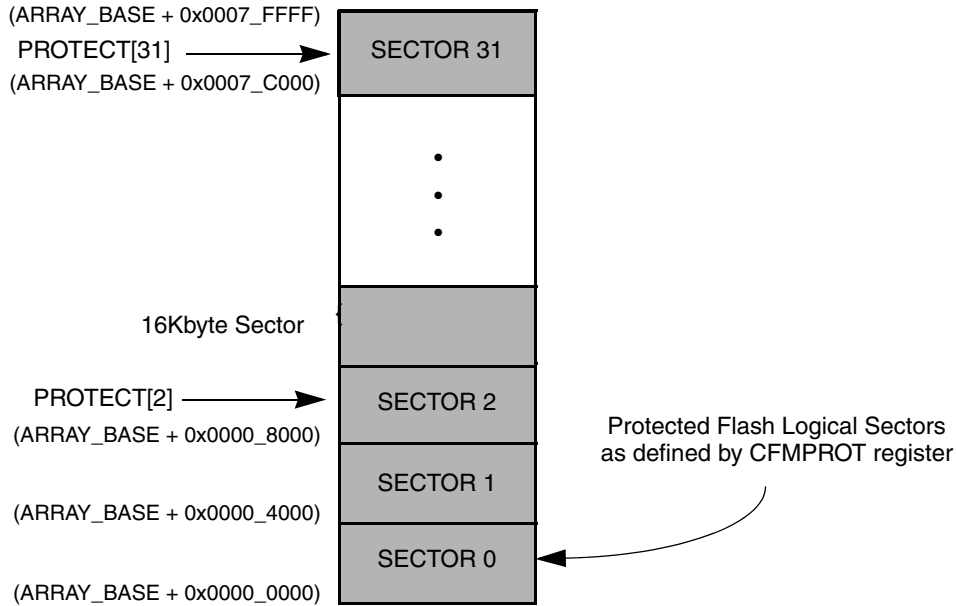
Figure 15-8. CFMPROT Protection Diagram

15.3.4.5 CFM Supervisor Access Register (CFMSACC)

The CFMSACC specifies the supervisor/user access permissions of Flash logical sectors.

	31		16
Field		SUPV	
Reset		See Note	
R/W		R/W	

	15		0
Field		SUPV	
Reset		See Note	
R/W		R/W	
Address		IPSBAR + 0x1D_0014	

Note: The CFMPROT register is loaded at reset from the Flash Supervisor/user Space Restrictions longword stored at the array base address + 0x0000_040C.

Figure 15-9. CFM Supervisor Access Register (CFMSACC)

Table 15-8. CFMSACC Field Descriptions

Bits	Name	Description
31–0	SUPV[31:0]	Supervisor address space assignment. The SUPV[31:0] bits are always readable and only writable when LOCK = 0. Each Flash logical sector can be mapped into supervisor or unrestricted address space. CFMSACC uses the same correspondence between logical sectors and register bits as does CFMPROT. See Figure 15-8 for details. When a logical sector is mapped into supervisor address space, only CPU supervisor accesses will be allowed. A CPU user access to a location in supervisor address space will result in a cycle termination transfer error. When a logical sector is mapped into unrestricted address space both supervisor and user accesses are allowed. 1 Logical sector is mapped in supervisor address space. 0 Logical sector is mapped in unrestricted address space.

15.3.4.6 CFM Data Access Register (CFMDACC)

The CFMDACC specifies the data/program access permissions of Flash logical sectors.

	31				16
Field			DATA		
Reset			See Note		
R/W			R/W		

	15				0
Field			DATA		
Reset			See Note		
R/W			R/W		
Address			IPSBAR + 0x1D_0018		

Note: The CFMPROT register is loaded at reset from the Flash Program/Data Space Restrictions longword stored at the array base address + 0x0000_0410.

Figure 15-10. CFM Data Access Register (CFMDACC)

Table 15-9. CFMDACC Field Descriptions

Bits	Name	Description
31–0	DATA[31:0]	Data address space assignment. The DATA[31:0] bits are always readable and only writable when LOCK = 0. Each Flash logical sector can be mapped into data or both data and program address space. CFMDACC uses the same correspondence between logical sectors and register bits as does CFMPROT. See Figure 15-8 for details. When a logical sector is mapped into data address space, only CPU data accesses will be allowed. A CPU program access to a location in data address space will result in a cycle termination transfer error. When an array sector is mapped into both data and program address space both data and program accesses are allowed. 1 Logical sector is mapped in data address space. 0 Logical sector is mapped in data and program address space.

15.3.4.7 CFM User Status Register (CFMUSTAT)

The CFMUSTAT reports Flash state machine command status, array access errors, protection violations, and blank check status.

	7	6	5			1	0
Field	CBEIF	CCIF	PVIOL	ACCERR	—	BLANK	—
Reset				1100_0000			
R/W	R/W	R			R/W		
Address				IPSBAR + 0x1D_0020			

Figure 15-11. CFM User Status Register (CFMUSTAT)

NOTE

Only one CFMUSTAT bit should be cleared at a time.

Table 15-10. CFMUSTAT Field Descriptions

Bits	Name	Description
7	CBEIF	Command buffer empty interrupt flag. The CBEIF flag indicates that the command buffer for the interleaved Flash physical blocks is empty and that a new command sequence can be started. Clear CBEIF by writing it to 1. Writing a 0 to CBEIF has no effect but can be used to abort a command sequence. The CBEIF bit can trigger an interrupt request if the CBEIE bit is set in CFMMCR. While CBEIF is clear, the CFMCMD register is not writable. 1 Command buffer is ready to accept a new command. 0 Command buffer is full.
6	CCIF	Command complete interrupt flag. The CCIF flag indicates that no commands are pending for the Flash physical blocks. CCIF is set and cleared automatically upon start and completion of a command. Writing to CCIF has no effect. The CCIF bit can trigger an interrupt request if the CCIE bit is set in CFMCR. 1 All commands are completed 0 Command in progress
5	PVIOL	Protection violation flag. The PVIOL flag indicates an attempt was made to initiate a program or erase operation in a Flash logical sector denoted as protected by CFMPROT. Clear PVIOL by writing it to 1. Writing a 0 to PVIOL has no effect. While PVIOL is set in any this register, it is not possible to launch another command. 1 A protection violation has occurred 0 No failure
4	ACCERR	Access error flag. The ACCERR flag indicates an illegal access to the CFM array or registers caused by a bad program or erase sequence. ACCERR is cleared by writing it to 1. Writing a 0 to ACCERR has no effect. While ACCERR is set in this register, it is not possible to launch another command. See Section 15.4.3.4, "Flash User Mode Illegal Operations," for details on what sets the ACCERR flag. 1 Access error has occurred 0 No failure
3	—	Reserved, should be cleared.
2	BLANK	Erase Verified Flag. The BLANK flag indicates that the erase verify command (RDARY1) has checked the two interleaved Flash physical blocks and found them to be blank. Clear BLANK by writing it to 1. Writing a 0 has no effect. 1 Flash physical blocks verify as erased. 0 If an erase verify command has been requested, and the CCIF flag is set, then the selected Flash physical blocks are not blank.
1–0	—	Reserved, should be cleared.

15.3.4.8 CFM Command Register (CFMCMD)

The CFMCMD is the register to which Flash program, erase, and verify commands are written.

	7	6	0
Field	—	CMD	
Reset	0000_0000		
R/W	R/W		
Address	IPSBAR + 0x1D_0024		

Figure 15-12. CFM Command Register (CFMCMD)

MCF5213 Reference Manual, Rev. 1.1

Preliminary

Table 15-11. CFMCMD Field Descriptions

Bits	Name	Description
7	—	Reserved, should be cleared.
6–0	CMD[6:0]	Command. Valid Flash user mode commands are shown in Table 15-12. Writing a command in user mode other than those listed in Table 15-12 will set the ACCERR flag in CFMUSTAT.

CFMCMD is readable and writable in all modes. Writes to bit 7 have no effect and reads return 0.

Table 15-12. CFMCMD User Mode Commands

Command	Name	Description
0x05	RDARY1	Erase verify (all 1s)
0x20	PGM	Longword program
0x40	PGERS	Page erase
0x41	MASERS	Mass erase
0x06	PGERSVER	Page erase verify

15.4 CFM Operation

The CFM registers, subject to the restrictions previously noted, can generally be read and written (see Section 15.3.4, "Register Descriptions" for details). Reads of the CFM array occur normally and writes behave according to the setting of the KEYACC bit in CFMCR. Program, erase, and verify operations are initiated by the CPU. Special cases of user mode apply when the CPU is in low-power or debug modes and when the MCU boots in master mode or emulation mode.

15.4.1 Read Operations

A valid read operation occurs whenever a transfer request is initiated by the ColdFire core, the address is equal to an address within the valid range of the CFM memory space, and the read/write control indicates a read cycle.

In order to reduce power at low system clock frequencies, the sense amplifier timeout (SATO) block minimizes the time during which the sense amplifiers are enabled for read operations. The sense amplifier enable signals to the Flash timeout after approximately 50 ns.

15.4.2 Write Operations

A valid write operation occurs whenever a transfer request is initiated by the ColdFire core, the address is equal to an address within the valid range of the CFM memory space, and the read/write control indicates a write cycle.

The action taken on a valid CFM array write depends on the subsequent user command issued as part of a valid command sequence. Only aligned 32-bit write operations are allowed to the CFM array. Byte and word write operations will result in a cycle termination transfer error.

15.4.3 Program and Erase Operations

Read and write operations are both used for the program and erase algorithms described in this subsection. These algorithms are controlled by a state machine whose timebase is derived from the CFM module clock via a programmable counter.

The command register and associated address and data buffers operate as a two stage FIFO so that a new command along with the necessary address and data can be stored while the previous command is still in progress. This pipelining speeds when programming more than one longword on a specific row, as the charge pumps can be kept on in between two programming commands, thus saving the overhead needed to set up the charge pumps. Buffer empty and command completion are indicated by flags in the CFM user status register. Interrupts will be requested if enabled.

15.4.3.1 Setting the CFMCLKD Register

Prior to issuing any program or erase commands, CFMCLKD must be written to set the Flash state machine clock (FCLK). The CFM module runs at the system clock frequency ÷ 2, but FCLK must be divided down from this frequency to a frequency between 150 kHz and 200 kHz. Use the following procedure to set the PRDIV8 and DIV[5:0] bits in CFMCLKD:

1. If $f_{SYS} \div 2$ is greater than 12.8 MHz, PRDIV8 = 1; otherwise PRDIV8 = 0.
2. Determine DIV[5:0] by using the following equation. Keep only the integer portion of the result and discard any fraction. Do not round the result.

$$\text{DIV[5:0]} = \frac{f_{SYS}}{2 \times 200\text{kHz} \times (1 + (\text{PRDIV8} \times 7))}$$

3. Thus the Flash state machine clock will be:

$$f_{CLK} = \frac{f_{SYS}}{2 \times (\text{DIV[5:0]} + 1) \times (1 + (\text{PRDIV8} \times 7))}$$

Consider the following example for f_{SYS} = 66 MHz:

$$\text{DIV[5:0]} = \frac{f_{SYS}}{2 \times 200\text{kHz} \times (1 + (\text{PRDIV8} \times 7))}$$

$$= \frac{66 \text{ MHz}}{400 \text{ kHz} \times (1 + (1 \times 7))} = 20$$

$$f_{CLK} = \frac{f_{SYS}}{2 \times (\text{DIV[5:0]} + 1) \times (1 + (\text{PRDIV8} \times 7))}$$

$$= \frac{66 \text{ MHz}}{2 \times (20 + 1) \times (1 + (1 \times 7))} = 196.43 \text{ kHz}$$

So, for $f_{SYS} = 66$ MHz, writing 0x54 to CFMCLKD will set f_{CLK} to 196.43 kHz which is a valid frequency for the timing of program and erase operations.

WARNING

For proper program and erase operations, it is critical to set f_{CLK} between 150 kHz and 200 kHz. Array damage due to overstress can occur when f_{CLK} is less than 150 kHz. Incomplete programming and erasure can occur when f_{CLK} is greater than 200 kHz.

NOTE

Command execution time increases proportionally with the period of f_{CLK}.

When CFMCLKD is written, the DIVLD bit is set automatically. If DIVLD is 0, CFMCLKD has not been written since the last reset. Program and erase commands will not execute if this register has not been written (see Section 15.4.3.4, "Flash User Mode Illegal Operations").

15.4.3.2   Program, Erase, and Verify Sequences

A command state machine is used to supervise the write sequencing of program, erase, and verify commands. To prepare for a command, the CFMUSTAT[CBEIF] flag should be tested to ensure that the address, data, and command buffers are empty. If CBEIF is set, the command write sequence can be started.

This three-step command write sequence must be strictly followed. No intermediate writes to the CFM module are permitted between these three steps. The command write sequence is:

1. Write the 32-bit longword to be programmed to its location in the CFM array. The address and data will be stored in internal buffers. All address bits are valid for program commands. The value of the data written for verify and erase commands is ignored. For mass erase or verify, the address can be any location in the CFM array. For page erase, address bits [9:0] are ignored.

NOTE

The page erase command operates simultaneously on adjacent erase pages in two interleaved Flash physical blocks. Thus, a single erase page is effectively 2 Kbyte.

2. Write the program, erase, or verify command to CFMCMD, the command buffer. See Section 15.4.3.3, "Flash Valid Commands."

3. Launch the command by writing a 1 to the CBEIF flag. This clears CBEIF. When command execution is complete, the Flash state machine sets the CCIF flag. The CBEIF flag is also set again, indicating that the address, data, and command buffers are ready for a new command sequence to begin.

The Flash state machine flags errors in command write sequences by means of the ACCERR and PVIOL flags in the CFMUSTAT register. An erroneous command write sequence self-aborts and sets the appropriate flag. The ACCERR or PVIOL flags must be cleared before commencing another command write sequence.

NOTE

By writing a 0 to CBEIF, a command sequence can be aborted after the longword write to the CFM array or the command write to the CFMCMD and before the command is launched. The ACCERR flag will be set on aborted commands and must be cleared before a new command write sequence.

A summary of the programming algorithm is shown in Figure 15-13. The flow is similar for the erase and verify algorithms with the exceptions noted in step 1 above.

15.4.3.3 Flash Valid Commands

Table 15-13 summarizes the valid Flash user commands.

Table 15-13. Flash User Commands

CFMCMD	Meaning	Description
0x05	Erase verify	Verify that all 256 Kbytes of Flash from two interleaving physical blocks are erased. If both blocks are erased, the BLANK bit will be set in the CFMUSTAT register upon command completion.
0x20	Program	Program a 32-bit longword.
0x40	Page erase	Erase 2 Kbyte of Flash. Two 1024-byte pages from interleaving physical blocks are erased in this operation.
0x41	Mass erase	Erase all 256 Kbytes of Flash from two interleaving physical blocks. A mass erase is only possible when no PROTECT bits are set for that block.
0x06	Page erase verify	Verify that the two 1024-byte pages are erased. If both pages are erased, the BLANK bit will be set in the CFMUSTAT register upon command completion.
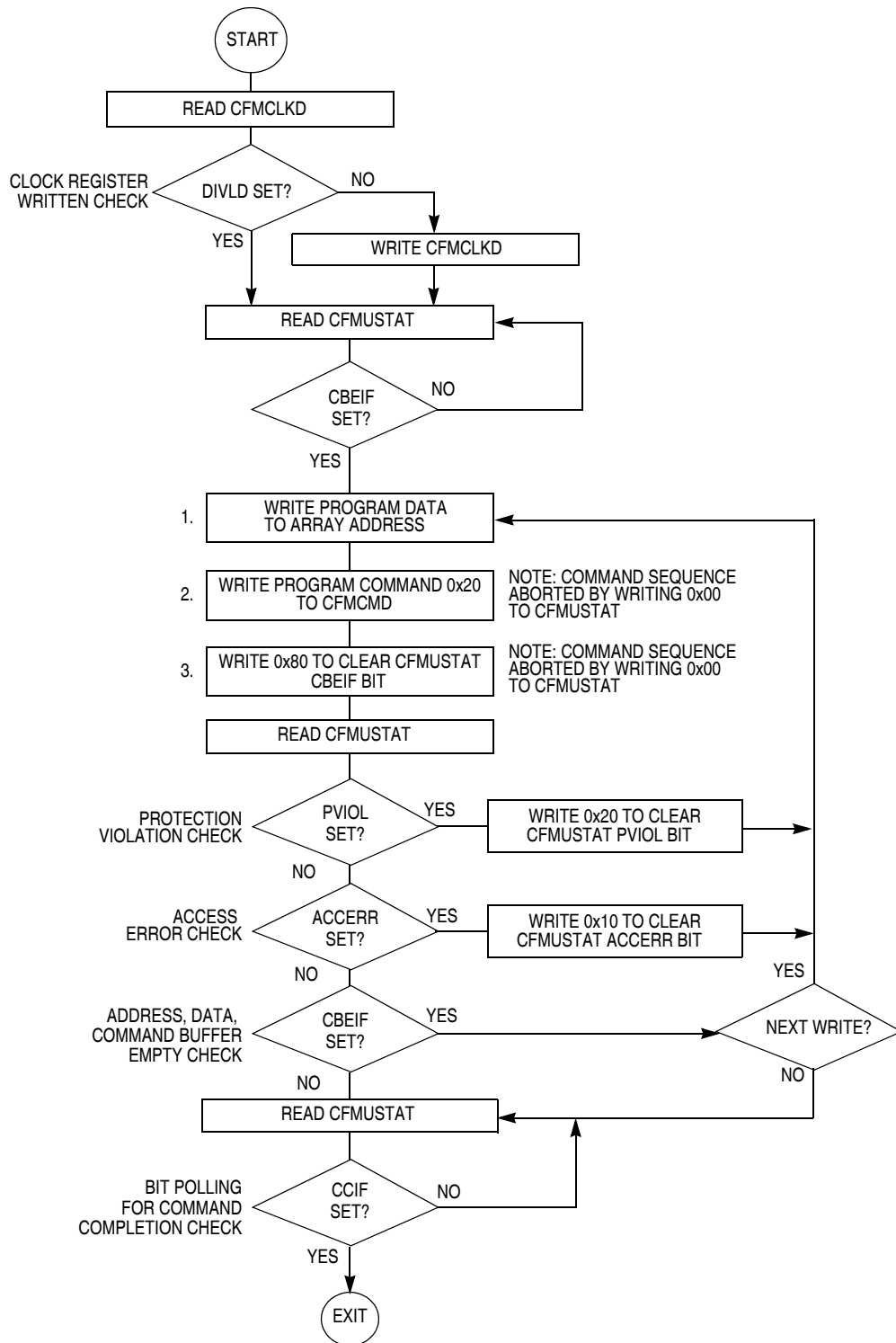
Figure 15-13. Example Program Algorithm

15.4.3.4 Flash User Mode Illegal Operations

The ACCERR flag will be set during a command write sequence if any of the illegal operations below are performed. Such operations will cause the command sequence to immediately abort.

1. Writing to the CFM array before initializing CFMCLKD.
2. Writing to the CFM array while in emulation mode.
3. Writing a byte or a word to the CFM array. Only 32-bit longword programming is allowed.
4. Writing to the CFM array while CBEIF is not set.
5. Writing an invalid user command to the CFMCMD.
6. Writing to any CFM other than CFMCMD after writing a longword to the CFM array.
7. Writing a second command to CFMCMD before executing the previously written command.
8. Writing to any CFM register other than CFMUSTAT (to clear CBEIF) after writing to the command register.
9. Entering stop mode while a program or erase command is in progress.
10. Aborting a command sequence by writing a 0 to CBEIF after the longword write to the CFM array or after writing a command to CFMCMD and before launching it.

The PVIOL flag will be set during a command write sequence after the longword write to the CFM array if any of the illegal operations below are performed. Such operations will cause the command sequence to immediately abort.

1. Writing to an address in a protected area of the CFM array.
2. Writing a mass erase command to CFMCMD while any logical sector is protected (see Section 15.3.4.4, "CFM Protection Register (CFMPROT)").

If a Flash physical block is read during a program or erase operation on that block (CFMUSTAT bit CCIF = 0), the read will return non-valid data and the ACCERR flag will not be set.

15.4.4 Stop Mode

If a command is active (CCIF = 0) when the MCU enters stop mode, the command sequence monitor performs the following:

1. The command in progress aborts
2. The Flash high voltage circuitry switches off and any pending command (CBEIF = 0) does not executed when the MCU exits stop mode.
3. The CCIF and ACCERR flags are set if a command is active when the MCU enters stop mode.

NOTE

The state of any longword(s) being programmed or any erase pages/physical blocks being erased is not guaranteed if the MCU enters stop mode with a command in progress.

WARNING

Active commands are immediately aborted when the MCU enters stop mode. Do not execute the STOP instruction during program and erase operations.

15.5 Flash Security Operation

The CFM array provides security information to the integration module and the rest of the MCU. A longword in the Flash configuration field stores this information. This longword is read automatically after each reset and is stored in the CFMSEC register.

NOTE

Enabling Flash security will disable BDM communications.

In user mode, security can be bypassed via a back door access scheme using an 8-byte long key. Upon successful completion of the back door access sequence, the module output signal and status bit indicating that the chip is secure are cleared.

The CFM may be unsecured via one of two methods:

1. Executing a back door access scheme.
2. Passing an erase verify check.

15.5.1 Back Door Access

If the KEYEN bit is set, security can be bypassed by:

1. Setting the KEYACC bit in the CFM configuration register (CFMMCR).
2. Writing the correct 8-byte back door comparison key to the CFM array at addresses 0x0000_0400 to 0x0000_0407. This operation must consist of two 32-bit writes to address 0x0000_0400 and 0x0000_0404 in that order. The two back door write cycles can be separated by any number of bus cycles.
3. Clearing the KEYACC bit.
4. If all 8 bytes written match the array contents at addresses 0x0000_0400 to 0x0000_0407, then security is bypassed until the next reset.

NOTE

The security of the Flash as defined by the Flash security longword at address 0x0000_0414 is not changed by the back door method of unsecuring the device. After the next reset the device is again secured and the same back door key remains in effect unless changed by program or erase operations. The back door method of unsecuring the device has no effect on the program and erase protections defined by the CFM protection register (CFMPROT).

15.5.2 Erase Verify Check

Security can be disabled by verifying that the CFM array is blank. If required, the mass erase command can be executed for each pair of Flash physical blocks that comprise the array. The erase verify command must then be executed for all Flash physical blocks within the array. The CFM will be unsecured if the erase verify command determines that the entire array is blank. After the next reset, the security state of the CFM will be determined by the Flash security longword, which, after being erased, will read 0xffff_ffff, thus unsecuring the module.

15.6 Reset

The CFM array is not accessible for any operations via the address and data buses during reset. If a reset occurs while any command is in progress that command will immediately abort. The state of any longword being programmed or any erase pages/physical blocks being erased is not guaranteed.

15.7 Interrupts

The CFM module can request an interrupt when all commands are completed or when the address, data, and command buffers are empty. Table 15-14 shows the CFM interrupt mechanism.

Table 15-14. CFM Interrupt Sources

Interrupt Source	Interrupt Flag	Local Enable
Command, data and address buffers empty	CBEIF (CFMUSTAT)	CBEIE (CFMCR)

Table 15-14. CFM Interrupt Sources

Interrupt Source	Interrupt Flag	Local Enable
All commands are completed	CCIF (CFMUSTAT)	CCIE (CFMCR)
Access error	ACCERR (CFMUSTAT)	AEIE (CFMCR)

Chapter 16
EzPort

EzPort is a serial Flash programming interface that allows the Flash memory contents on a 32 bit general purpose microcontroller to be read, erased and programmed from off-chip in a compatible format to many standalone Flash memory chips.

16.1 Features

The EzPort includes the following features:

- Serial interface that is compatible with a subset of the SPI format.
- Able to read, erase and program flash memory.
- Able to reset the micro-controller, allowing it to boot from the flash memory after the memory has been configured.

16.2 Modes of Operation

The EzPort can operate in one of two different modes, enabled or disabled.

- Enabled

  When enabled, the EzPort 'steals' access to the Flash memory, preventing access from other cores or peripherals. The rest of the micro-controller is disabled when the EzPort is enabled to avoid conflicts.

- Disabled

  When the EzPort is disabled, the rest of the micro-controller can access Flash memory as normal.

Figure 16-1 is a block diagram of the EzPort.
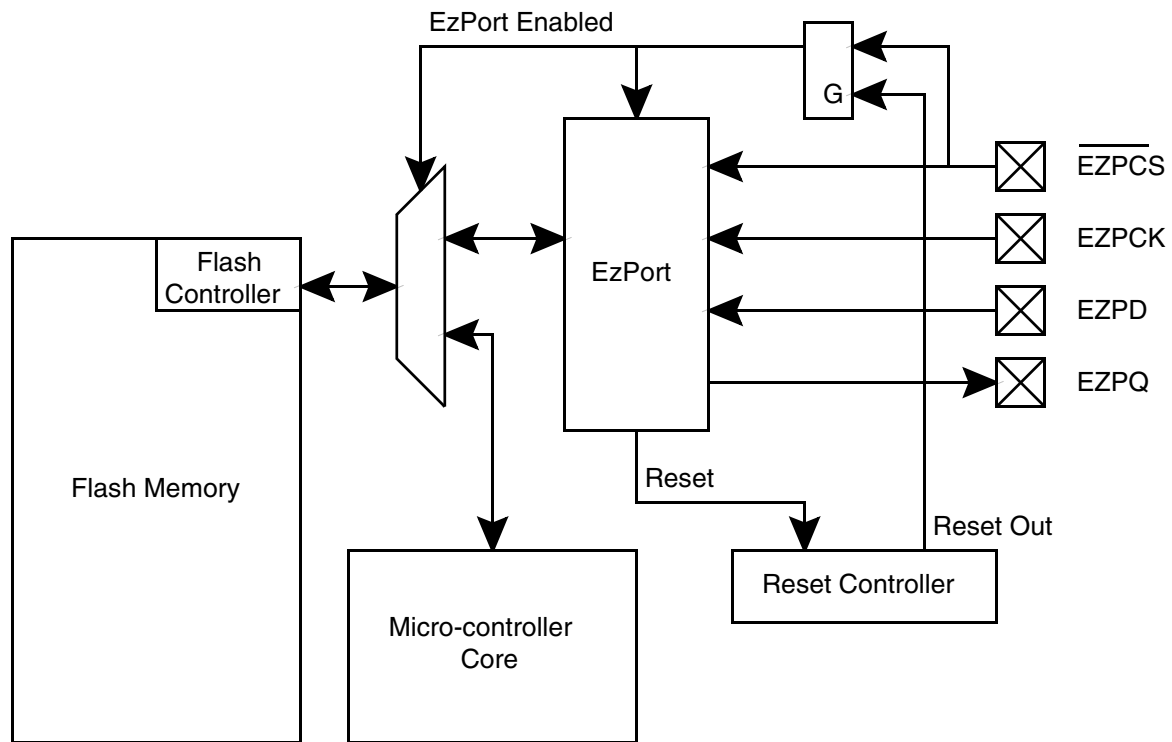
Figure 16-1. EzPort Block Diagram

16.3 External Signal Description

16.3.1 Overview

Table 16-1 contains a list of EzPort external signals.

Table 16-1. Signal Descriptions

Name	Description	I/O
EZPCK	EzPort Clock	Input
$\overline{\text{EZPCS}}$	EzPort Chip Select	Input
EZPD	EzPort Serial Data In	Input
EZPQ	EzPort Serial Data Out	Output

16.3.2 Detailed Signal Descriptions

16.3.2.1 EZPCK — EzPort Clock

Serial clock for data transfers. Serial Data In (EZPD) and Chip Select ($\overline{\text{EZPCS}}$) are registered on the rising edge of EZPCK while Serial Data Out (EZPQ) is driven on the falling edge of EZPCK. The maximum frequency of the EzPort clock is half the system clock frequency for all commands except when executing

MCF5213 Reference Manual, Rev. 1.1

Freescale Semiconductor

the Read Data command. When executing the Read Data command, the EzPort clock has a maximum frequency of one eighth the system clock frequency.

16.3.2.2 $\overline{\text{EZPCS}}$ — EzPort Chip Select

Chip select for signalling the start and end of serial transfers. If $\overline{\text{EZPCS}}$ is asserted during and when the micro-controller's reset out signal is negated then EzPort is enabled out of reset; otherwise it is disabled. Once EzPort is enabled, asserting $\overline{\text{EZPCS}}$ commences a serial data transfer, which continues until $\overline{\text{EZPCS}}$ is negated again.. The negation of $\overline{\text{EZPCS}}$ indicates the current command is finished and resets the EzPort state machine so that it is ready to receive the next command.

16.3.2.3 EZPD — EzPort Serial Data In

Serial data in for data transfers. Serial Data In (EZPD) is registered on the rising edge of EZPCK. All commands, addresses and data are shifted in most significant bit first. When EzPort is driving output data on EZPQ, the data shifted in EZPD is ignored.

16.3.2.4 EZPQ — EzPort Serial Data Out

Serial data out for data transfers. Serial Data Out (EZPQ) is driven on the falling edge of EZPCK. It is tri-stated unless $\overline{\text{EZPCS}}$ is asserted and the EzPort is driving data out. All data is shifted out most significant bit first.

16.4 Command Definition

The EzPort receives commands from an external device and translates those commands into flash memory accesses. Table 16-2 lists the supported commands.

Table 16-2. EzPort Commands

Command	Description	Code	Address Bytes	Dummy Bytes	Data Bytes	Compatible Commands[1]
WREN	Write Enable	0x06	0	0	0	WREN
WRDI	Write Disable	0x04	0	0	0	WRDI
RDSR	Read Status Register	0x05	0	0	1	RDSR
WRCR	Write Config Register	0x01	0	0	1	WRSR
READ	Read Data	0x03	3	0	1+	READ
FAST_READ	Read Data at High Speed	0x0B	3	1	1+	FAST_READ
PP	Page Program	0x02	3	0	4 to 256	PP
SE	Sector Erase	0xD8	3	0	0	SE
BE	Bulk Erase	0xC7	0	0	0	BE
RESET	Reset Chip	0xB9	0	0	0	DP

[1]Lists the compatible commands on the ST Microelectronics Serial Flash Memory parts.

16.4.1 Command Descriptions

16.4.1.1 Write Enable

The Write Enable command sets the write enable register bit in the status register. The write enable bit must be set for a Write Configuration Register (WRCR), Page Program (PP), Sector Erase (SE) or Bulk Erase (BE) command to be accepted. The write enable register bit clears on reset, on a Write Disable command and at the completion of a write, program or erase command.

This command should not be used if a write is already in progress.

16.4.1.2 Write Disable

The Write Disable command clears the write enable register bit in the status register.

This command should not be used if a write is already in progress.

16.4.1.3 Read Status Register

The Read Status Register command returns the contents of the EzPort Status register.

EzPort Status Register

	7	6	5	4	3	2	1	0
R	FS	WEF	CRL				WEN	WIP
W								
RESET:	0/1[1]	0	0	0	0	0	0	0

☐ Unimplemented or reserved.

[1]Reset value reflects if Flash Security is enabled or disabled out of reset.

Write In Progress (WIP)

> Status flag that sets after a Write Configuration Register (WRCR), Page Program (PP), Sector Erase (SE) or Bulk Erase (BE) command is accepted and clears once the flash memory erase or program is completed. Only the Read Status Register (RDSR) command is accepted while a write is in progress.
> 1 = Write is in progress. Only accept RDSR command.
> 0 = Write is not in progress.Accept any command.

Write Enable (WEN)

> Control bit that must be set before a Write Configuration Register (WRCR), Page Program (PP), Sector Erase (SE) or Bulk Erase (BE) command is accepted. Is set by the Write Enable (WREN) command and cleared by reset or a Write Disable (WRDI) command. It also clears on completion of a write, erase or program command.
> 1 = Enables the following write, erase or program command.
> 0 = Disables the following write, erase or program command.

Configuration Register Loaded (CRL)

Status flag that indicates if the configuration register has been loaded. The configuration register initializes the Flash controllers clock configuration register to generate a divided down clock from the system clock that runs at a frequency of 150kHz to 200 kHz. This register must be initialized before any erase or program commands are accepted.

   1 = Configuration register has been loaded, erase and program commands are accepted.
   0 = Configuration register has not been loaded, erase and program commands are not accepted.

Write Error Flag (WEF)

   Status flag that indicates if there has been an error with an erase or program instruction inside the flash controller due to attempting to program or erase a protected sector, or if there is an error in the flash memory after performing a Bulk Erase command. The flag clears after a Read Status Register (RDSR) command.

   1 = Error on previous erase/program command.
   0 = No error on previous erase/program command.

Flash Security (FS)

   Status flag that indicates if the flash memory is in secure mode. In secure mode, the following commands are not accepted: Read (READ), Fast Read (FAST_READ), Page Program (PP), Sector Erase (SE). Secure mode can be exited by performing a Bulk Erase (BE) command (which erases the entire contents of the flash memory).

   1 = Flash is in secure mode.
   0 = Flash is not in secure mode.

16.4.1.4 Write Configuration Register

The Write Configuration Command updates the Flash controller's Clock Configuration register. The clock configuration register divides down the Flash controller's internal system clock to a 150kHz to 200 kHz clock. This register must be initialized before any erase or program commands are issued to the Flash controller.

This command should not be used if the write error flag is set, a write is in progress or the configuration register has already been loaded (as it is a write-once register).

EzPort Configuration Register

	7	6	5	4	3	2	1	0
R								
W		PRDIV8			DIV[5:0]			
:								
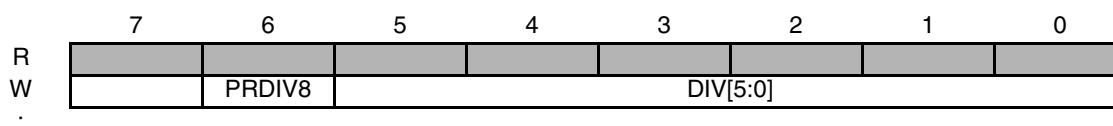
Unimplemented or reserved.

PRDIV

   Enables prescaler divide by 8.
   1 = Enables a prescaler that divides the system clock by 8 before it enters the divider.
   0 = The system clock is fed directly into the divider.

DIV[5:0]

Clock divider field. The combination of PRDIV8 and DIV[5:0] effectively divides the system clock down to a frequency between 150 kHz and 200 kHz.

16.4.1.5   Read Data

The Read Data command returns data from the flash memory, starting at the address specified in the command word. Data will continue being returned for as long as the EzPort chip select ($\overline{\text{EZPCS}}$) is asserted, with the address automatically incrementing. When the address reaches the highest Flash memory address, it will wrap around to the lowest Flash memory address. In this way, the entire contents of the Flash memory can be returned by one command.

For this command to return the correct data, the EzPort Clock (EZPCK) must run at no more than divide by eight of the internal system clock.

This command should not be used if the write error flag is set, or a write is in progress. This command is not accepted if Flash security is enabled.

16.4.1.6   Read Data at High Speed

This command is identical to the Read Data command, except for the inclusion of a dummy byte following the address bytes and before the first data byte is returned.

This allows the command to run at any frequency of the EzPort Clock (EZPCK) up to and including half the internal system clock frequency of the micro-controller.This command should not be used if the write error flag is set, or a write is in progress. This command is not accepted if Flash security is enabled.

16.4.1.7   Page Program

The Page Program command programs locations in Flash memory that have previously been erased. The starting address of the memory to program is sent after the command word and must be a 32 bit aligned address (the two LSBs must be zero). After every four bytes of data are received by the EzPort, that 32 bit word is programmed into Flash memory with the address automatically incrementing after each write. For this reason, the number of bytes to program must be a multiple of four. Only a maximum of 256 bytes can be programmed at a time; when the address reaches the highest address within any given 256 byte space of memory, it will wrap around to the lowest address in that 256 byte space of memory.

This command should not be used if the write error flag is set, a write is in progress, the write enable bit is not set or the Configuration register has not been written.This command is not accepted if Flash security is enabled.

The Write Error Flag will set if there is an attempt to program a protected area of the Flash memory.

16.4.1.8   Sector Erase

The Sector Erase command erases the contents of a 2Kbyte space of Flash memory. The three byte address sent after the command byte can be any address within the space to erase.

This command should not be used if the write error flag is set, a write is in progress, the write enable bit is not set or the Configuration register has not been written. This command is not accepted if Flash security is enabled.

The Write Error Flag will set if there is an attempt to erase a protected area of the Flash memory.

16.4.1.9 Bulk Erase

The Bulk Erase command erases the entire contents of Flash memory, ignoring any protected sectors or Flash security. The write error flag will set if the Bulk Erase command does not successfully erase the entire contents of flash memory. Flash security will be disabled if the Bulk Erase command is followed by a Reset Chip command.

This command should not be used if the write error flag is set, a write is in progress, the write enable bit is not set or the Configuration register has not been written.

16.4.1.10 Reset Chip

The Reset Chip command forces the chip into the reset state. If the EzPort chip select ($\overline{\text{EZPCS}}$) pin is asserted at the end of the reset period then EzPort will be enabled, otherwise it will be disabled.

This command allows the chip to boot up from flash memory after it has been programmed by an external source.

This command should not be used if a write is in progress.

16.5 Functional Description

The EzPort provides a simple interface to connect an external device to the Flash memory on board a 32 bit micro-controller. The interface itself is compatible with the SPI interface (with the EzPort operating as a slave) running in either of the two following modes with data transmitted most significant bit first:

- CPOL = 0, CPHA = 0
- CPOL = 1, CPHA = 1

Commands are issued by the external device to erase, program or read the contents of the Flash memory. The serial data out from the EzPort is tri-stated unless data is being driven, allowing the signal to be shared among several different EzPort (or compatible) devices in parallel, provided they have different chip selects.

16.6 Initialization/Application Information

Prior to issuing any program or erase commands, the Clock Configuration register must be written to set the Flash state machine clock (FCLK). The Flash controller module runs at the system clock frequency divide by 2, but FCLK must be divided down from this frequency to a frequency between 150 kHz and 200 kHz. Use the following procedure to set the PRDIV8 and DIV[5:0] bits in the Clock Configuration register.

1. If f_{SYS} is greater than 25.6 MHz, PRDIV8 = 1; otherwise PRDIV8 = 0.

2. Determine DIV[5:0] by using the following equation. Keep only the integer portion of the result and discard any fraction. Do not round the result.

$$DIV = \frac{Fsys}{2 \times 200kHz \times (1 + (PRDIV8 \times 7))}$$

3. Thus the Flash state machine clock will be:

$$Fclk = \frac{Fsys}{2 \times (DIV + 1) \times (1 + (PRDIV8 \times 7))}$$

So, for Fsys = 66 MHz, writing 0x54 to the Clock Configuration register will set Fclk to 196.43 kHz which is a valid frequency for the timing of program and erase operations.

For proper program and erase operations, it is critical to set Fclk between 150 kHz and 200 kHz. Array damage due to overstress can occur when Fclk is less than 150 kHz. Incomplete programming and erasure can occur when Fclk is greater than 200 kHz.

Chapter 17
Programmable Interrupt Timer Modules (PIT0–PIT1)

17.1 Introduction

This chapter describes the operation of the two programmable interrupt timer modules, PIT0–PIT1.

17.1.1 Overview

Each PIT is a 16-bit timer that provides precise interrupts at regular intervals with minimal processor intervention. The timer can either count down from the value written in the modulus register, or it can be a free-running down-counter.

17.1.2 Block Diagram
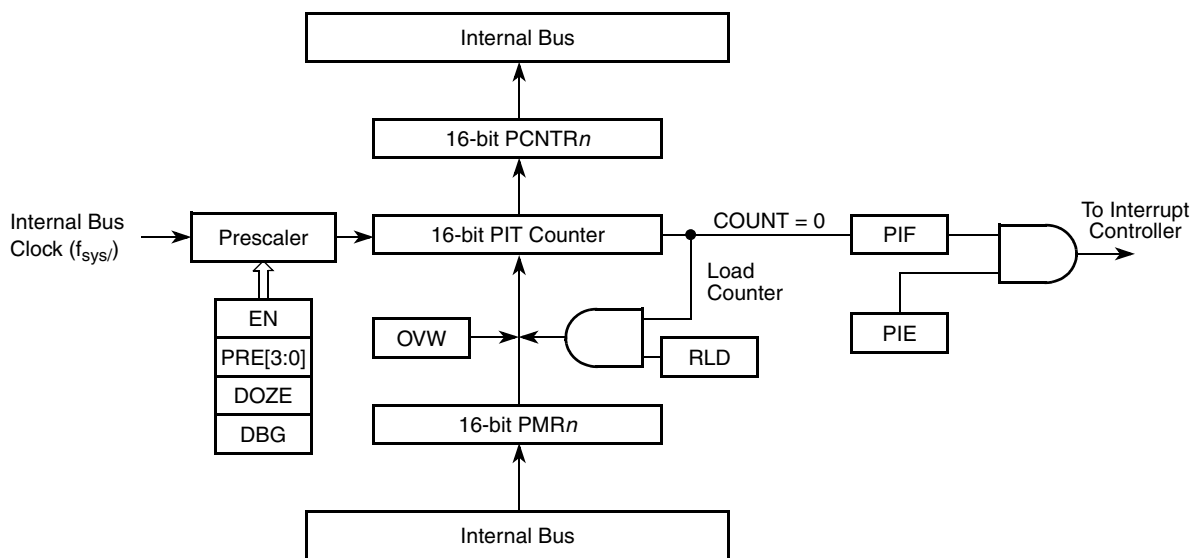
Figure 17-1. PIT Block Diagram

17.1.3 Low-Power Mode Operation

This subsection describes the operation of the PIT modules in low-power modes and debug mode of operation. Low-power modes are described in the power management module, Chapter 8, "Power Management." Table 17-1 shows the PIT module operation in low-power modes, and how it can exit from each mode.

NOTE

The low-power interrupt control register (LPICR) in the system control module specifies the interrupt level at or above which the device can be brought out of a low-power mode.

Table 17-1. PIT Module Operation in Low-power Modes

Low-power Mode	PIT Operation	Mode Exit
Wait	Normal	N/A
Doze	Normal if PCSRn[DOZE] cleared, stopped otherwise	Any interrupt at or above level in LPICR, will exit doze mode if PCSRn[DOZE] is set. Otherwise interrupt assertion has no effect.
Stop	Stopped	No
Debug	Normal if PCSRn[DBG] cleared, stopped otherwise	No. Any interrupt will be serviced upon normal exit from debug mode

In wait mode, the PIT module continues to operate as in run mode and can be configured to exit the low-power mode by generating an interrupt request. In doze mode with the PCSRn[DOZE] bit set, PIT module operation stops. In doze mode with the PCSRn[DOZE] bit cleared, doze mode does not affect PIT operation. When doze mode is exited, the PIT continues to operate in the state it was in prior to doze mode. In stop mode, the internal bus clock is absent, and PIT module operation stops.

In debug mode with the PCSRn[DBG] bit set, PIT module operation stops. In debug mode with the PCSRn[DBG] bit cleared, debug mode does not affect PIT operation. When debug mode is exited, the PIT continues to operate in its pre-debug mode state, but any updates made in debug mode remain.

17.2 Memory Map/Register Definition

This section contains a memory map, shown in Table 17-2, and describes the register structure for PIT0–PIT1.

Table 17-2. Programmable Interrupt Timer Modules Memory Map

IPSBAR Offset PIT 0 PIT 1	Register	Access[1]	Reset Value	Section/Page
	Supervisor Access Only Registers[2]			
0x15_0000 0x16_0000	PIT Control and Status Register (PCSRn)	R/W	0x0000	17.2.1/17-3
0x15_0002 0x16_0002	PIT Modulus Register (PMRn)	R/W	0x0000	17.2.2/17-5

Table 17-2. Programmable Interrupt Timer Modules Memory Map (continued)

IPSBAR Offset PIT 0 PIT 1	Register	Access[1]	Reset Value	Section/Page
User/Supervisor Access Registers				
0x15_0004 0x16_0004	PIT Count Register (PCNTR*n*)	R	0xFFFF	17.2.3/17-5

[1] Accesses to reserved address locations have no effect and result in a cycle termination transfer error.

[2] User mode accesses to supervisor only addresses have no effect and result in a cycle termination transfer error.

17.2.1 PIT Control and Status Register (PCSR*n*)

The PCSR*n* registers configure the corresponding timer's operation.

IPSBAR 0x15_0000 (PCSR0)
Offset: 0x16_0000 (PCSR1)

Access: Supervisor
read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0		PRE			0	DOZE	DBG	OVW	PIE	PIF	RLD	EN
W														w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
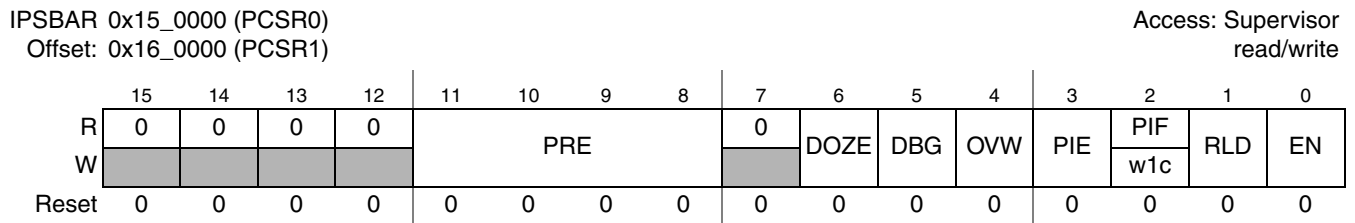
Figure 17-2. PIT Control and Status Register (PCSR*n*)

MCF5213 Reference Manual, Rev. 1.1

Preliminary

Table 17-3. PCSR*n* Field Descriptions

Field	Description
15–12	Reserved, should be cleared.
11–8 PRE	Prescaler. The read/write prescaler bits select the internal bus clock divisor to generate the PIT clock. To accurately predict the timing of the next count, change the PRE[3:0] bits only when the enable bit (EN) is clear. Changing PRE[3:0] resets the prescaler counter. System reset and the loading of a new value into the counter also reset the prescaler counter. Setting the EN bit and writing to PRE[3:0] can be done in this same write cycle. Clearing the EN bit stops the prescaler counter. <table><tr><th>PRE</th><th>Internal Bus Clock Divisor</th><th>Decimal Equivalent</th></tr><tr><td>0000</td><td>2^0</td><td>1</td></tr><tr><td>0001</td><td>2^1</td><td>2</td></tr><tr><td>0010</td><td>2^2</td><td>4</td></tr><tr><td>...</td><td>...</td><td>...</td></tr><tr><td>1101</td><td>2^{13}</td><td>8192</td></tr><tr><td>1110</td><td>2^{14}</td><td>16384</td></tr><tr><td>1111</td><td>2^{15}</td><td>32768</td></tr></table>
7	Reserved, should be cleared.
6 DOZE	Doze mode bit. The read/write DOZE bit controls the function of the PIT in doze mode. Reset clears DOZE. 0 PIT function not affected in doze mode 1 PIT function stopped in doze mode. When doze mode is exited, timer operation continues from the state it was in before entering doze mode.
5 DBG	Debug mode bit. Controls the function of the PIT in halted/debug mode. Reset clears DBG. During debug mode, register read and write accesses function normally. When debug mode is exited, timer operation continues from the state it was in before entering debug mode, but any updates made in debug mode remain. 0 PIT function not affected in debug mode 1 PIT function stopped in debug mode Note: Changing the DBG bit from 1 to 0 during debug mode starts the PIT timer. Likewise, changing the DBG bit from 0 to 1 during debug mode stops the PIT timer.
4 OVW	Overwrite. Enables writing to PMR*n* to immediately overwrite the value in the PIT counter. 0 Value in PMR*n* replaces value in PIT counter when count reaches 0x0000. 1 Writing PMR*n* immediately replaces value in PIT counter.
3 PIE	PIT interrupt enable. This read/write bit enables the PIF flag to generate interrupt requests. 0 PIF interrupt requests disabled 1 PIF interrupt requests enabled
2 PIF	PIT interrupt flag. This read/write bit is set when the PIT counter reaches 0x0000. Clear PIF by writing a 1 to it or by writing to PMR. Writing 0 has no effect. Reset clears PIF. 0 PIT count has not reached 0x0000. 1 PIT count has reached 0x0000.

Table 17-3. PCSR*n* Field Descriptions (continued)

Field	Description
1 RLD	Reload bit. The read/write reload bit enables loading the value of PMR*n* into the PIT counter when the count reaches 0x0000. 0 Counter rolls over to 0xFFFF on count of 0x0000 1 Counter reloaded from PMR*n* on count of 0x0000
0 EN	PIT enable bit. Enables PIT operation. When the PIT is disabled, the counter and prescaler are held in a stopped state. This bit is read anytime, write anytime. 0 PIT disabled 1 PIT enabled

17.2.2   PIT Modulus Register (PMR*n*)

The 16-bit read/write PMR*n* contains the timer modulus value that is loaded into the PIT counter when the count reaches 0x0000 and the PCSR*n*[RLD] bit is set.

When the PCSR*n*[OVW] bit is set, PMR*n* is transparent, and the value written to PMR*n* is immediately loaded into the PIT counter. The prescaler counter is reset (0xFFFF) anytime a new value is loaded into the PIT counter and also during reset. Reading the PMR*n* returns the value written in the modulus latch. Reset initializes PMR*n* to 0xFFFF.

IPSBAR 0x15_0002 (PMR0)   Access: Supervisor
  Offset: 0x16_0002 (PMR1)   read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W								PM								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
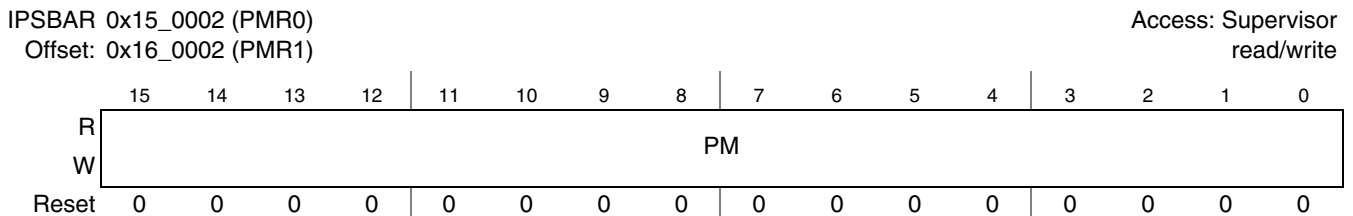
Figure 17-3. PIT Modulus Register (PMR*n*)

17.2.3   PIT Count Register (PCNTR*n*)

The 16-bit, read-only PCNTR*n* contains the counter value. Reading the 16-bit counter with two 8-bit reads is not guaranteed to be coherent. Writing to PCNTR*n* has no effect, and write cycles are terminated normally.

IPSBAR 0x15_0004 (PCNTR0)   Access: User read only
  Offset: 0x16_0004 (PCNTR1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R								PC								
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
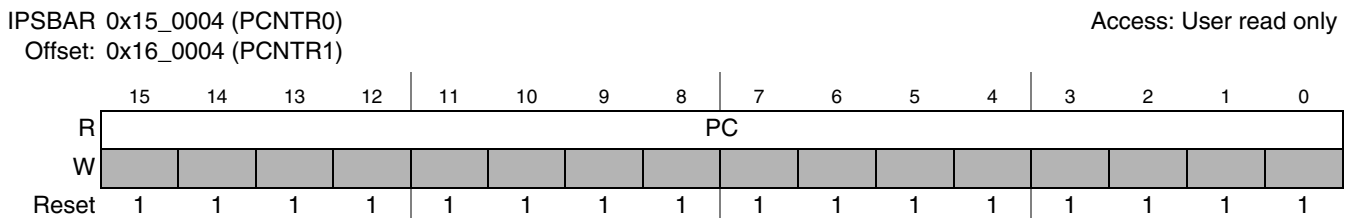
Figure 17-4. PIT Count Register (PCNTR*n*)

17.3   Functional Description

This section describes the PIT functional operation.

Preliminary

17.3.1   Set-and-Forget Timer Operation

This mode of operation is selected when the RLD bit in the PCSR register is set.

When the PIT counter reaches a count of 0x0000, the PIF flag is set in PCSRn. The value in the modulus register is loaded into the counter, and the counter begins decrementing toward 0x0000. If the PCSRn[PIE] bit is set, the PIF flag issues an interrupt request to the CPU.

When the PCSRn[OVW] bit is set, the counter can be directly initialized by writing to PMRn without having to wait for the count to reach 0x0000.
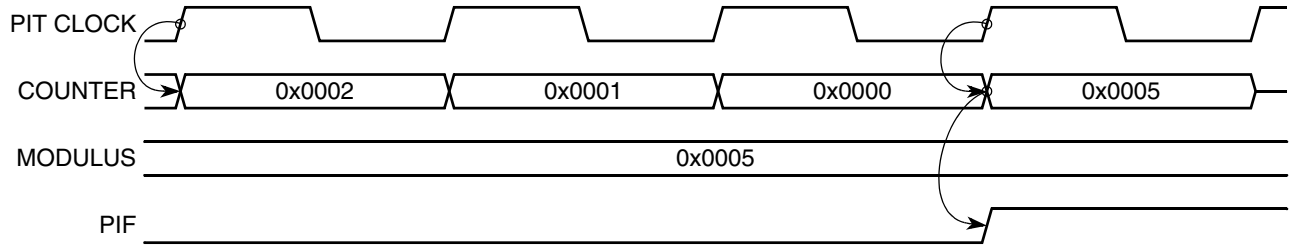
Figure 17-5. Counter Reloading from the Modulus Latch

17.3.2   Free-Running Timer Operation

This mode of operation is selected when the PCSRn[RLD] bit is clear. In this mode, the counter rolls over from 0x0000 to 0xFFFF without reloading from the modulus latch and continues to decrement.

When the counter reaches a count of 0x0000, the PCSRn[PIF] flag is set. If the PCSRn[PIE] bit is set, the PIF flag issues an interrupt request to the CPU.

When the PCSRn[OVW] bit is set, the counter can be directly initialized by writing to PMRn without having to wait for the count to reach 0x0000.
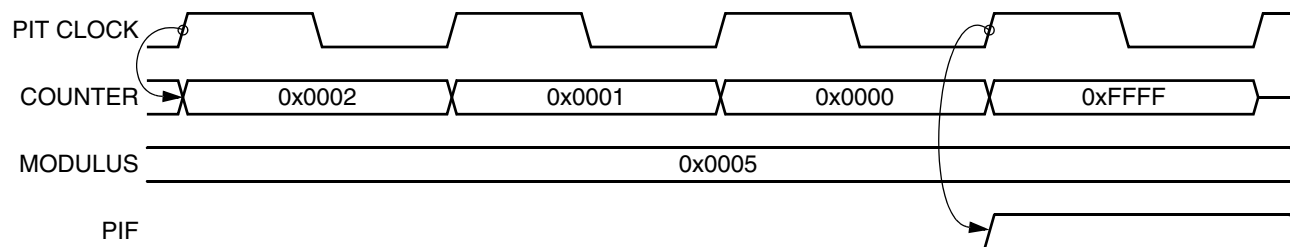
Figure 17-6. Counter in Free-Running Mode

17.3.3   Timeout Specifications

The 16-bit PIT counter and prescaler supports different timeout periods. The prescaler divides the internal bus clock period as selected by the PCSRn[PRE] bits. The PMRn[PM] bits select the timeout period.

17.3.4 Interrupt Operation

Table 17-4 shows the interrupt request generated by the PIT.

Table 17-4. PIT Interrupt Requests

Interrupt Request	Flag	Enable Bit
Timeout	PIF	PIE

The PIF flag is set when the PIT counter reaches 0x0000. The PIE bit enables the PIF flag to generate interrupt requests. Clear PIF by writing a 1 to it or by writing to the PMR.

Chapter 18
General Purpose Timer Module (GPT)

18.1 Introduction

This device has one 4-channel general purpose timer module (GPT). It consists of a 16-bit counter driven by a 7-stage programmable prescaler.

A timer overflow function allows software to extend the timing capability of the system beyond the 16-bit range of the counter. Each of the four timer channels can be configured for input capture, which can capture the time of a selected transition edge, or for output compare, which can generate output waveforms and timer software delays. These functions allow simultaneous input waveform measurements and output waveform generation.

Additionally, one of the channels, channel 3, can be configured as a 16-bit pulse accumulator that can operate as a simple event counter or as a gated time accumulator. The pulse accumulator uses the GPT channel 3 input/output pin in either event mode or gated time accumulation mode.

18.2 Features

Features of the general-purpose timer include:

- Four 16-bit input capture/output compare channels
- 16-bit architecture
- Programmable prescaler
- Pulse widths variable from microseconds to seconds
- Single 16-bit pulse accumulator
- Toggle-on-overflow feature for pulse-width modulator (PWM) generation
- External timer clock input (SYNCA/SYNCB)

18.3 Block Diagram
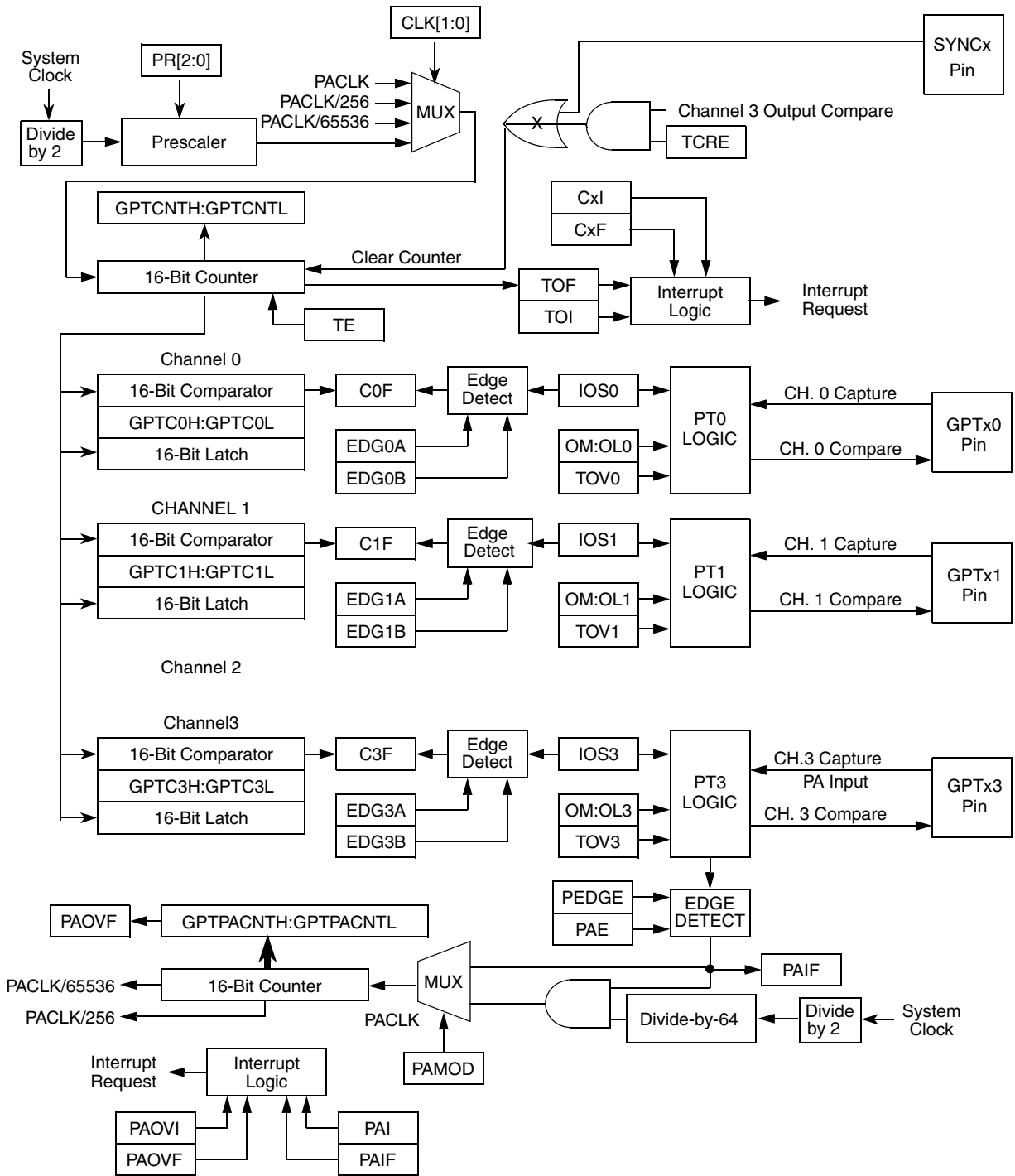
Figure 18-1. GPT Block Diagram

18.4 Low-Power Mode Operation

This subsection describes the operation of the general purpose time module in low-power modes and halted mode of operation. Low-power modes are described in Chapter 7, "Power Management."
Table 18-1 shows the general purpose timer module operation in the low-power modes, and shows how this module may facilitate exit from each mode.

Table 18-1. Watchdog Module Operation in Low-power Modes

Low-power Mode	Watchdog Operation	Mode Exit
Wait	Normal	No
Doze	Normal	No
Stop	Stopped	No
Halted	Normal	No

General purpose timer operation stops in stop mode. When stop mode is exited, the general purpose timer continues to operate in its pre-stop mode state.

18.5 Signal Description

Table 18-2 provides an overview of the signal properties.

Table 18-2. Signal Properties

Pin Name	GPTPORT Register Bit	Function	Reset State	Pull-up
GPT0	PORTT*n*0	GPT channel 0 IC/OC pin	Input	Active
GPT1	PORTT*n*1	GPT channel 1 IC/OC pin	Input	Active
GPT2	PORTT*n*2	GPT channel 2 IC/OC pin	Input	Active
GPT3	PORTT*n*3	GPT channel 3 IC/OC or PA pin	Input	Active
SYNC*n*	PORTE[3:0][1]	GPT counter synchronization	Input	Active

[1] SYNCA is available on either PORTE3 or PORTE1; SYNCB is available on either PORTE2 or PORTE0.

18.5.1 GPT[2:0]

The GPT[2:0] pins are for channel 2–0 input capture and output compare functions. These pins are available for general-purpose input/output (I/O) when not configured for timer functions.

18.5.2 GPT3

The GPT3 pin is for channel 3 input capture and output compare functions or for the pulse accumulator input. This pin is available for general-purpose I/O when not configured for timer functions.

18.5.3 SYNC*n*

The SYNC*n* pin is for synchronization of the timer counter. It can be used to synchronize the counter with externally-timed or clocked events. A high signal on this pin clears the counter.

18.6 Memory Map and Registers

Table 20-3 shows the memory map of the GPT module. The base address for GPT is IPSBAR + 0x1A_0000.

NOTE

Reading reserved or unimplemented locations returns zeroes. Writing to reserved or unimplemented locations has no effect.

Table 18-3. GPT Module Memory Map

IPSBAR Offset	Bits 7–0	Access[1]
0x1A_0000	GPT IC/OC Select Register (GPTIOS)	S
0x1A_0001	GPT Compare Force Register (GPTCFORC)	S
0x1A_0002	GPT Output Compare 3 Mask Register (GPTOC3M)	S
0x1A_0003	GPT Output Compare 3 Data Register (GPTOC3D)	S
0x1A_0004	GPT Counter Register (GPTCNT)	S
0x1A_0005	GPT System Control Register 1 (GPTSCR1)	S
0x1A_0006	Reserved[2]	—
0x1A_0007	GPT Toggle-on-Overflow Register (GPTTOV)	S
0x1A_0008	GPT Control Register 1 (GPTCTL1)	S
0x1A_0009	Reserved[2]	—
0x1A_000A	GPT Control Register 2 (GPTCTL2)	S
0x1A_000B	GPT Interrupt Enable Register (GPTIE)	S
0x1A_000C	GPT System Control Register 2 (GPTSCR2)	S
0x1A_000D	GPT Flag Register 1 (GPTFLG1)	S
0x1A_000E	GPT Flag Register 2 (GPTFLG2)	S
0x1A_000F	GPT Channel 0 Register High (GPTC0H)	S
0x1A_0010	GPT Channel 0 Register Low (GPTC0L)	S
0x1A_0011	GPT Channel 1 Register High (GPTC1H)	S
0x1A_0012	GPT Channel 1 Register Low (GPTC1L)	S
0x1A_0013	GPT Channel 2 Register High (GPTC2H)	S
0x1A_0014	GPT Channel 2 Register Low (GPTC2L)	S
0x1A_0015	GPT Channel 3 Register High (GPTC3H)	S

Table 18-3. GPT Module Memory Map (continued)

IPSBAR Offset	Bits 7–0	Access[1]
0x1A_0016	GPT Channel 3 Register Low (GPTC3L)	S
0x1A_0017	Pulse Accumulator Control Register (GPTPACTL)	S
0x1A_0018	Pulse Accumulator Flag Register (GPTPAFLG)	S
0x1A_0019	Pulse Accumulator Counter Register High (GPTPACNTH)	S
0x1A_001A	Pulse Accumulator Counter Register Low (GPTPACNTL)	S
0x1A_001B	Reserved[2]	—
0x1A_001C	GPT Port Data Register (GPTPORT)	S
0x1A_001D	GPT Port Data Direction Register (GPTDDR)	S
0x1A_001E	GPT Test Register (GPTTST)	S

[1] S = CPU supervisor mode access only.

[2] Writes have no effect, reads return 0s, and the access terminates without a transfer error exception.

18.6.1 GPT Input Capture/Output Compare Select Register (GPTIOS)

	7			4	3			0
Field	—				IOS			
Reset	0000_0000							
R/W	R/W							
Address	IPSBAR + 0x00_0400, 0x00_0440, 0x00_0480, 0x00_04C0							

Figure 18-2. GPT Input Capture/Output Compare Select Register (GPTIOS)

Table 18-4. GPTIOS Field Descriptions

Bit(s)	Name	Description
7–4	—	Reserved, should be cleared.
3–0	IOS	I/O select. The IOS[3:0] bits enable input capture or output compare operation for the corresponding timer channels. These bits are read anytime (always read 0x00), write anytime. 1 Output compare enabled 0 Input capture enabled

18.6.2 GPT Compare Force Register (GPCFORC)

	7		4	3				0
Field		—				FOC		
Reset				0000_0000				
R/W				R/W				
Address			IPSBAR + 0x00_0401, 0x00_0441, 0x00_0481, 0x00_04C1					
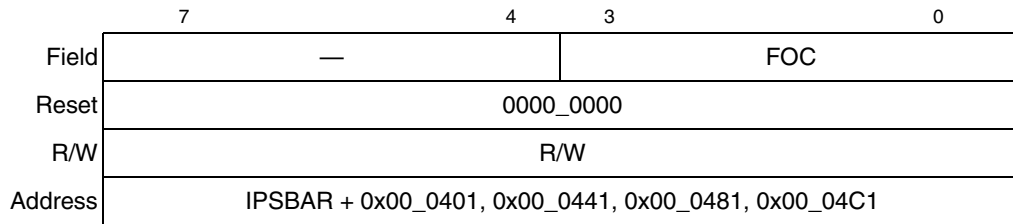
Figure 18-3. GPT Input Compare Force Register (GPCFORC)

Table 18-5. GPTCFORC Field Descriptions

Bit(s)	Name	Description
7–4	—	Reserved, should be cleared.
3–0	FOC	Force output compare.Setting an FOC bit causes an immediate output compare on the corresponding channel. Forcing an output compare does not set the output compare flag. These bits are read anytime, write anytime. 1 Force output compare 0 No effect

NOTE

A successful channel 3 output compare overrides any compare on channels 2:0. For each OC3M bit that is set, the output compare action reflects the corresponding OC3D bit.

18.6.3 GPT Output Compare 3 Mask Register (GPTOC3M)

	7		4	3				0
Field		—				OC3M		
Reset				0000_0000				
R/W				R/W				
Address			IPSBAR + 0x00_0402, 0x00_0442, 0x00_0482, 0x00_04C2					
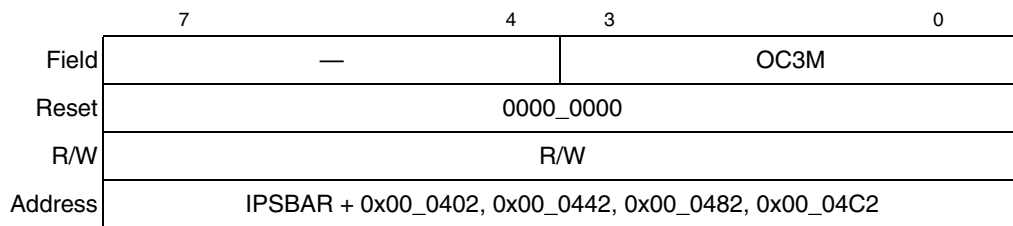
Figure 18-4. GPT Output Compare 3 Mask Register (GPTOC3M)

Table 18-6. GPTOC3M Field Descriptions

Bit(s)	Name	Description
7–4	—	Reserved, should be cleared.
3–0	OC3M	Output compare 3 mask. Setting an OC3M bit configures the corresponding PORTT*n* pin to be an output. OC3M*n* makes the GPT port pin an output regardless of the data direction bit when the pin is configured for output compare (IOSx = 1). The OC3M*n* bits do not change the state of the PORTT*n*DDR bits. These bits are read anytime, write anytime. 1 Corresponding PORTT*n* pin configured as output 0 No effect

18.6.4 GPT Output Compare 3 Data Register (GPTOC3D)

	7	4	3	0
Field	—		OC3D	
Reset	0000_0000			
R/W	R/W			
Address	IPSBAR + 0x00_0403, 0x00_0443, 0x00_0483, 0x00_04C3			

Figure 18-5. GPT Output Compare 3 Data Register (GPTOC3D)

Table 18-7. GPTOC3D Field Descriptions

Bit(s)	Name	Description
7–4	—	Reserved, should be cleared.
3–0	OC3D	Output compare 3 data. When a successful channel 3 output compare occurs, these bits transfer to the PORTT*n* data register if the corresponding OC3M*n* bits are set. These bits are read anytime, write anytime.

NOTE

A successful channel 3 output compare overrides any channel 2:0 compares. For each OC3M bit that is set, the output compare action reflects the corresponding OC3D bit.

18.6.5 GPT Counter Register (GPTCNT)

	15	0
Field	CNTR	
Reset	0000_0000_0000_0000	
R/W	Read only	
Address	IPSBAR + 0x00_0404, 0x00_0444, 0x00_0484, 0x00_04C4	

Figure 18-6. GPT Counter Register (GPTCNT)

Table 18-8. GPTCNT Field Descriptions

Bit(s)	Name	Description
15–0	CNTR	Read-only field that provides the current count of the timer counter. To ensure coherent reading of the timer counter, such that a timer rollover does not occur between two back-to-back 8-bit reads, it is recommended that only word (16-bit) accesses be used. A write to GPTCNT may have an extra cycle on the first count because the write is not synchronized with the prescaler clock. The write occurs at least one cycle before the synchronization of the prescaler clock. These bits are read anytime. They should be written to only in test (special) mode; writing to them has no effect in normal modes.

18.6.6 GPT System Control Register 1 (GPTSCR1)

	7	6	5	4	3			0
Field	GPTEN	—		TFFCA	—			
Reset	0000_0000							
R/W	R/W							
Address	IPSBAR + 0x00_0405, 0x00_0445, 0x00_0485, 0x00_04C5							

Figure 18-7. GPT System Control Register 1 (GPTSCR1)

Table 18-9. GPTSCR1 Field Descriptions

Bit(s)	Name	Description
7	GPTEN	Enables the general purpose timer. When the timer is disabled, only the registers are accessible. Clearing GPTEN reduces power consumption. These bits are read anytime, write anytime. 1 GPT enabled 0 GPT and GPT counter disabled
6–5	—	Reserved, should be cleared.
4	TFFCA	Timer fast flag clear all. Enables fast clearing of the main timer interrupt flag registers (GPTFLG1 and GPTFLG2) and the PA flag register (GPTPAFLG). TFFCA eliminates the software overhead of a separate clear sequence. See Figure 18-8. When TFFCA is set: • An input capture read or a write to an output compare channel clears the corresponding channel flag, CxF. • Any access of the GPT count registers (GPTCNTH/L) clears the TOF flag. • Any access of the PA counter registers (GPTPACNT) clears both the PAOVF and PAIF flags in GPTPAFLG. Writing logic 1s to the flags clears them only when TFFCA is clear. 1 Fast flag clearing 0 Normal flag clearing
3–0	—	Reserved, should be cleared.

Figure 18-8. Fast Clear Flag Logic

18.6.7 GPT Toggle-On-Overflow Register (GPTTOV)

Figure 18-9. GPT Toggle-On-Overflow Register (GPTTOV)

Table 18-10. GPTTOV Field Description

Bit(s)	Name	Description
7–4	—	Reserved, should be cleared.
3–0	TOV	Toggles the output compare pin on overflow for each channel. This feature only takes effect when in output compare mode. When set, it takes precedence over forced output compare but not channel 3 override events. These bits are read anytime, write anytime. 1 Toggle output compare pin on overflow feature enabled 0 Toggle output compare pin on overflow feature disabled

18.6.8 GPT Control Register 1 (GPTCTL1)

Figure 18-10. GPT Control Register 1 (GPTCTL1)

MCF5213 Reference Manual, Rev. 1.1

Preliminary

Table 18-11. GPTCL1 Field Descriptions

Bit(s)	Name	Description
7–0	OMx/OLx	Output mode/output level. Selects the output action to be taken as a result of a successful output compare on each channel. When either OM*n* or OL*n* is set and the IOS*n* bit is set, the pin is an output regardless of the state of the corresponding DDR bit. These bits are read anytime, write anytime. 00 GPT disconnected from output pin logic 01 Toggle OC*n* output line 10 Clear OC*n* output line 11 Set OC*n* line Note: Channel 3 shares a pin with the pulse accumulator input pin. To use the PAI input, clear both the OM3 and OL3 bits and clear the OC3M3 bit in the output compare 3 mask register.

18.6.9 GPT Control Register 2 (GPTCTL2)

	7	6	5	4	3	2	1	0
Field	EDG3B	EDG3A	EDG2B	EDG2A	EDG1B	EDG1A	EDG0B	EDG0A
Reset	0000_0000							
R/W	R/W							
Address	IPSBAR + 0x00_040B, 0x00_044B							

Figure 18-11. GPT Control Register 2 (GPTCTL2)

Table 18-12. GPTLCTL2 Field Descriptions

Bit(s)	Name	Description
7–0	EDG*n*[B:A]	Input capture edge control. Configures the input capture edge detector circuits for each channel. These bits are read anytime, write anytime. 00 Input capture disabled 01 Input capture on rising edges only 10 Input capture on falling edges only 11 Input capture on any edge (rising or falling)

18.6.10 GPT Interrupt Enable Register (GPTIE)

	7	6	5	4	3			0
Field	—				CI			
Reset	0000_0000							
R/W	R/W							
Address	IPSBAR + 0x00_040C, 0x00_044C							

Figure 18-12. GPT Interrupt Enable Register (GPTIE)

Table 18-13. GPTIE Field Descriptions

Bit(s)	Name	Description
7–4	—	Reserved, should be cleared.
3–0	C*n*I	Channel interrupt enable. Enables the C[3:0]F flags in GPT flag register 1 to generate interrupt requests for each channel. These bits are read anytime, write anytime. 1 Corresponding channel interrupt requests enabled 0 Corresponding channel interrupt requests disabled

18.6.11 GPT System Control Register 2 (GPTSCR2)

	7	6	5	4	3	2		0
Field	TOI	—	PUPT	RDPT	TCRE		PR	
Reset	0000_0000
R/W	R/W
Address	IPSBAR + 0x00_040D, 0x00_044D

Figure 18-13. GPT System Control Register 2 (GPTSCR2)

Table 18-14. GPTSCR2 Field Descriptions

Bit(s)	Name	Description
7	TOI	Enables timer overflow interrupt requests. 1 Overflow interrupt requests enabled 0 Overflow interrupt requests disabled
6	—	Reserved, should be cleared.
5	PUPT	Enables pull-up resistors on the GPT ports when the ports are configured as inputs. 1 Pull-up resistors enabled 0 Pull-up resistors disabled
4	RDPT	GPT drive reduction. Reduces the output driver size. 1 Output drive reduction enabled 0 Output drive reduction disabled
3	TCRE	Enables a counter reset after a channel 3 compare. 1 Counter reset enabled 0 Counter reset disabled Note: When the GPT channel 3 registers contain 0x0000 and TCRE is set, the GPT counter registers remain at 0x0000 all the time. When the GPT channel 3 registers contain 0xFFFF and TCRE is set, TOF does not get set even though the GPT counter registers go from 0xFFFF to 0x0000.

Table 18-14. GPTSCR2 Field Descriptions (continued)

Bit(s)	Name	Description
2–0	PR*n*	Prescaler bits. Select the prescaler divisor for the GPT counter. 000 Prescaler divisor 1 001 Prescaler divisor 2 010 Prescaler divisor 4 011 Prescaler divisor 8 100 Prescaler divisor 16 101 Prescaler divisor 32 110 Prescaler divisor 64 111 Prescaler divisor 128 Note: The newly selected prescaled clock does not take effect until the next synchronized edge of the prescaled clock when the clock count transitions to 0x0000.)

18.6.12   GPT Flag Register 1 (GPTFLG1)

	7	6	5	4	3			0
Field		—				CF		
Reset				0000_0000				
R/W				R/W				
Address			IPSBAR + 0x00_040E, 0x00_044E					
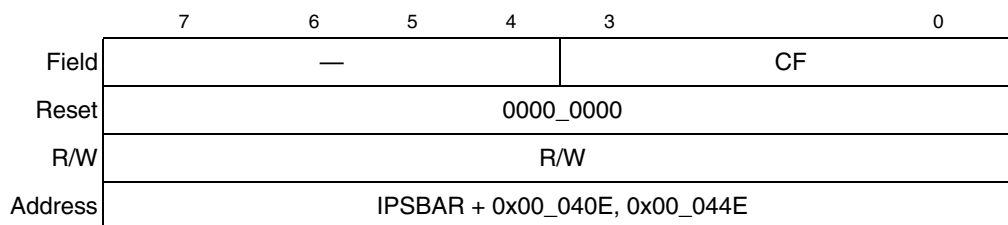
Figure 18-14. GPT Flag Register 1 (GPTFLG1)

Table 18-15. GPTFLG1 Field Descriptions

Bit(s)	Name	Description
7–4	—	Reserved, should be cleared.
3–0	C*n*F	Channel flags. A channel flag is set when an input capture or output compare event occurs. These bits are read anytime, write anytime (writing 1 clears the flag, writing 0 has no effect). Note: When the fast flag clear all bit, GPTSCR1[TFFCA], is set, an input capture read or an output compare write clears the corresponding channel flag. When a channel flag is set, it does not inhibit subsequent output compares or input captures.

18.6.13   GPT Flag Register 2 (GPTFLG2)

	7	6	5	4	3			0
Field	TOF		—			CF		
Reset				0000_0000				
R/W				R/W				
Address			IPSBAR + 0x00_040F, 0x00_044F					
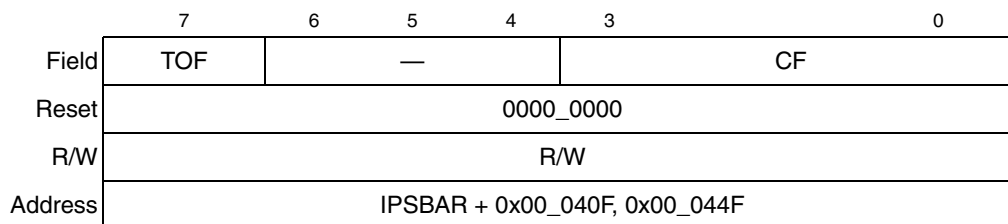
Figure 18-15. GPT Flag Register 2 (GPTFLG2)

Table 18-16. GPTFLG2 Field Descriptions

Bit(s)	Name	Description
7	TOF	Timer overflow flag. Set when the GPT counter rolls over from 0xFFFF to 0x0000. If the TOI bit in GPTSCR2 is also set, TOF generates an interrupt request. This bit is read anytime, write anytime (writing 1 clears the flag, and writing 0 has no effect). 1 Timer overflow 0 No timer overflow Note: When the GPT channel 3 registers contain 0xFFFF and TCRE is set, TOF does not get set even though the GPT counter registers go from 0xFFFF to 0x0000. When TOF is set, it does not inhibit subsequent overflow events.
6–4	—	Reserved, should be cleared.
3–0	C*n*F	Channel flags. A channel flag is set when an input capture or output compare event occurs. These bits are read anytime, write anytime (writing 1 clears the flag, writing 0 has no effect).

Note: When the fast flag clear all bit, GPTSCR1[TFFCA], is set, any access to the GPT counter registers clears GPT flag register 2.

18.6.14 GPT Channel Registers (GPTC*n*)

	15		0
Field		CCNT	
Reset		0000_0000_0000_0000	
R/W		R/W	
Address		IPSBAR + 0x00_0410, 0x00_0412, 0x00_0414, 0x00_0416, 0x1B_0010, 0x1B_0012, 0x1B_0014, 0x1B_0016	

Figure 18-16. GPT Channel[0:3] Register (GPTC*n*)

Table 18-17. GPTC*n* Field Descriptions

Bit(s)	Name	Description
15–0	CCNT	When a channel is configured for input capture (IOS*n* = 0), the GPT channel registers latch the value of the free-running counter when a defined transition occurs on the corresponding input capture pin. When a channel is configured for output compare (IOS*n* = 1), the GPT channel registers contain the output compare value. To ensure coherent reading of the GPT counter, such that a timer rollover does not occur between back-to-back 8-bit reads, it is recommended that only word (16-bit) accesses be used. These bits are read anytime, write anytime (for the output compare channel); writing to the input capture channel has no effect.

18.6.15 Pulse Accumulator Control Register (GPTPACTL)

	7	6	5	4	3			0
Field	—	PAE	PAMOD	PEDGE	CLK		PAOVI	PAI
Reset	0000_0000							
R/W	R/W							
Address	IPSBAR + 0x00_0418, 0x1B_0018							

Figure 18-17. Pulse Accumulator Control Register (GPTPACTL)

Table 18-18. GPTPACTL Field Descriptions

Bit(s)	Name	Description
7	—	Reserved, should be cleared.
6	PAE	Enables the pulse accumulator. 1 Pulse accumulator enabled 0 Pulse accumulator disabled Note: The pulse accumulator can operate in event mode even when the GPT enable bit, GPTEN, is clear.
5	PAMOD	Pulse accumulator mode. Selects event counter mode or gated time accumulation mode. 1 Gated time accumulation mode 0 Event counter mode
4	PEDGE	Pulse accumulator edge. Selects falling or rising edges on the PAI pin to increment the counter. In event counter mode (PAMOD = 0): 1 Rising PAI edge increments counter 0 Falling PAI edge increments counter In gated time accumulation mode (PAMOD = 1): 1 Low PAI input enables divide-by-64 clock to pulse accumulator and trailing rising edge on PAI sets PAIF flag. 0 High PAI input enables divide-by-64 clock to pulse accumulator and trailing falling edge on PAI sets PAIF flag. Note: The timer prescaler generates the divide-by-64 clock. If the timer is not active, there is no divide-by-64 clock. To operate in gated time accumulation mode: 1. Apply logic 0 to $\overline{RSTI}$ pin. 2. Initialize registers for pulse accumulator mode test. 3. Apply appropriate level to PAI pin. 4. Enable GPT.
3–2	CLK	Select the GPT counter input clock. Changing the CLK bits causes an immediate change in the GPT counter clock input. 00 GPT prescaler clock (When PAE = 0, the GPT prescaler clock is always the GPT counter clock.) 01 PACLK 10 PACLK/256 11 PACLK/65536

Table 18-18. GPTPACTL Field Descriptions (continued)

Bit(s)	Name	Description
1	PAOVI	Pulse accumulator overflow interrupt enable. Enables the PAOVF flag to generate interrupt requests. 1 PAOVF interrupt requests enabled 0 PAOVF interrupt requests disabled
0	PAI	Pulse accumulator input interrupt enable. Enables the PAIF flag to generate interrupt requests. 1 PAIF interrupt requests enabled 0 PAIF interrupt requests disabled

18.6.16 Pulse Accumulator Flag Register (GPTPAFLG)

	7					2	1	0
Field	—						PAOVF	PAIF
Reset	0000_0000							
R/W	R/W							
Address	IPSBAR + 0x00_0419, 0x1B_0019							

Figure 18-18. Pulse Accumulator Flag Register (GPTPAFLG)

Table 18-19. GPTPAFLG Field Descriptions

Bit(s)	Name	Description
7–2	—	Reserved, should be cleared.
1	PAOVF	Pulse accumulator overflow flag. Set when the 16-bit pulse accumulator rolls over from 0xFFFF to 0x0000. If the GPTPACTL[PAOVI] bit is also set, PAOVF generates an interrupt request. Clear PAOVF by writing a 1 to it. This bit is read anytime, write anytime. (Writing 1 clears the flag; writing 0 has no effect.) 1 Pulse accumulator overflow 0 No pulse accumulator overflow
0	PAIF	Pulse accumulator input flag. Set when the selected edge is detected at the PAI pin. In event counter mode, the event edge sets PAIF. In gated time accumulation mode, the trailing edge of the gate signal at the PAI pin sets PAIF. If the PAI bit in GPTPACTL is also set, PAIF generates an interrupt request. Clear PAIF by writing a 1 to it. 1 Active PAI input 0 No active PAI input

NOTE

When the fast flag clear all enable bit, GPTSCR1[TFFCA], is set, any access to the pulse accumulator counter registers clears all the flags in GPTPAFLG.

MCF5213 Reference Manual, Rev. 1.1

Preliminary

18.6.17 Pulse Accumulator Counter Register (GPTPACNT)

	15		0
Field		PACNT	
Reset		0000_0000_0000_0000	
R/W		R/W	
Address		IPSBAR + 0x00_041A, 0x1B_001B	

Figure 18-19. Pulse Accumulator Counter Register (GPTPACNT)

Table 18-20. GPTPACR Field Descriptions

Bit(s)	Name	Description
15–0	PACNT	Contains the number of active input edges on the PAI pin since the last reset. Note: Reading the pulse accumulator counter registers immediately after an active edge on the PAI pin may miss the last count since the input first has to be synchronized with the bus clock. To ensure coherent reading of the PA counter, such that the counter does not increment between back-to-back 8-bit reads, it is recommended that only word (16-bit) accesses be used. These bits are read anytime, write anytime.

18.6.18 GPT Port Data Register (GPTPORT)

	7	6	5	4	3	0
Field		—			PORTT	
Reset		0000_0000				
R/W		R/W				
Address		IPSBAR + 0x00_041D, 0x1B_001D				

Figure 18-20. GPT Port Data Register (GPTPORT)

Table 18-21. GPTPORT Field Descriptions

Bit(s)	Name	Description
7–4	—	Reserved, should be cleared.
3–0	PORTT	GPT port input capture/output compare data. Data written to GPTPORT is buffered and drives the pins only when they are configured as general-purpose outputs. Reading an input (DDR bit = 0) reads the pin state; reading an output (DDR bit = 1) reads the latched value. Writing to a pin configured as a GPT output does not change the pin state. These bits are read anytime (read pin state when corresponding PORTTn bit is 0, read pin driver state when corresponding GPTDDR bit is 1), write anytime.

18.6.19 GPT Port Data Direction Register (GPTDDR)

	7	6	5	4	3		0
Field		—				DDRT	
GPT Function		—				IC/OC	
Pulse Accumulator Function		—			PAI	—	
Reset				0000_0000			
R/W				R/W			
Address			IPSBAR + 0x00_041E, 0x1B_001E				

Figure 18-21. GPT Port Data Direction Register (GPTDDR)

Table 18-22. GPTDDR Field Descriptions

Bit(s)	Name	Description
7–4	—	Reserved, should be cleared.
3–0	DDRT	Control the port logic of PORTT*n*. Reset clears the PORTT*n* data direction register, configuring all GPT port pins as inputs. These bits are read anytime, write anytime. 1 Corresponding pin configured as output 0 Corresponding pin configured as input

18.7 Functional Description

The General Purpose Timer (GPT) module is a 16-bit, 4-channel timer with input capture and output compare functions and a pulse accumulator.

18.7.1 Prescaler

The prescaler divides the module clock by 1 or 16 . The PR[2:0] bits in GPTSCR2 select the prescaler divisor.

18.7.2 Input Capture

Clearing an I/O select bit, IOS*n*, configures channel *n* as an input capture channel. The input capture function captures the time at which an external event occurs. When an active edge occurs on the pin of an input capture channel, the timer transfers the value in the GPT counter into the GPT channel registers, GPTC*n*.

The minimum pulse width for the input capture input is greater than two module clocks.

The input capture function does not force data direction. The GPT port data direction register controls the data direction of an input capture pin. Pin conditions such as rising or falling edges can trigger an input capture only on a pin configured as an input.

An input capture on channel *n* sets the C*n*F flag. The C*n*I bit enables the C*n*F flag to generate interrupt requests.

18.7.3   Output Compare

Setting an I/O select bit, IOS*n*, configures channel *n* as an output compare channel. The output compare function can generate a periodic pulse with a programmable polarity, duration, and frequency. When the GPT counter reaches the value in the channel registers of an output compare channel, the timer can set, clear, or toggle the channel pin. An output compare on channel *n* sets the C*n*F flag. The C*n*I bit enables the C*n*F flag to generate interrupt requests.

The output mode and level bits, OM*n* and OL*n*, select, set, clear, or toggle on output compare. Clearing both OM*n* and OL*n* disconnects the pin from the output logic.

Setting a force output compare bit, FOC*n*, causes an output compare on channel *n*. A forced output compare does not set the channel flag.

A successful output compare on channel 3 overrides output compares on all other output compare channels. A channel 3 output compare can cause bits in the output compare 3 data register to transfer to the GPT port data register, depending on the output compare 3 mask register. The output compare 3 mask register masks the bits in the output compare 3 data register. The GPT counter reset enable bit, TCRE, enables channel 3 output compares to reset the GPT counter. A channel 3 output compare can reset the GPT counter even if the OC3/PAI pin is being used as the pulse accumulator input.

An output compare overrides the data direction bit of the output compare pin but does not change the state of the data direction bit.

Writing to the PORTT*n* bit of an output compare pin does not affect the pin state. The value written is stored in an internal latch. When the pin becomes available for general-purpose output, the last value written to the bit appears at the pin.

18.7.4   Pulse Accumulator

The pulse accumulator (PA) is a 16-bit counter that can operate in two modes:

1. Event counter mode: counts edges of selected polarity on the pulse accumulator input pin, PAI
2. Gated time accumulation mode: counts pulses from a divide-by-64 clock

The PA mode bit, PAMOD, selects the mode of operation.

The minimum pulse width for the PAI input is greater than two module clocks.

18.7.5   Event Counter Mode

Clearing the PAMOD bit configures the PA for event counter operation. An active edge on the PAI pin increments the PA. The PA edge bit, PEDGE, selects falling edges or rising edges to increment the PA.

An active edge on the PAI pin sets the PA input flag, PAIF.   The PA input interrupt enable bit, PAI, enables the PAIF flag to generate interrupt requests.

NOTE

The PAI input and GPT channel 3 use the same pin. To use the PAI input, disconnect it from the output logic by clearing the channel 3 output mode and output level bits, OM3 and OL3. Also clear the channel 3 output compare 3 mask bit, OC3M3.

The PA counter register, GPTPACNT, reflects the number of active input edges on the PAI pin since the last reset.

The PA overflow flag, PAOVF, is set when the PA rolls over from 0xFFFF to 0x0000. The PA overflow interrupt enable bit, PAOVI, enables the PAOVF flag to generate interrupt requests.

NOTE

The PA can operate in event counter mode even when the GPT enable bit, GPTEN, is clear.

18.7.6   Gated Time Accumulation Mode

Setting the PAMOD bit configures the PA for gated time accumulation operation. An active level on the PAI pin enables a divide-by-64 clock to drive the PA. The PA edge bit, PEDGE, selects low levels or high levels to enable the divide-by-64 clock.

The trailing edge of the active level at the PAI pin sets the PA input flag, PAIF. The PA input interrupt enable bit, PAI, enables the PAIF flag to generate interrupt requests.

NOTE

The PAI input and GPT channel 3 use the same pin. To use the PAI input, disconnect it from the output logic by clearing the channel 3 output mode and output level bits, OM3 and OL3. Also clear the channel 3 output compare mask bit, OC3M3.

The PA counter register, GPTPACNT, reflects the number of pulses from the divide-by-64 clock since the last reset.

NOTE

The GPT prescaler generates the divide-by-64 clock. If the timer is not active, there is no divide-by-64 clock.
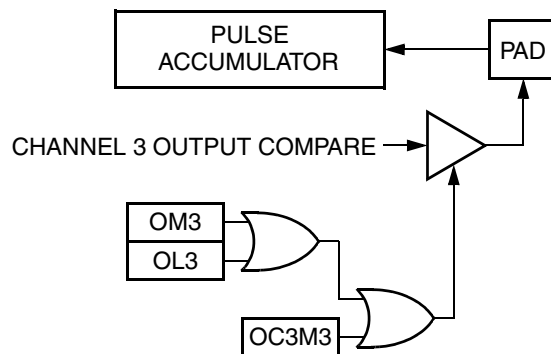
Figure 18-22. Channel 3 Output Compare/Pulse Accumulator Logic

18.7.7 General-Purpose I/O Ports

An I/O pin used by the timer defaults to general-purpose I/O unless an internal function which uses that pin is enabled.

The PORTT*n* pins can be configured for either an input capture function or an output compare function. The IOS*n* bits in the GPT IC/OC select register configure the PORTT*n* pins as either input capture or output compare pins.

The PORTT*n* data direction register controls the data direction of an input capture pin. External pin conditions trigger input captures on input capture pins configured as inputs.

To configure a pin for input capture:

1. Clear the pin's IOS bit in GPTIOS.
2. Clear the pin's DDR bit in PORTT*n*DDR.
3. Write to GPTCTL2 to select the input edge to detect.

PORTT*n*DDR does not affect the data direction of an output compare pin. The output compare function overrides the data direction register but does not affect the state of the data direction register.

To configure a pin for output compare:

1. Set the pin's IOS bit in GPTIOS.
2. Write the output compare value to GPTC*n*.
3. Clear the pin's DDR bit in PORTT*n*DDR.
4. Write to the OM*n*/OL*n* bits in GPTCTL1 to select the output action.

Table 18-23 shows how various timer settings affect pin functionality.

Table 18-23. GPT Settings and Pin Functions

GPTEN	DDR[1]	GPTIOS	EDGx [B:A]	OMx/ OLx[2]	OC3Mx[3]	Pin Data Dir.	Pin Driven by	Pin Function	Comments
0	0	X[4]	X	X	X	In	Ext.	Digital input	GPT disabled by GPTEN = 0
0	1	X	X	X	X	Out	Data reg.	Digital output	GPT disabled by GPTEN = 0

Table 18-23. GPT Settings and Pin Functions (continued)

GPTEN	DDR[1]	GPTIOS	EDGx [B:A]	OMx/ OLx[2]	OC3Mx[3]	Pin Data Dir.	Pin Driven by	Pin Function	Comments
1	0	0 (IC)	0 (IC disabled)	X	0	In	Ext.	Digital input	Input capture disabled by EDG*n* setting
1	1	0	0	X	0	Out	Data reg.	Digital output	Input capture disabled by EDG*n* setting
1	0	0	<> 0	X	0	In	Ext.	IC and digital input	Normal settings for input capture
1	1	0	<> 0	X	0	Out	Data reg.	Digital output	Input capture of data driven to output pin by CPU
1	0	0	<> 0	X	1	In	Ext.	IC and digital input	OC3M setting has no effect because IOS = 0
1	1	0	<> 0	X	1	Out	Data reg.	Digital output	OC3M setting has no effect because IOS = 0; input capture of data driven to output pin by CPU
1	0	1 (OC)	X[(3)]	0[5]	0	In	Ext.	Digital input	Output compare takes place but does not affect the pin because of the OM*n*/OL*n* setting
1	1	1	X	0	0	Out	Data reg.	Digital output	Output compare takes place but does not affect the pin because of the OM*n*/OL*n* setting
1	0	1	X	<> 0	0	Out	OC action	Output compare	Pin readable only if DDR = 0[(5)]
1	1	1	X	<> 0	0	Out	OC action	Output compare	Pin driven by OC action[(5)]
1	0	1	X	X	1	Out	OC action/ OC3D*n*	Output compare (ch 3)	Pin readable only if DDR = 0[6]
1	1	1	X	X	1	Out	OC action/ OC3D*n*	Output compare/ OC3D*n* (ch 3)	Pin driven by channel OC action and OC3D*n* via channel 3 OC[(6)]

[1] When DDR set the pin as input (0), reading the data register will return the state of the pin. When DDR set the pin as output (1), reading the data register will return the content of the data latch. Pin conditions such as rising or falling edges can trigger an input capture on a pin configured as an input.

[2] OM*n*/OL*n* bit pairs select the output action to be taken as a result of a successful output compare. When either OM*n* or OL*n* is set and the IOS*n* bit is set, the pin is an output regardless of the state of the corresponding DDR bit.

[3] Setting an OC3M bit configures the corresponding PORTT*n* pin to be output. OC3M*n* makes the PORTT*n* pin an output regardless of the data direction bit when the pin is configured for output compare (IOS*n* = 1). The OC3M*n* bits do not change the state of the PORTT*n*DDR bits.

[4] X = Don't care

[5] An output compare overrides the data direction bit of the output compare pin but does not change the state of the data direction bit. Enabling output compare disables data register drive of the pin.

6 A successful output compare on channel 3 causes an output value determined by OC3D*n* value to temporarily override the output compare pin state of any other output compare channel.The next OC action for the specific channel will still be output to the pin. A channel 3 output compare can cause bits in the output compare 3 data register to transfer to the GPT port data register, depending on the output compare 3 mask register.

18.8   Reset

Reset initializes the GPT registers to a known startup state as described in Section 18.6, "Memory Map and Registers."

18.9   Interrupts

Table 18-24 lists the interrupt requests generated by the timer.

Table 18-24. GPT Interrupt Requests

Interrupt Request	Flag	Enable Bit
Channel 3 IC/OC	C3F	C3I
Channel 2 IC/OC	C2F	C2I
Channel 1 IC/OC	C1F	C1I
Channel 0 IC/OC	C0F	C0I
PA overflow	PAOVF	PAOVI
PA input	PAIF	PAI
Timer overflow	TOF	TOI

18.9.1   GPT Channel Interrupts (C*n*F)

A channel flag is set when an input capture or output compare event occurs. Clear a channel flag by writing a 1 to it.

NOTE

When the fast flag clear all bit, GPTSCR1[TFFCA], is set, an input capture read or an output compare write clears the corresponding channel flag.

When a channel flag is set, it does not inhibit subsequent output compares or input captures

18.9.2   Pulse Accumulator Overflow (PAOVF)

PAOVF is set when the 16-bit pulse accumulator rolls over from 0xFFFF to 0x0000. If the PAOVI bit in GPTPACTL is also set, PAOVF generates an interrupt request. Clear PAOVF by writing a 1 to this flag.

NOTE

When the fast flag clear all enable bit, GPTSCR1[TFFCA], is set, any access to the pulse accumulator counter registers clears all the flags in GPTPAFLG.

18.9.3   Pulse Accumulator Input (PAIF)

PAIF is set when the selected edge is detected at the PAI pin. In event counter mode, the event edge sets PAIF. In gated time accumulation mode, the trailing edge of the gate signal at the PAI pin sets PAIF. If the PAI bit in GPTPACTL is also set, PAIF generates an interrupt request. Clear PAIF by writing a 1 to this flag.

NOTE

When the fast flag clear all enable bit, GPTSCR1[TFFCA], is set, any access to the pulse accumulator counter registers clears all the flags in GPTPAFLG.

18.9.4   Timer Overflow (TOF)

TOF is set when the GPT counter rolls over from 0xFFFF to 0x0000. If the GPTSCR2[TOI] bit is also set, TOF generates an interrupt request. Clear TOF by writing a 1 to this flag.

NOTE

When the GPT channel 3 registers contain 0xFFFF and TCRE is set, TOF does not get set even though the GPT counter registers go from 0xFFFF to 0x0000.

When the fast flag clear all bit, GPTSCR1[TFFCA], is set, any access to the GPT counter registers clears GPT flag register 2.

When TOF is set, it does not inhibit future overflow events.

Chapter 19
DMA Timers (DTIM0–DTIM3)

19.1 Introduction

This chapter describes the configuration and operation of the four direct memory access (DMA) timer modules (DTIM0, DTIM1, DTIM2, and DTIM3). These 32-bit timers provide input capture and reference compare capabilities with optional signaling of events using interrupts or DMA triggers. Additionally, programming examples are included.

NOTE

> The designation '*n*' is used throughout this section to refer to registers or signals associated with one of the four identical timer modules—DTIM0, DTIM1, DTIM2, or DTIM3.

19.1.1 Overview

Each DMA timer module has a separate register set for configuration and control. The timers can be configured to operate from the system clock or from an external clocking source using the DT*n*IN signal. If the system clock is selected, it can be divided by 16 or 1. The selected clock source is routed to an 8-bit programmable prescaler that clocks the actual DMA timer counter register (DTCN*n*). Using the DTMR*n*, DTXMR*n*, DTCR*n*, and DTRR*n* registers, the DMA timer may be configured to assert an output signal, generate an interrupt, or initiate a DMA transfer on a particular event.

NOTE

> The GPIO module must be configured to enable the peripheral function of the appropriate pins (refer to Chapter 13, "General Purpose I/O Module") prior to configuring the DMA Timers.

Figure 19-1 is a block diagram of one of the four identical timer modules.
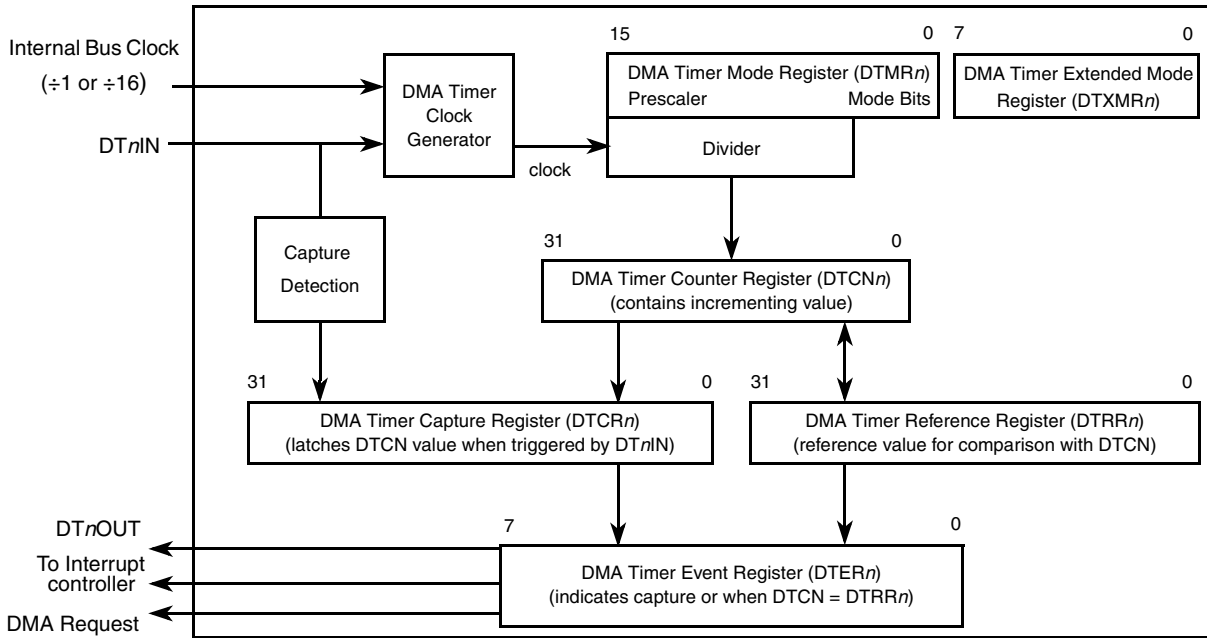
Figure 19-1. DMA Timer Block Diagram

19.1.2   Features

Each DMA timer module has the following features:

- Maximum timeout period of 266,521 seconds at 66 MHz (~74 hours)
- 15-ns resolution at 66 MHz
- Programmable sources for the clock input, including external clock
- Programmable prescaler
- Input-capture capability with programmable trigger edge on input pin
- Programmable mode for the output pin on reference compare
- Free run and restart modes
- Programmable interrupt or DMA request on input capture or reference-compare

19.2   Memory Map/Register Definition

The following features are programmable through the timer registers, shown in Table 19-1:

19.2.1   Prescaler

The prescaler clock input is selected from system clock (divided by 1 or 16) or from the corresponding timer input, DTnIN. DTnIN is synchronized to the system clock. The synchronization delay is between two and three system clocks. The corresponding DTMRn[CLK] selects the clock input source. A programmable prescaler divides the clock input by values from 1 to 256. The prescaler output is an input to the 32-bit counter, DTCNn.

19.2.2 Capture Mode

Each DMA timer has a 32-bit timer capture register (DTCR*n*) that latches the counter value when the corresponding input capture edge detector senses a defined DT*n*IN transition. The capture edge bits (DTMR*n*[CE]) select the type of transition that triggers the capture and sets the timer event register capture event bit, DTER*n*[CAP]. If DTER*n*[CAP] is set and DTXMR*n*[DMAEN] is one, a DMA request is asserted. If DTER*n*[CAP] is set and DTXMR*n*[DMAEN] is zero, an interrupt is asserted.

19.2.3 Reference Compare

Each DMA timer can be configured to count up to a reference value, at which point DTER*n*[REF] is set. If DTMR*n*[ORRI] is one and DTXMR*n*[DMAEN] is zero, an interrupt is asserted. If DTMR*n*[ORRI] is one and DTXMR*n*[DMAEN] is one, a DMA request is asserted. If the free run/restart bit DTMR*n*[FRR] is set, a new count starts. If it is clear, the timer keeps running.

19.2.4 Output Mode

When a timer reaches the reference value selected by DTRR, it can send an output signal on DT*n*OUT. DT*n*OUT can be an active-low pulse or a toggle of the current output, as selected by the DTMR*n*[OM] bit.

19.2.5 Memory Map

The timer module registers, shown in Table 19-1, can be modified at any time.

Table 19-1. DMA Timer Module Memory Map

IPSBAR Offset DMA Timer 0 DMA Timer 1 DMA Timer 2 DMA Timer 3	Register	Access	Reset Value	Section/Page
0x00_0400 0x00_0440 0x00_0480 0x00_04C0	DMA Timer *n* Mode Register (DTMR*n*)	R/W	0x0000	19.2.6/19-4
0x00_0402 0x00_0442 0x00_0482 0x00_04C2	DMA Timer *n* Extended Mode Register (DTXMR*n*)	R/W	0x0000	19.2.7/19-5
0x00_0403 0x00_0443 0x00_0483 0x00_04C3	DMA Timer *n* Event Register (DTER*n*)	R/W	0x0000	19.2.8/19-6
0x00_0404 0x00_0444 0x00_0484 0x00_04C4	DMA Timer *n* Reference Register (DTRR*n*)	R/W	0x1111_1111	19.2.9/19-7

Table 19-1. DMA Timer Module Memory Map (continued)

IPSBAR Offset DMA Timer 0 DMA Timer 1 DMA Timer 2 DMA Timer 3	Register	Access	Reset Value	Section/Page
0x00_0408 0x00_0448 0x00_0488 0x00_04C8	DMA Timer *n* Capture Register (DTCR*n*)	R/W	0x0000_0000	19.2.10/19-8
0x00_040C 0x00_044C 0x00_048C 0x00_04CC	DMA Timer *n* Counter Register (DTCN*n*)	R	0x0000_0000	19.2.11/19-8

19.2.6 DMA Timer Mode Registers (DTMR*n*)

DTMRs, shown in Figure 19-2, program the prescaler and various timer modes.

IPSBAR 0x00_0400 (DTMR0)   Access: User read/write
Offset: 0x00_0440 (DTMR1)
        0x00_0480 (DTMR2)
        0x00_04C0 (DTMR3)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W					PS					CE	OM	ORRI	FRR		CLK	RST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 19-2. DMA Timer Mode Registers (DTMR*n*)

Table 19-2. DTMR*n* Field Descriptions

Field	Description
15–8 PS	Prescaler value. The prescaler is programmed to divide the clock input (system clock/(16 or 1) or clock on DT*n*IN) by values from 1 (PS = 0x00) to 256 (PS = 0xFF).
7–6 CE	Capture edge. 00 Disable capture event output 01 Capture on rising edge only 10 Capture on falling edge only 11 Capture on any edge
5 OM	Output mode. 0 Active-low pulse for one system clock cycle (-ns resolution at 3 MHz). 1 Toggle output.
4 ORRI	Output reference request, interrupt enable. If ORRI is set when DTER*n*[REF] = 1, a DMA request or an interrupt occurs, depending on the value of DTXMR*n*[DMAEN] (DMA request if =1, interrupt if =0). 0 Disable DMA request or interrupt for reference reached (does not affect DMA request or interrupt on capture function). 1 Enable DMA request or interrupt upon reaching the reference value.

Table 19-2. DTMR*n* Field Descriptions (continued)

Field	Description
3 FRR	Free run/restart 0 Free run. Timer count continues to increment after reaching the reference value. 1 Restart. Timer count is reset immediately after reaching the reference value.
2–1 CLK	Input clock source for the timer 00 Stop count 01 System clock divided by 1 10 System clock divided by 16. Note that this clock source is not synchronized with the timer; thus successive time-outs may vary slightly. 11 DT*n*IN pin (falling edge)
0 RST	Reset timer. Performs a software timer reset similar to an external reset, although other register values can still be written while RST = 0. A transition of RST from 1 to 0 resets register values. The timer counter is not clocked unless the timer is enabled. 0 Reset timer (software reset) 1 Enable timer

19.2.7   DMA Timer Extended Mode Registers (DTXMR*n*)

The DTXMR*n* register programs DMA request and increment modes for the timers.

IPSBAR 0x00_0402 (DTXMR0)   Access: User read/write
Offset: 0x00_0442 (DTXMR1)
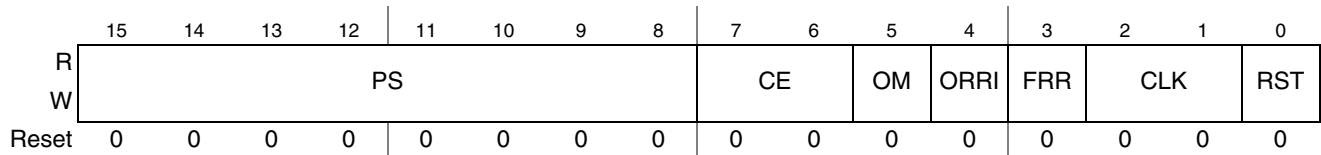   0x00_0482 (DTXMR2)
         0x00_04C2 (DTXMR3)

	7	6	5	4	3	2	1	0
R	DMAEN	0	0	0	0	0	0	MODE16
W								
Reset:	0	0	0	0	0	0	0	0
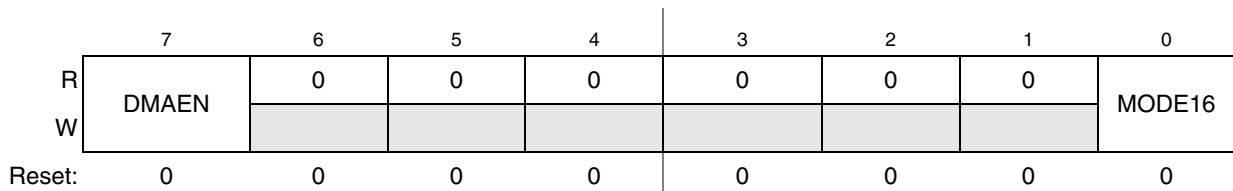
Figure 19-3. DMA Timer Extended Mode Registers (DTXMR*n*)

Table 19-3. DTXMR*n* Field Descriptions

Field	Description
7 DMAEN	DMA request. Enables DMA request output on counter reference match or capture edge event. 0 DMA request disabled 1 DMA request enabled
6–1	Reserved, should be cleared.
0 MODE16	Selects the increment mode for the timer. MODE16 = 1 is intended to exercise the upper bits of the 32-bit timer in diagnostic software without requiring the timer to count through its entire dynamic range. When set, the counter's upper 16 bits mirror its lower 16 bits. All 32 bits of the counter are still compared to the reference value. 0 Increment timer by 1 1 Increment timer by 65,537

19.2.8   DMA Timer Event Registers (DTER*n*)

DTER*n*, shown in Figure 19-4, reports capture or reference events by setting DTER*n*[CAP] or
DTER*n*[REF]. This reporting is done regardless of the corresponding DMA request or interrupt enable
values, DTXMR*n*[DMAEN] and DTMR*n*[ORRI,CE].

Writing a 1 to either DTER*n*[REF] or DTER*n*[CAP] clears it (writing a 0 does not affect bit value); both
bits can be cleared at the same time. If configured to generate an interrupt request, the REF and CAP bits
should be cleared early in the interrupt service routine so the timer module can negate the interrupt request
signal to the interrupt controller. If configured to generate a DMA request, the processing of the DMA data
transfer automatically clears both the REF and CAP flags via the internal DMA ACK signal.

IPSBAR 0x00_0403 (DTER0)   Access: User read/write
Offset: 0x00_0443 (DTER1)
         0x00_0483 (DTER2)
         0x00_04C3 (DTER3)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	REF	CAP
W							w1c	w1c
Reset:	0	0	0	0	0	0	0	0
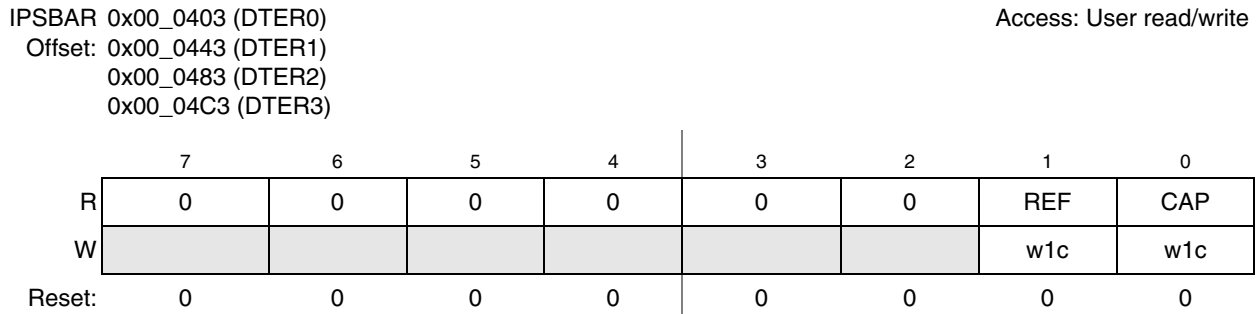
Figure 19-4. DMA Timer Event Registers (DTER*n*)

Table 19-4. DTER*n* Field Descriptions

Field	Description
7–2	Reserved, should be cleared.

Table 19-4. DTER*n* Field Descriptions (continued)

Field	Description
1 REF	Output reference event. The counter value, DTCN*n,* equals the reference value, DTRR*n*. Writing a one to REF clears the event condition. Writing a zero has no effect. table1
0 CAP	Capture event. The counter value has been latched into DTCR*n*. Writing a one to CAP clears the event condition. Writing a zero has no effect. table2

table1:

REF	DTMR*n*[ORRI]	DTXMR*n*[DMAEN]	
0	X	X	No event
1	0	0	No request asserted
1	0	1	No request asserted
1	1	0	Interrupt request asserted
1	1	1	DMA request asserted

table2:

CAP	DTMR*n*[CE]	DTXMR*n* [DMAEN]	
0	XX	X	No event
1	00	0	Disable capture event output
1	00	1	Disable capture event output
1	01	0	Capture on rising edge & trigger interrupt
1	01	1	Capture on rising edge & trigger DMA
1	10	0	Capture on falling edge & trigger interrupt
1	10	1	Capture on falling edge & trigger DMA
1	11	0	Capture on any edge & trigger interrupt
1	11	1	Capture on any edge & trigger DMA

19.2.9 DMA Timer Reference Registers (DTRR*n*)

Each DTRR*n*, shown in Figure 19-5, contains the reference value compared with the respective free-running timer counter (DTCN*n*) as part of the output-compare function. The reference value is not matched until DTCN*n* equals DTRR*n*, and the prescaler indicates that DTCN*n* should be incremented again. Thus, the reference register is matched after DTRR*n*+1 time intervals.

IPSBAR 0x00_0404 (DTRR0)   Access: User read/write
Offset: 0x00_0444 (DTRR1)
0x00_0484 (DTRR2)
0x00_04C4 (DTRR3)

	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				REF				
W								
Reset	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1
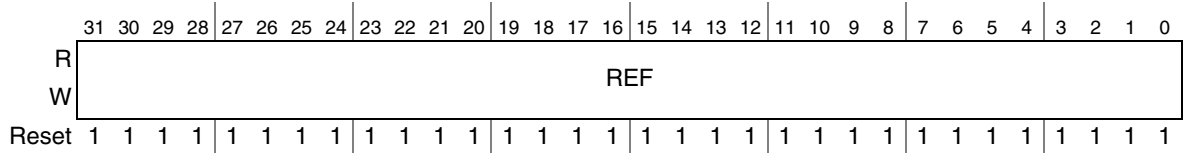
Figure 19-5. DMA Timer Reference Registers (DTRR*n*)

19.2.10 DMA Timer Capture Registers (DTCR*n*)

Each DTCR*n* latches the corresponding DTCN*n* value during a capture operation when an edge occurs on DT*n*IN, as programmed in DTMR*n*. The system clock is assumed to be the clock source. DT*n*IN cannot simultaneously function as a clocking source and as an input capture pin. Indeterminate operation will result if DT*n*IN is set as the clock source when the input capture mode is used.

IPSBAR 0x00_0408 (DTCR0)   Access: User read-only
Offset: 0x00_0448 (DTCR1)
0x00_0488 (DTCR2)
0x00_04C8 (DTCR3)

	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				CAP (32-bit capture counter value)				
W								
Reset	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
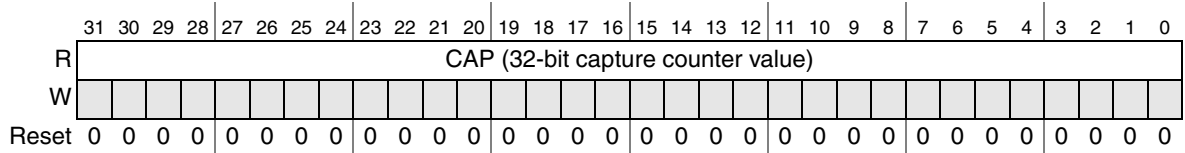
Figure 19-6. DMA Timer Capture Registers (DTCR*n*)

19.2.11 DMA Timer Counters (DTCN*n*)

The current value of the 32-bit DTCNs can be read at anytime without affecting counting. Writing to DTCN*n* clears it. The timer counter increments on the clock source rising edge (system clock ÷ 1, system clock ÷ 16, or DT*n*IN).

IPSBAR 0x00_040C (DTCN0)   Access: User read/write
Offset: 0x00_044C (DTCN1)
0x00_048C (DTCN2)
0x00_04CC (DTCN3)

	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				CNT (32-bit timer counter value count)				
W								
Reset	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
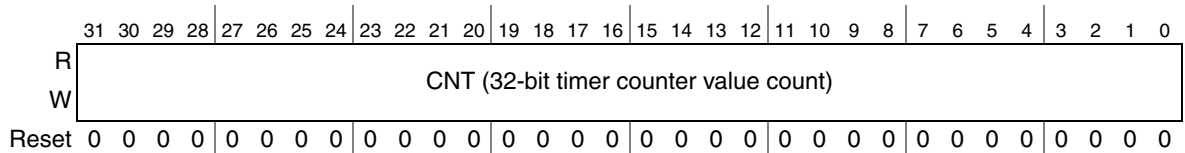
Figure 19-7. DMA Timer Counters (DTCN*n*)

19.3   Initialization/Application Information

The general-purpose timer modules are typically used in the following manner, though this is not necessarily the program order in which these actions must occur:

- The DTMR*n* and DTXMR*n* registers are configured for the desired function and behavior.
  - — Count and compare to a reference value stored in the DTRR*n* register
  - — Capture the timer value on an edge detected on DT*n*IN
  - — Configure DT*n*OUT output mode
  - — Increment counter by 1 or by 65,537 (16-bit mode)
  - — Enable/disable interrupt or DMA request on counter reference match or capture edge
- The DTMR*n*[CLK] register is configured to select the clock source to be routed to the prescaler.
  - — System clock (can be divided by 1 or 16)
  - — DT*n*IN, the maximum value of DT*n*IN is 1/5 of the internal bus clock, as described in the device's electrical characteristics

NOTE

DT*n*IN may not be configured as a clock source when the timer capture mode is selected or indeterminate operation will result.

- The 8-bit DTMR*n*[PS] prescaler value is set.
- Using DTMR*n*[RST] the counter is cleared and started.
- Timer events are either handled with an interrupt service routine, a DMA request, or by a software polling mechanism.

19.3.1   Code Example

The following code provides an example of how to initialize DMA Timer0 and how to use the timer for counting time-out periods.

```
DTMR0 EQU IPSBARx+0x400 ;Timer0 mode register
DTMR1 EQU IPSBARx+0x440 ;Timer1 mode register
DTRR0 EQU IPSBARx+0x404 ;Timer0 reference register
DTRR1 EQU IPSBARx+0x444 ;Timer1 reference register
DTCR0 EQU IPSBARx+0x408 ;Timer0 capture register
DTCR1 EQU IPSBARx+0x448 ;Timer1 capture register
DTCN0 EQU IPSBARx+0x40C ;Timer0 counter register
DTCN1 EQU IPSBARx+0x44C ;Timer1 counter register
DTER0 EQU IPSBARx+0x403 ;Timer0 event register
DTER1 EQU IPSBARx+0x443 ;Timer1 event register

* TMR0 is defined as: *
*[PS] = 0xFF,    divide clock by 256
*[CE] = 00       disable capture event output
*[OM] = 0        output=active-low pulse
*[ORRI] = 0,     disable ref. match output
*[FRR] = 1,      restart mode enabled
*[CLK] = 10,     system clock/16
*[RST] = 0,      timer0 disabled
```

```
        move.w #0xFF0C,D0
        move.w D0,TMR0
        move.l #0x0000,D0;writing to the timer counter with any
        move.l DO,TCN0 ;value resets it to zero
        move.l #0xAFAF,DO ;set the timer0 reference to be
        move.l #D0,TRR0 ;defined as 0xAFAF
```

The simple example below uses Timer0 to count time-out loops. A time-out occurs when the reference value, 0xAFAF, is reached.

```
timer0_ex
        clr.l DO
        clr.l D1
        clr.l D2
        move.l #0x0000,D0
        move.l D0,TCN0                ;reset the counter to 0x0000
        move.b #0x03,D0               ;writing ones to TER0[REF,CAP]
        move.b D0,TER0               ;clears the event flags
        move.w TMR0,D0               ;save the contents of TMR0 while setting
        bset #0,D0                   ;the 0 bit. This enables timer 0 and starts counting
        move.w D0,TMR0               ;load the value back into the register, setting TMR0[RST]
T0_LOOP
        move.b TER0,D1               ;load TER0 and see if
        btst #1,D1                   ;TER0[REF] has been set
        beq T0_LOOP
        addi.l #1,D2                 ;Increment D2
        cmp.l #5,D2                  ;Did D2 reach 5? (i.e. timer ref has timed)
        beq T0_FINISH                ;If so, end timer0 example. Otherwise jump back.
        move.b #0x02,D0              ;writing one to TER0[REF] clears the event flag
        move.b D0,TER0
        jmp T0_LOOP
T0_FINISH
        HALT                         ;End processing. Example is finished
```

19.3.2 Calculating Time-Out Values

The formula below determines time-out periods for various reference values:

$$\text{Timeout period} = (1/\text{clock frequency}) \times (1 \text{ or } 16) \times (\text{DTMR}n[\text{PS}] + 1) \times (\text{DTRR}n[\text{REF}] + 1) \qquad \textbf{\textit{Eqn. 19-1}}$$

When calculating time-out periods, add 1 to the prescaler to simplify calculating, because DTMRn[PS] = 0x00 yields a prescaler of 1 and DTMRn[PS] = 0xFF yields a prescaler of 256.

For example, if a 66-MHz timer clock is divided by 16, DTMRn[PS] = 0x7F, and the timer is referenced at 0xFBC5 (64453 decimal), the time-out period is as follows:

$$\text{Timeout period} = \frac{1}{66 \times 10^6} \times 16 \times (127 + 1) \times (64453 + 1) = 2.00s \qquad \textbf{\textit{Eqn. 19-2}}$$

Chapter 20
Queued Serial Peripheral Interface (QSPI)

20.1 Introduction

This chapter describes the queued serial peripheral interface (QSPI) module. Following a feature set overview is a description of operation including details of the QSPI's internal RAM organization. The chapter concludes with the programming model and a timing diagram.

20.1.1 Block Diagram

Figure 20-1 illustrates the QSPI module.
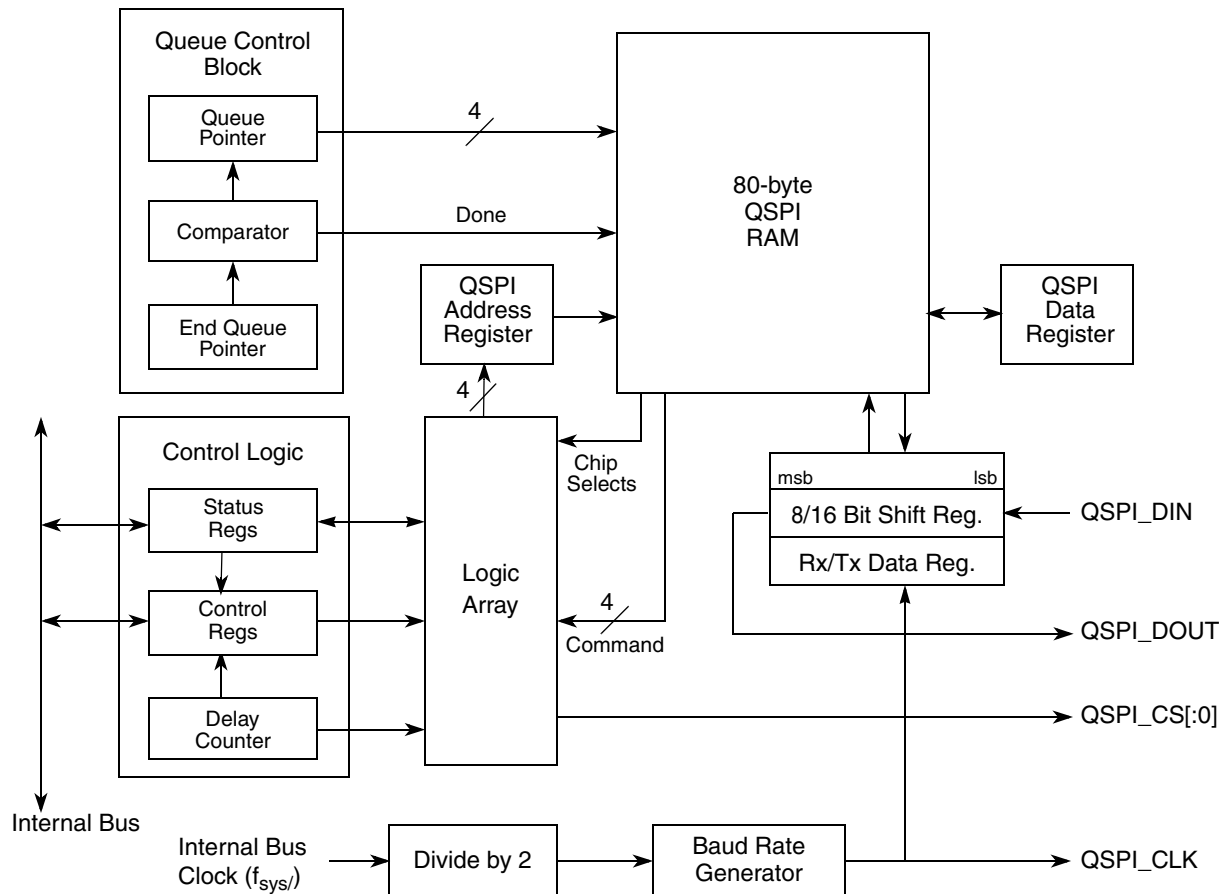
Figure 20-1. QSPI Block Diagram

20.1.2   Overview

The queued serial peripheral interface module provides a serial peripheral interface with queued transfer capability. It allows users to queue up to 16 transfers at once, eliminating CPU intervention between transfers. Transfer RAM in the QSPI is indirectly accessible using address and data registers.

NOTE

The GPIO module must be configured to enable the peripheral function of the appropriate pins (refer to Chapter 13, "General Purpose I/O Module") prior to configuring the QSPI Module.

20.1.3   Features

Features include the following:

- Programmable queue to support up to 16 transfers without user intervention
- Supports transfer sizes of 8 to 16 bits in 1-bit increments
- Four peripheral chip-select lines for control of up to 15 devices
- Baud rates from 129.4 Kbps to 16.6 Mbps at 66 MHz internal bus frequency
- Programmable delays before and after transfers
- Programmable QSPI clock phase and polarity
- Supports wraparound mode for continuous transfers

20.1.4   External Signals Description

The module provides access to as many as 15 devices with a total of seven signals: QSPI_DOUT, QSPI_DIN, QSPI_CLK, QSPI_CS0, QSPI_CS1, QSPI_CS2, and QSPI_CS3.

Peripheral chip-select signals, QSPI_CS*n*, are used to select an external device as the source or destination for serial data transfer. Signals are asserted whenever a command in the queue is executed. More than one chip-select signal can be asserted simultaneously.

Although QSPI_CS*n* will function as simple chip selects in most applications, up to 15 devices can be selected by decoding them with an external 4-to-16 decoder.

Table 20-1. QSPI Input and Output Signals and Functions

Signal Name	Hi-Z or Actively Driven	Function
QSPI Data Output (QSPI_DOUT)	Configurable	Serial data output from QSPI
QSPI Data Input (QSPI_DIN)	N/A	Serial data input to QSPI
Serial Clock (QSPI_CLK)	Actively driven	Clock output from QSPI
Peripheral Chip Selects (QSPI_CS*n*)	Actively driven	Peripheral selects

20.1.5   Modes of Operation

Because the QSPI module only operates in master mode, the master bit in the QSPI mode register, QMR[MSTR], must be set for the QSPI to function properly. The QSPI can initiate serial transfers but cannot respond to transfers initiated by other QSPI masters.

20.2   Memory Map/Register Definition

Table 20-2 is the QSPI register memory map. Reading reserved locations returns zeros.

Table 20-2. QSPI Memory Map

IPSBAR Offset[1]	Register	Access	Reset Value	Section/Page
0x00_0340	QSPI Mode Register (QMR)	R/W	0x0104	20.2.1/20-3
0x00_0344	QSPI Delay Register (QDLYR)	R/W	0x0404	20.2.2/20-5
0x00_0348	QSPI Wrap Register (QWR)	R/W[2]	0x0000	20.2.3/20-6
0x00_034C	QSPI Interrupt Register (QIR)	R/W[2]	0x0000	20.2.4/20-6
0x00_0350	QSPI Address Register (QAR)	R/W[2]	0x0000	20.2.5/20-8
0x00_0354	QSPI Data Register (QDR)	R/W	0x0000	20.2.6/20-8

[1]   Addresses not assigned to a register and undefined register bits are reserved for expansion.

[2]   See the register description for special cases. Some bits may be read- or write-only.

20.2.1   QSPI Mode Register (QMR)

The QMR, shown in Figure 20-2, determines the basic operating modes of the QSPI module. Parameters such as QSPI_CLK polarity and phase, baud rate, master mode operation, and transfer size are determined by this register. The data output high impedance enable, DOHIE, controls the operation of QSPI_DOUT between data transfers. When DOHIE is cleared, QSPI_DOUT is actively driven between transfers. When DOHIE is set, QSPI_DOUT assumes a high impedance state.

NOTE

Because the QSPI does not operate in slave mode, the master mode enable bit, QMR[MSTR], must be set for the QSPI module to operate correctly.

IPSBAR 0x00_0340   Access: User read/write
Offset:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MSTR	DOHIE			BITS		CPOL	CPHA				BAUD				
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
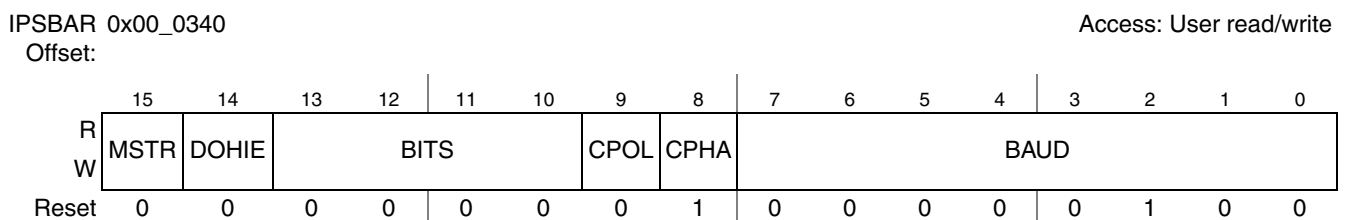
Figure 20-2. QSPI Mode Register (QMR)

Table 20-3. QMR Field Descriptions

Field	Description
15 MSTR	Master mode enable. 0 Reserved, do not use. 1 The QSPI is in master mode. Must be set for the QSPI module to operate correctly.
14 DOHIE	Data output high impedance enable. Selects QSPI_DOUT mode of operation. 0 Default value after reset. QSPI_DOUT is actively driven between transfers. 1 QSPI_DOUT is high impedance between transfers.
13–10 BITS	Transfer size. Determines the number of bits to be transferred for each entry in the queue. Bits Table: <table><tr><th>BITS</th><th>Bits per Transfer</th></tr><tr><td>0000</td><td>16</td></tr><tr><td>0001–0111</td><td>Reserved</td></tr><tr><td>1000</td><td>8</td></tr><tr><td>1001</td><td>9</td></tr><tr><td>1010</td><td>10</td></tr><tr><td>1011</td><td>11</td></tr><tr><td>1100</td><td>12</td></tr><tr><td>1101</td><td>13</td></tr><tr><td>1110</td><td>14</td></tr><tr><td>1111</td><td>15</td></tr></table>
9 CPOL	Clock polarity. Defines the clock polarity of QSPI_CLK. 0 The inactive state value of QSPI_CLK is logic level 0. 1 The inactive state value of QSPI_CLK is logic level 1.
8 CPHA	Clock phase. Defines the QSPI_CLK clock-phase. 0 Data captured on the leading edge of QSPI_CLK and changed on the following edge of QSPI_CLK. 1 Data changed on the leading edge of QSPI_CLK and captured on the following edge of QSPI_CLK.
7–0 BAUD	Baud rate divider. The baud rate is selected by writing a value in the range 2–255. A value of zero disables the QSPI. A value of 1 is an invalid setting. The desired QSPI_CLK baud rate is related to the internal bus clock and QMR[BAUD] by the following expression: $$QMR[BAUD] = f_{sys} / (2 \times [\text{desired QSPI\_CLK baud rate}])$$

Figure 20-3 shows an example of a QSPI clocking and data transfer.
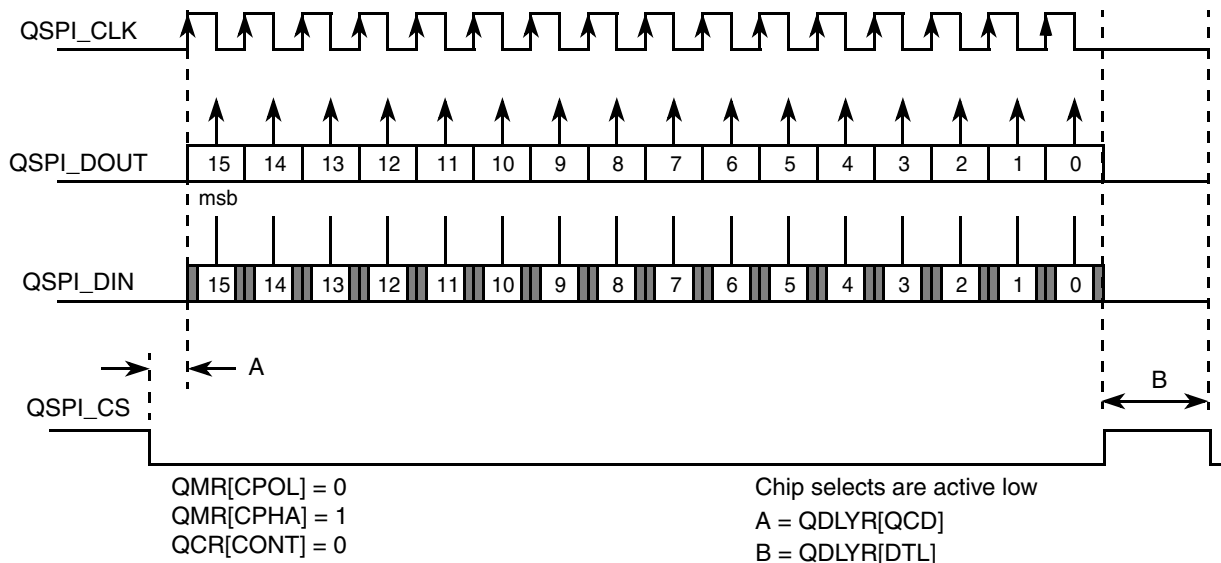
Figure 20-3. QSPI Clocking and Data Transfer Example

20.2.2   QSPI Delay Register (QDLYR)

IPSBAR 0x00_0344   Access: User read/write
Offset:

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

Figure 20-4. QSPI Delay Register (QDLYR)

Table 20-4. QDLYR Field Descriptions

Field	Description
15 SPE	QSPI enable. When set, the QSPI initiates transfers in master mode by executing commands in the command RAM. Automatically cleared by the QSPI when a transfer completes. The user can also clear this bit to abort transfer unless QIR[ABRTL] is set. The recommended method for aborting transfers is to set QWR[HALT].
14–8 QCD	QSPI_CLK delay. When the DSCK bit in the command RAM is set this field determines the length of the delay from assertion of the chip selects to valid QSPI_CLK transition. See Section 20.3.3, "Transfer Delays" for information on setting this bit field.
7–0 DTL	Delay after transfer. When the DT bit in the command RAM is set this field determines the length of delay after the serial transfer.

20.2.3   QSPI Wrap Register (QWR)

IPSBAR 0x00_0348   Access: User read/write
Offset:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HALT	WREN	WRTO	CSIV		ENDQP					CPTQP				NEWQP	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
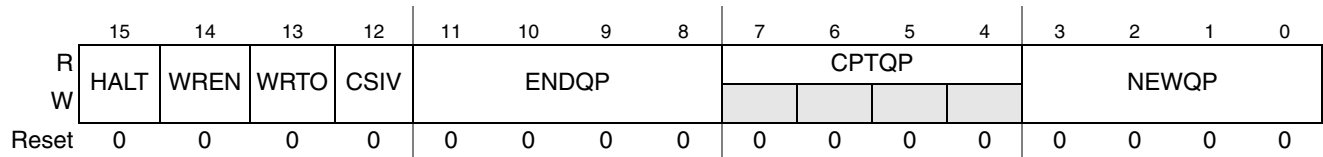
Figure 20-5. QSPI Wrap Register (QWR)

Table 20-5. QWR Field Descriptions

Field	Description
15 HALT	Halt transfers. Assertion of this bit causes the QSPI to stop execution of commands once it has completed execution of the current command.
14 WREN	Wraparound enable. Enables wraparound mode. 0 Execution stops after executing the command pointed to by QWR[ENDQP]. 1 After executing command pointed to by QWR[ENDQP], wrap back to entry zero, or the entry pointed to by QWR[NEWQP] and continue execution.
13 WRTO	Wraparound location. Determines where the QSPI wraps to in wraparound mode. 0 Wrap to RAM entry zero. 1 Wrap to RAM entry pointed to by QWR[NEWQP].
12 CSIV	QSPI_CS inactive level. 0 QSPI chip select outputs return to zero when not driven from the value in the current command RAM entry during a transfer (that is, inactive state is 0, chip selects are active high). 1 QSPI chip select outputs return to one when not driven from the value in the current command RAM entry during a transfer (that is, inactive state is 1, chip selects are active low).
11–8 ENDQP	End of queue pointer. Points to the RAM entry that contains the last transfer description in the queue.
7–4 CPTQP	Completed queue entry pointer. Points to the RAM entry that contains the last command to have been completed. This field is read only.
3–0 NEWQP	Start of queue pointer. This 4-bit field points to the first entry in the RAM to be executed on initiating a transfer.

20.2.4   QSPI Interrupt Register (QIR)

IPSBAR 0x00_034C   Access: User read/write
Offset:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WCEFB	ABRTB	0	ABRTL	WCEFE	ABRTE	0	SPIFE	0	0	0	0	WCEF	ABRT	0	SPIF
W													w1c	w1c		w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
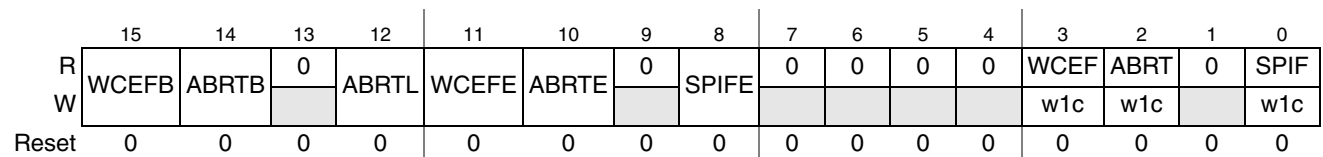
Figure 20-6. QSPI Interrupt Register (QIR)

Preliminary

Table 20-6. QIR Field Descriptions

Field	Description
15 WCEFB	Write collision access error enable. A write collision occurs during a data transfer when the RAM entry containing the current command is written to by the CPU with the QDR. When this bit is asserted, the write access to QDR results in an access error.
14 ABRTB	Abort access error enable. An abort occurs when QDLYR[SPE] is cleared during a transfer. When set, an attempt to clear QDLYR[SPE] during a transfer results in an access error.
13	Reserved, should be cleared.
12 ABRTL	Abort lock-out. When set, QDLYR[SPE] cannot be cleared by writing to the QDLYR. QDLYR[SPE] is only cleared by the QSPI when a transfer completes.
11 WCEFE	Write collision (WCEF) interrupt enable. 0 Write collision interrupt disabled 1 Write collision interrupt enabled
10 ABRTE	Abort (ABRT) interrupt enable. 0 Abort interrupt disabled 1 Abort interrupt enabled
9	Reserved, should be cleared.
8 SPIFE	QSPI finished (SPIF) interrupt enable. 0 SPIF interrupt disabled 1 SPIF interrupt enabled
7–4	Reserved, should be cleared.
3 WCEF	Write collision error flag. Indicates that an attempt has been made to write to the RAM entry that is currently being executed. Writing a 1 to this bit clears it and writing 0 has no effect.
2 ABRT	Abort flag. Indicates that QDLYR[SPE] has been cleared by the user writing to the QDLYR rather than by the QSPI completing the command queue. Writing a 1 to this bit clears it and writing 0 has no effect.
1	Reserved, should be cleared.
0 SPIF	QSPI finished flag. Asserted when the QSPI has completed all the commands in the queue. Set on completion of the command pointed to by QWR[ENDQP], and on completion of the current command after assertion of QWR[HALT]. In wraparound mode, this bit is set every time the command pointed to by QWR[ENDQP] is completed. Writing a 1 to this bit clears it and writing 0 has no effect.

The command and data RAM in the QSPI are indirectly accessible with QDR and QAR as 48 separate locations that comprise 16 words of transmit data, 16 words of receive data, and 16 bytes of commands.

A write to QDR causes data to be written to the RAM entry specified by QAR[ADDR]. This also causes the value in QAR to increment.

Correspondingly, a read at QDR returns the data in the RAM at the address specified by QAR[ADDR]. This also causes QAR to increment. A read access requires a single wait state.

NOTE

The QAR does not wrap after the last queue entry within each section of the RAM. The application software must handle address range errors.

MCF5213 Reference Manual, Rev. 1.1

20.2.5   QSPI Address Register (QAR)

The QAR is used to specify the location in the QSPI RAM that read and write operations affect. As shown in Section 20.3.1, "QSPI RAM", the transmit RAM is located at addresses 0x0 to 0xF, the receive RAM is located at 0x10 to 0x1F, and the command RAM is located at 0x20 to 0x2F.

IPSBAR 0x0x00_0350   Access: User read/write
Offset:

	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				ADDR		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
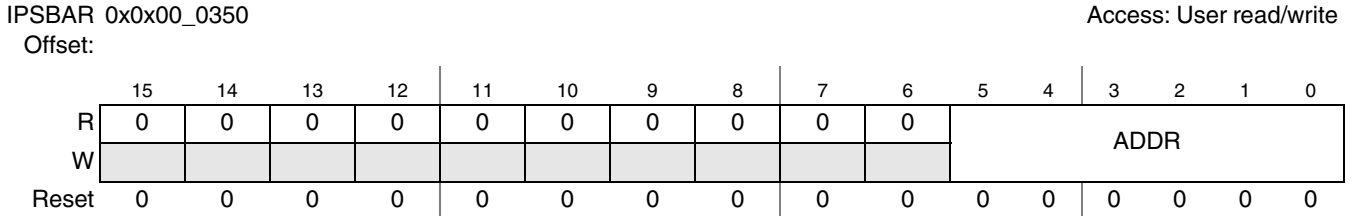
Figure 20-7. QSPI Address Register (QAR)

20.2.6   QSPI Data Register (QDR)

The QDR, shown in Figure 20-8, is used to access QSPI RAM indirectly. The CPU reads and writes all data from and to the QSPI RAM through this register.

IPSBAR 0x0x00_0354   Access: User read/write
Offset:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R								DATA								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
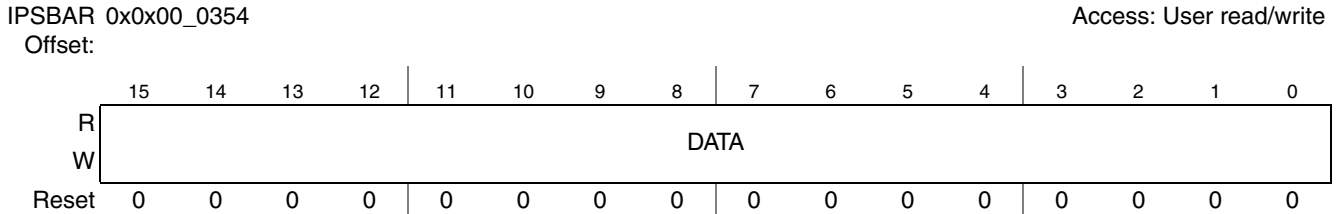
Figure 20-8. QSPI Data Register (QDR)

20.2.7   Command RAM Registers (QCR0–QCR15)

The QSPI cannot modify information in command RAM. However, it can access the command RAM by using the upper byte of the QDR.

There are 16 bytes in the command RAM. Each byte is divided into two fields. The chip select field enables external peripherals for transfer. The command field provides transfer operations.

NOTE
The command RAM is accessed only using the most significant byte of QDR and indirect addressing based on QAR[ADDR].

Figure 20-9 shows the command RAM register.

Address: QAR[ADDR]   Access: CPU write-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	CONT	BITSE	DT	DSCK		QSPI_CS			0	0	0	0	0	0	0	0
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
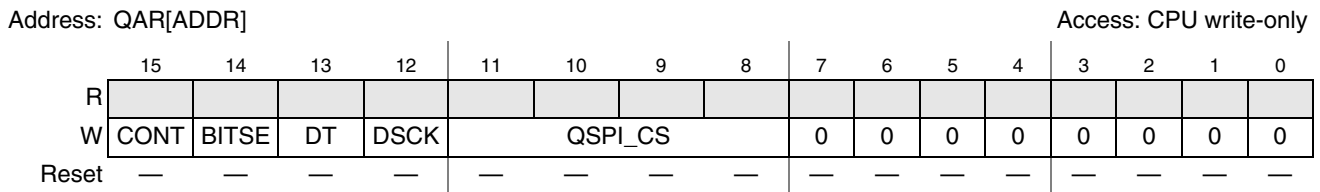
Figure 20-9. Command RAM Registers (QCR0–QCR15)

Table 20-7. QCR0–QCR15 Field Descriptions

Field	Description
15 CONT	Continuous. 0 Chip selects return to inactive level defined by QWR[CSIV] when a single word transfer is complete. 1 Chip selects return to inactive level defined by QWR[CSIV] only after the transfer of the queue entries (max of 16 words). **Note:** In order to keep the chip selects asserted beyond 16 words, the QWR[CSIV] bit must be set to control the level that the chip selects return to after the first transfer.
14 BITSE	Bits per transfer enable. 0 Eight bits 1 Number of bits set in QMR[BITS]
13 DT	Delay after transfer enable. 0 Default reset value. 1 The QSPI provides a variable delay at the end of serial transfer to facilitate interfacing with peripherals that have a latency requirement. The delay between transfers is determined by QDLYR[DTL].
12 DSCK	Chip select to QSPI_CLK delay enable. 0 Chip select valid to QSPI_CLK transition is one-half QSPI_CLK period. 1 QDLYR[QCD] specifies the delay from QSPI_CS valid to QSPI_CLK.
–8 QSPI_CS	Peripheral chip selects. Used to select an external device for serial data transfer. More than one chip select may be active at once, and more than one device can be connected to each chip select. Each bit maps directly to the corresponding QSPI_CS*n* pin. If more than four chip selects are needed, then an external demultiplexor can be used with the QSPI_CS*n* pins.
7–0	Reserved, should be cleared.

Preliminary
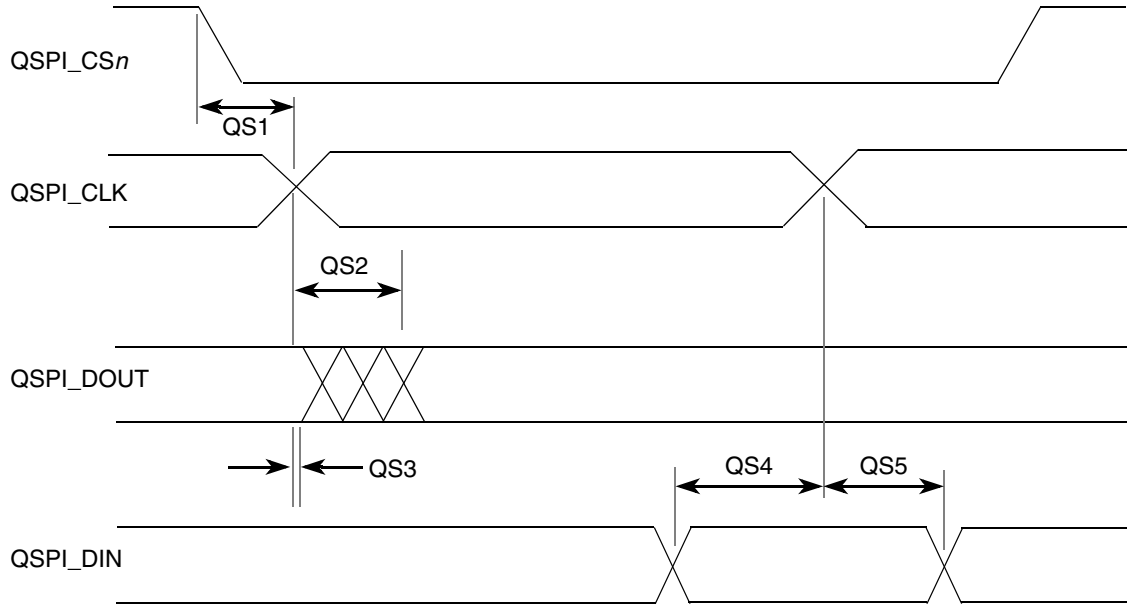
	Min	Max
QS1: QSPI_CS to QSPI_CLK	1T1	
QS2: QSPI_CLK to QSPI_DOUT VALID		20 ns
QS3: QSPI_CLK to QSPI_DOUT HOLD	0 ns	
QS4: QSPI_DIN to QSPI_CLK SETUP	10 ns	
QS5: QSPI_DIN to QSPI_CLK HOLD	10 ns	

1T1 is defined as the clock period in ns.

Figure 20-10. QSPI Timing

20.3   Functional Description

The QSPI uses a dedicated 80-Byte block of static RAM accessible both to the module and the CPU to perform queued operations. The RAM is divided into three segments as follows:

- 16 command control bytes (command RAM)
- 32 transmit data bytes (transfer RAM)
- 32 receive data bytes (transfer RAM)

The RAM is organized so that 1 byte of command control data, 1 word of transmit data, and 1 word of receive data comprise 1 of the 16 queue entries (0x0–0xF).

NOTE

Throughout ColdFire documentation, 'word' is used to designate a 16-bit data unit. The only exceptions to this appear in discussions of serial communication modules such as QSPI that support variable-length data units. To simplify these discussions, the functional unit is referred to as a 'word' regardless of length.

The user initiates QSPI operation by loading a queue of commands in command RAM, writing transmit data into transmit RAM, and then enabling the QSPI data transfer. The QSPI executes the queued commands and sets the completion flag in the QSPI interrupt register (QIR[SPIF]) to signal their completion. As another option, QIR[SPIFE] can be enabled to generate an interrupt.

The QSPI uses four queue pointers. The user can access three of them through fields in QSPI wrap register (QWR):

- The new queue pointer, QWR[NEWQP], points to the first command in the queue.
- An internal queue pointer points to the command currently being executed.
- The completed queue pointer, QWR[CPTQP], points to the last command executed.
- The end queue pointer, QWR[ENDQP], points to the final command in the queue.

The internal pointer is initialized to the same value as QWR[NEWQP]. During normal operation, the following sequence repeats:

1. The command pointed to by the internal pointer is executed.
2. The value in the internal pointer is copied into QWR[CPTQP].
3. The internal pointer is incremented.

Execution continues at the internal pointer address unless the QWR[NEWQP] value is changed. After each command is executed, QWR[ENDQP] and QWR[CPTQP] are compared. When a match occurs, QIR[SPIF] is set and the QSPI stops unless wraparound mode is enabled. Setting QWR[WREN] enables wraparound mode.

QWR[NEWQP] is cleared at reset. When the QSPI is enabled, execution begins at address 0x0 unless another value has been written into QWR[NEWQP]. QWR[ENDQP] is cleared at reset but is changed to show the last queue entry before the QSPI is enabled. QWR[NEWQP] and QWR[ENDQP] can be written at any time. When the QWR[NEWQP] value changes, the internal pointer value also changes unless a transfer is in progress, in which case the transfer completes normally. Leaving QWR[NEWQP] and QWR[ENDQP] set to 0x0 causes a single transfer to occur when the QSPI is enabled.

Data is transferred relative to QSPI_CLK which can be generated in any one of four combinations of phase and polarity using QMR[CPHA,CPOL]. Data is transferred with the most significant bit (msb) first. The number of bits transferred defaults to 8, but can be set to any value between 8 and 16 by writing a value into the BITSE field of the command RAM (QCR[BITSE]).

20.3.1   QSPI RAM

The QSPI contains an 80-Byte block of static RAM that can be accessed by both the user and the QSPI. This RAM does not appear in the device memory map because it can only be accessed by the user indirectly through the QSPI address register (QAR) and the QSPI data register (QDR). The RAM is divided into three segments with 16 addresses each:

- Receive data RAM, the initial destination for all incoming data
- Transmit data RAM, a buffer for all out-bound data
- Command RAM, where commands are loaded

The transmit and command RAM are user write-only. The receive RAM is user read-only. Figure 20-11 shows the RAM configuration. The RAM contents are undefined immediately after a reset.

The command and data RAM in the QSPI is indirectly accessible with QDR and QAR as 48 separate locations that comprise 16 words of transmit data, 16 words of receive data and 16 bytes of commands.

A write to QDR causes data to be written to the RAM entry specified by QAR[ADDR] and causes the value in QAR to increment. Correspondingly, a read at QDR returns the data in the RAM at the address specified by QAR[ADDR]. This also causes QAR to increment. A read access requires a single wait state.

Relative Address	Register	Function
0x00	QTR0	Transmit RAM
0x01	QTR1	
...	...	16 bits wide
0x0F	QTR15	

0x10	QRR0	Receive RAM
0x11	QRR1	
...	...	16 bits wide
0x1F	QRR15	

0x20	QCR0	Command RAM
0x21	QCR1	
...	...	8 bits wide
0x2F	QCR15	

Figure 20-11. QSPI RAM Model

20.3.1.1 Receive RAM

Data received by the QSPI is stored in the receive RAM segment located at 0x10 to 0x1F in the QSPI RAM space. The user reads this segment to retrieve data from the QSPI. Data words with less than 16 bits are stored in the least significant bits of the RAM. Unused bits in a receive queue entry are set to zero upon completion of the individual queue entry.

QWR[CPTQP] shows which queue entries have been executed. The user can query this field to determine which locations in receive RAM contain valid data.

20.3.1.2 Transmit RAM

Data to be transmitted by the QSPI is stored in the transmit RAM segment located at addresses 0x0 to 0xF. The user normally writes 1 word into this segment for each queue command to be executed. The user cannot read data in the transmit RAM.

Outbound data must be written to transmit RAM in a right-justified format. The unused bits are ignored. The QSPI copies the data to its data serializer (shift register) for transmission. The data is transmitted most significant bit first and remains in transmit RAM until overwritten by the user.

20.3.1.3 Command RAM

The CPU writes one byte of control information to this segment for each QSPI command to be executed. Command RAM, referred to as QCR0–15, is write-only memory from a user's perspective.

Command RAM consists of 16 bytes with each byte divided into two fields. The peripheral chip select field controls the QSPI_CS signal levels for the transfer. The command control field provides transfer options.

A maximum of 16 commands can be in the queue. Queue execution proceeds from the address in QWR[NEWQP] through the address in QWR[ENDQP].

The QSPI executes a queue of commands defined by the control bits in each command RAM entry which sequence the following actions:

- Chip-select pins are activated.
- Data is transmitted from transmit RAM and received into the receive RAM.
- The synchronous transfer clock QSPI_CLK is generated.

Before any data transfers begin, control data must be written to the command RAM, and any out-bound data must be written to transmit RAM. Also, the queue pointers must be initialized to the first and last entries in the command queue.

Data transfer is synchronized with the internally generated QSPI_CLK, whose phase and polarity are controlled by QMR[CPHA] and QMR[CPOL]. These control bits determine which QSPI_CLK edge is used to drive outgoing data and to latch incoming data.

20.3.2 Baud Rate Selection

The maximum QSPI clock frequency is one-fourth the clock frequency of the internal bus clock ($f_{sys/2}$). Baud rate is selected by writing a value from 2–255 into QMR[BAUD]. The QSPI uses a prescaler to derive the QSPI_CLK rate from the internal bus clock divided by two.

A baud rate value of zero turns off the QSPI_CLK.

The desired QSPI_CLK baud rate is related to the internal bus clock and QMR[BAUD] by the following expression:

$$QMR[BAUD] = \frac{f_{sys/2}}{2 \times [\text{desired QSPI\_CLK baud rate}]}$$

Table 20-8. QSPI_CLK Frequency as Function of Internal Bus Clock and Baud Rate

Internal Bus Clock = 66 MHz	
QMR [BAUD]	**QSPI_CLK**
2	16.5 MHz
4	8.25 MHz
8	4.1 MHz
16	2.06 MHz
32	1.0 MHz
255	12.9 kHz

20.3.3 Transfer Delays

The QSPI supports programmable delays for the QSPI_CS signals before and after a transfer. The time between QSPI_CS assertion and the leading QSPI_CLK edge, and the time between the end of one transfer and the beginning of the next, are both independently programmable.

The chip select to clock delay enable bit in command RAM, QCR[DSCK], enables the programmable delay period from QSPI_CS assertion until the leading edge of QSPI_CLK. QDLYR[QCD] determines the period of delay before the leading edge of QSPI_CLK. The following expression determines the actual delay before the QSPI_CLK leading edge:

$$QSPI\_CS\text{-to-}QSPI\_CLK\ delay = \frac{QDLYR[QCD]}{f_{sys/2}}$$ QDLYR[QCD] has a range of 1–127.

When QDLYR[QCD] or QCR[DSCK] equals zero, the standard delay of one-half the QSPI_CLK period is used.

The command RAM delay after transmit enable bit, QCR[DT], enables the programmable delay period from the negation of the QSPI_CS signals until the start of the next transfer. The delay after transfer can be used to provide a peripheral deselect interval. A delay can also be inserted between consecutive transfers to allow serial A/D converters to complete conversion. There are two transfer delay options: the user can choose to delay a standard period after serial transfer is complete or can specify a delay period. Writing a value to QDLYR[DTL] specifies a delay period. QCR[DT] determines whether the standard delay period (DT = 0) or the specified delay period (DT = 1) is used. The following expression is used to calculate the delay when DT = 1:

$$\text{Delay after transfer} = \frac{32 \times QDLYR[DTL]}{f_{sys/2}}\quad (DT = 1)$$ where QDLYR[DTL] has a range of 1–255. A zero value for DTL causes a delay-after-transfer value of $8192/f_{sys/}$. Standard delay period (DT = 0) is calculated by the following:

$$\text{Standard delay after transfer} = \frac{17}{f_{sys/2}}\quad (DT = 0)$$ Adequate delay between transfers must be specified for long data streams because the QSPI module requires time to load a transmit RAM entry for transfer. Receiving devices need at least the standard delay between successive transfers. If the internal bus clock is operating at a slower rate, the delay between transfers must be increased proportionately.

20.3.4 Transfer Length

There are two transfer length options. The user can choose a default value of 8 bits or a programmed value of 8 to 16 bits. The programmed value must be written into QMR[BITS]. The command RAM bits per transfer enable field, QCR[BITSE], determines whether the default value (BITSE = 0) or the BITS[3–0] value (BITSE = 1) is used. QMR[BITS] gives the required number of bits to be transferred, with 0b0000 representing 16.

20.3.5 Data Transfer

The transfer operation is initiated by setting QDLYR[SPE]. Shortly after QDLYR[SPE] is set, the QSPI executes the command at the command RAM address pointed to by QWR[NEWQP]. Data at the pointer address in transmit RAM is loaded into the data serializer and transmitted. Data that is simultaneously received is stored at the pointer address in receive RAM.

When the proper number of bits has been transferred, the QSPI stores the working queue pointer value in QWR[CPTQP], increments the working queue pointer, and loads the next data for transfer from the transmit RAM. The command pointed to by the incremented working queue pointer is executed next unless a new value has been written to QWR[NEWQP]. If a new queue pointer value is written while a transfer is in progress, the current transfer is completed normally.

When the CONT bit in the command RAM is set, the QSPI_CS signals are asserted between transfers. When CONT is cleared, QSPI_CS*n* are negated between transfers. Note, the QSPI_CS signals are not high impedance.

When the QSPI reaches the end of the queue, it asserts the SPIF flag, QIR[SPIF]. If QIR[SPIFE] is set, an interrupt request is generated when QIR[SPIF] is asserted. Then the QSPI clears QDLYR[SPE] and stops, unless wraparound mode is enabled.

Wraparound mode is enabled by setting QWR[WREN]. The queue can wrap to pointer address 0x0, or to the address specified by QWR[NEWQP], depending on the state of QWR[WRTO].

In wraparound mode, the QSPI cycles through the queue continuously, even while requesting interrupt service. QDLYR[SPE] is not cleared when the last command in the queue is executed. New receive data overwrites previously received data in the receive RAM. Each time the end of the queue is reached, QIR[SPIFE] is set. QIR[SPIF] is not automatically reset. If interrupt driven QSPI service is used, the service routine must clear QIR[SPIF] to abort the current request. Additional interrupt requests during servicing can be prevented by clearing QIR[SPIFE].

There are two recommended methods of exiting wraparound mode: clearing QWR[WREN] or setting QWR[HALT]. Exiting wraparound mode by clearing QDLYR[SPE] is not recommended because this may abort a serial transfer in progress. The QSPI sets SPIF, clears QDLYR[SPE], and stops the first time it reaches the end of the queue after QWR[WREN] is cleared. After QWR[HALT] is set, the QSPI finishes the current transfer, then stops executing commands. After the QSPI stops, QDLYR[SPE] can be cleared.

20.3.6   Initialization/Application Information

The following steps are necessary to set up the QSPI 12-bit data transfers and a QSPI_CLK of 4.125 MHz. The QSPI RAM is set up for a queue of 16 transfers. All QSPI_CS signals are used in this example.

1. Write the QMR with 0xB308 to set up 12-bit data words with the data shifted on the falling clock edge, and a QSPI_CLK frequency of 4.125 MHz (assuming a 66-MHz internal bus clock).

2. Write QDLYR with the desired delays.

3. Write QIR with 0xD00F to enable write collision, abort bus errors, and clear any interrupts.

4. Write QAR with 0x0020 to select the first command RAM entry.

5. Write QDR with 0x7E00, 0x7E00, 0x7E00, 0x7E00, 0x7D00, 0x7D00, 0x7D00, 0x7D00, 0x7B00, 0x7B00, 0x7B00, 0x7B00, 0x7700, 0x7700, 0x7700, and 0x7700 to set up four transfers for each chip select. The chip selects are active low in this example.

6. Write QAR with 0x0000 to select the first transmit RAM entry.

7. Write QDR with sixteen 12-bit words of data.

8. Write QWR with 0x0F00 to set up a queue beginning at entry 0 and ending at entry 15.

9. Set QDLYR[SPE] to enable the transfers.

10. Wait until the transfers are complete. QIR[SPIF] is set when the transfers are complete.

11. Write QAR with 0x0010 to select the first receive RAM entry.

12. Read QDR to get the received data for each transfer.

13. Repeat steps 5 through 13 to do another transfer.

Chapter 21
UART Modules

21.1 Introduction

This chapter describes the use of the three universal asynchronous receiver/transmitters (UARTs) and includes programming examples.

NOTE

The designation '*n*' is used throughout this section to refer to registers or signals associated with one of the three identical UART modules: UART0, UART1, or UART2.

21.1.1 Overview

Each UART can be clocked by the internal bus clock, eliminating the need for an external UART clock. As Figure 21-1 shows, each UART module interfaces directly to the CPU and consists of the following:

- Serial communication channel
- Programmable clock generation
- Interrupt control logic and DMA request logic
- Internal channel control logic
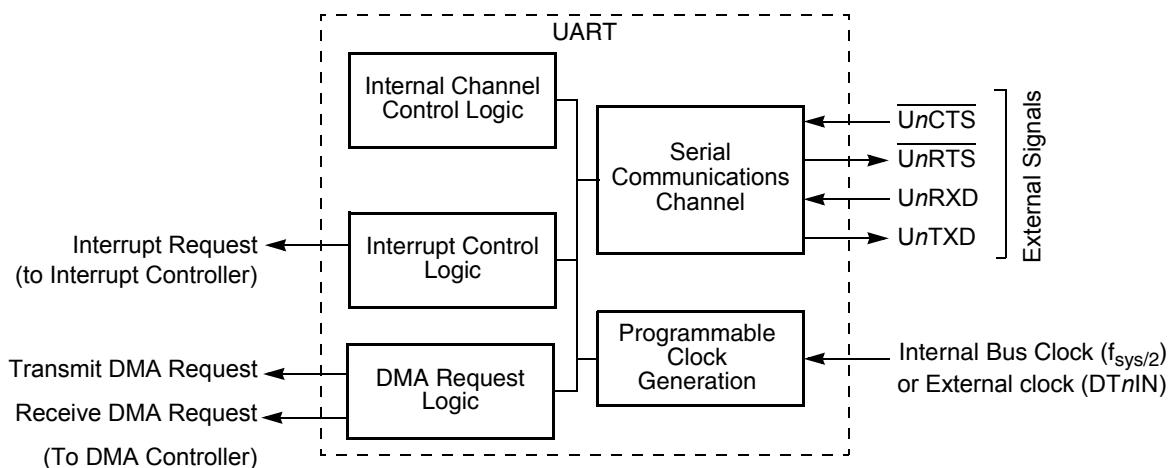
Figure 21-1. UART Block Diagram

NOTE

UART*n* can be clocked by the DT*n*IN pin. However, if the timers are used, then input capture mode is not available for that timer.

The serial communication channel provides a full-duplex asynchronous/synchronous receiver and transmitter deriving an operating frequency from the internal bus clock or an external clock using the timer pin. The transmitter converts parallel data from the CPU to a serial bit stream, inserting appropriate start, stop, and parity bits. It outputs the resulting stream on the channel transmitter serial data output (U*n*TXD). See Section 21.4.2.1, "Transmitter."

The receiver converts serial data from the channel receiver serial data input (U*n*RXD) to parallel format, checks for a start, stop, and parity bits, or break conditions, and transfers the assembled character onto the bus during read operations. The receiver may be polled, interrupt driven, or use DMA requests for servicing. See Section 21.4.2.2, "Receiver."

NOTE

The GPIO module must be configured to enable the peripheral function of the appropriate pins (refer to Chapter 13, "General Purpose I/O Module") prior to configuring the UART module.

21.1.2   Features

The device contains three independent UART modules with the following features:

- Each clocked by an external clock or by the internal bus clock (eliminating a need for an external UART clock)
- Full-duplex asynchronous/synchronous receiver/transmitter channel
- Quadruple-buffered receiver
- Double-buffered transmitter
- Independently programmable receiver and transmitter clock sources
- Programmable data format:
  — 5–8 data bits plus parity
  — Odd, even, no parity, or force parity
  — One, one-and-a-half, or two stop bits
- Each channel programmable to normal (full-duplex), automatic echo, local loop-back, or remote loop-back mode
- Automatic wake-up mode for multidrop applications
- Four maskable interrupt conditions
- All three UARTs have DMA request capability
- Parity, framing, and overrun error detection
- False-start bit detection
- Line-break detection and generation

- Detection of breaks originating in the middle of a character
- Start/end break interrupt/status

21.2 External Signal Description

Figure 21-1 shows both the external and internal signal groups.

An internal interrupt request signal is provided to notify the interrupt controller of an interrupt condition. The output is the logical NOR of unmasked UISR*n* bits. The interrupt level and priority are programmed in the interrupt controller. See Chapter 14, "Interrupt Controller Modules" for more information.

Note that the UARTs can also be configured to automatically transfer data by using the DMA rather than interrupting the core. When there is data in the receiver FIFO or when the transmit holding register is empty, a DMA request can be issued. For more information on generating DMA requests, refer to Section 21.4.6.1.2, "Setting up the UART to Request DMA Service," and Chapter 16, "Enhanced Direct Memory Access (eDMA)."

Table 21-1 briefly describes the UART module signals.

NOTE

The terms 'assertion' and 'negation' are used to avoid confusion between active-low and active-high signals. 'Asserted' indicates that a signal is active, independent of the voltage level; 'negated' indicates that a signal is inactive.

Table 21-1. UART Module Signals

Signal	Description
Transmitter Serial Data Output (U*n*TXD)	U*n*TXD is held high (mark condition) when the transmitter is disabled, idle, or operating in the local loop-back mode. Data is shifted out on U*n*TXD on the falling edge of the clock source, with the least significant bit (lsb) sent first.
Receiver Serial Data Input (U*n*RXD)	Data received on U*n*RXD is sampled on the rising edge of the clock source, with the lsb received first.
Clear-to- Send (U*n*CTS)	This input can generate an interrupt on a change of state.
Request-to-Send (U*n*RTS)	This output can be programmed to be negated or asserted automatically by either the receiver or the transmitter. When connected to a transmitter's U*n*CTS, U*n*RTS can control serial data flow.

Figure 21-2 shows a signal configuration for a UART/RS-232 interface.
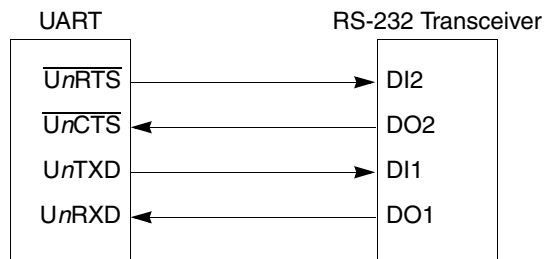
Figure 21-2. UART/RS-232 Interface

21.3 Memory Map/Register Definition

This section contains a detailed description of each register and its specific function. Flowcharts in Section 21.4.6, "Programming," describe basic UART module programming. The operation of the UART module is controlled by writing control bytes into the appropriate registers. Table 21-2 is a memory map for UART module registers.

NOTE
UART registers are accessible only as bytes.

NOTE
Interrupt can mean either an interrupt request asserted to the CPU or a DMA request.

Table 21-2. UART Module Memory Map

ISPBAR Offset UART0 UART1 UART2	Register	Access	Reset Value	Section/Page
0x00_0200 0x00_0240 0x00_0280	UART Mode Registers[1] (UMR1n), (UMR2n)	R/W	0x00	21.3.1/21-5 21.3.2/21-6
0x00_0204 0x00_0244 0x00_0284	UART Status Register (USRn)	R	0x00	21.3.3/21-8
	UART Clock Select Register[1] (UCSRn)	W	0x00	21.3.4/21-9
0x00_0208 0x00_0248 0x00_0288	UART Command Registers (UCRn)	W	0x00	21.3.5/21-10
0x00_020C 0x00_024C 0x00_028C	UART Receive Buffers (URBn)	R	0xFF	21.3.6/21-12
	UART Transmit Buffers (UTBn)	W	0x00	21.3.7/21-13
0x00_0210 0x00_0250 0x00_0290	UART Input Port Change Register (UIPCRn)	R	[$\overline{UnCTS}$]	21.3.8/21-13
	UART Auxiliary Control Register (UACRn)	W	0x00	21.3.9/21-14

Table 21-2. UART Module Memory Map (continued)

ISPBAR Offset UART0 UART1 UART2	Register	Access	Reset Value	Section/Page
0x00_0214 0x00_0254 0x00_0294	UART Interrupt Status Register (UISR*n*)	R	0x00	21.3.10/21-14
	UART Interrupt Mask Register (UIMR*n*)	W	0x00	
0x00_0218 0x00_0258 0x00_0298	UART Baud Rate Generator Register (UBG1*n*)	W[2]	0x00	21.3.11/21-16
0x00_021C 0x00_025C 0x00_029C	UART Baud Rate Generator Register (UBG2*n*)	W[2]	0x00	21.3.11/21-16
0x00_0234 0x00_0274 0x00_02B4	UART Input Port Register (UIP*n*)	R	0xFF	21.3.12/21-17
0x00_0238 0x00_0278 0x00_02B8	UART Output Port Bit Set Command Register (UOP1*n*)	W[2]	0x00	21.3.13/21-17
0x00_023C 0x00_027C 0x00_02BC	UART Output Port Bit Reset Command Register (UOP0*n*)	W[2]	0x00	21.3.13/21-17

[1] UMR1*n*, UMR2*n*, and UCSR*n* should be changed only after the receiver/transmitter is issued a software reset command. That is, if channel operation is not disabled, undesirable results may occur.

[2] Reading this register results in undesired effects and possible incorrect transmission or reception of characters. Register contents may also be changed.

21.3.1   UART Mode Registers 1 (UMR1*n*)

The UMR1*n* registers control configuration. UMR1*n* can be read or written when the mode register pointer points to it, at RESET or after a RESET MODE REGISTER POINTER command using UCR*n*[MISC]. After UMR1*n* is read or written, the pointer points to UMR2*n*.

Address: 0x00_0200 (UMR10)   Access: User read/write[1]
         0x00_0240 (UMR11)
         0x00_0280 (UMR12)

	7	6	5	4	3	2	1	0
R W	RXRTS	RXIRQ/ FFULL	ERR	PM		PT	B/C	
Reset:	0	0	0	0	0	0	0	0
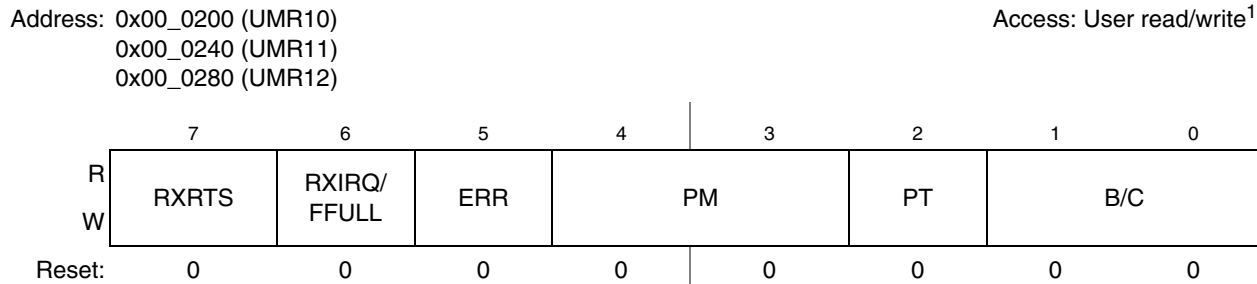
[1] After UMR1*n* is read or written, the pointer points to UMR2*n*

Figure 21-3. UART Mode Registers 1 (UMR1*n*)

MCF5213 Reference Manual, Rev. 1.1

Preliminary

Table 21-3. UMR1*n* Field Descriptions

Field	Description
7 RXRTS	Receiver request-to-send. Allows the U*n*RTS output to control the U*n*CTS input of the transmitting device to prevent receiver overrun. If both the receiver and transmitter are incorrectly programmed for U*n*RTS control, U*n*RTS control is disabled for both. Transmitter RTS control is configured in UMR2*n*[TXRTS]. 0 The receiver has no effect on U*n*RTS. 1 When a valid start bit is received, U*n*RTS is negated if the UART's FIFO is full. U*n*RTS is reasserted when the FIFO has an empty position available.
6 RXIRQ/ FFULL	Receiver interrupt select. 0 RXRDY is the source that generates interrupt or DMA requests. 1 FFULL is the source that generates interrupt or DMA requests.
5 ERR	Error mode. Configures the FIFO status bits, USR*n*[RB,FE,PE]. 0 Character mode. The USR*n* values reflect the status of the character at the top of the FIFO. ERR must be 0 for correct A/D flag information when in multidrop mode. 1 Block mode. The USR*n* values are the logical OR of the status for all characters reaching the top of the FIFO since the last RESET ERROR STATUS command for the channel was issued. See Section 21.3.5, "UART Command Registers (UCRn)."
4–3 PM	Parity mode. Selects the parity or multidrop mode for the channel. The parity bit is added to the transmitted character, and the receiver performs a parity check on incoming data. The value of PM affects PT, as shown below.
2 PT	Parity type. PM and PT together select parity type (PM = 0x) or determine whether a data or address character is transmitted (PM = 11). <table><tr><th>PM</th><th>Parity Mode</th><th>Parity Type (PT= 0)</th><th>Parity Type (PT= 1)</th></tr><tr><td>00</td><td>With parity</td><td>Even parity</td><td>Odd parity</td></tr><tr><td>01</td><td>Force parity</td><td>Low parity</td><td>High parity</td></tr><tr><td>10</td><td>No parity</td><td colspan="2">n/a</td></tr><tr><td>11</td><td>Multidrop mode</td><td>Data character</td><td>Address character</td></tr></table>
1–0 B/C	Bits per character. Select the number of data bits per character to be sent. The values shown do not include start, parity, or stop bits. 00 5 bits 01 6 bits 10 7 bits 11 8 bits

21.3.2 UART Mode Register 2 (UMR2*n*)

The UMR2*n* registers control UART module configuration. UMR2*n* can be read or written when the mode register pointer points to it, which occurs after any access to UMR1*n*. UMR2*n* accesses do not update the pointer.

Preliminary

Address: 0x00_0200 (UMR20)   Access: User read/write[1]
0x00_0240 (UMR21)
0x00_0280 (UMR22)

	7	6	5	4	3	2	1	0
R	CM		TXRTS	TXCTS	SB			
W								
Reset:	0	0	0	0	0	0	0	0
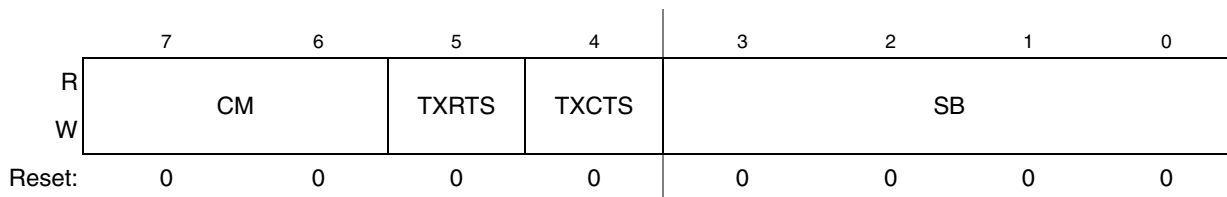
[1] After UMR1*n* is read or written, the pointer points to UMR2*n*

Figure 21-5. UART Mode Register 2 (UMR2*n*)

Table 21-4. UMR2*n* Field Descriptions

Field	Description
7–6 CM	Channel mode. Selects a channel mode. Section 21.4.3, "Looping Modes," describes individual modes. 00 Normal 01 Automatic echo 10 Local loop-back 11 Remote loop-back
5 TXRTS	Transmitter ready-to-send. Controls negation of $\overline{UnRTS}$ to automatically terminate a message transmission. Attempting to program a receiver and transmitter in the same channel for $\overline{UnRTS}$ control is not permitted and disables $\overline{UnRTS}$ control for both. 0 The transmitter has no effect on $\overline{UnRTS}$. 1 In applications where the transmitter is disabled after transmission completes, setting this bit automatically clears UOP[RTS] one bit time after any characters in the channel transmitter shift and holding registers are completely sent, including the programmed number of stop bits.

Table 21-4. UMR2*n* Field Descriptions (continued)

Field	Description
4 TXCTS	Transmitter clear-to-send. If both TXCTS and TXRTS are set, TXCTS controls the operation of the transmitter. 0 U*n*CTS has no effect on the transmitter. 1 Enables clear-to-send operation. The transmitter checks the state of U*n*CTS each time it is ready to send a character. If U*n*CTS is asserted, the character is sent; if it is deasserted, the channel U*n*TXD remains in the high state and transmission is delayed until U*n*CTS is asserted. Changes in U*n*CTS as a character is being sent do not affect its transmission.
3–0 SB	Stop-bit length control. Selects the length of the stop bit appended to the transmitted character. Stop-bit lengths of 9/16 to 2 bits are programmable for 6–8 bit characters. Lengths of 1-1/16 to 2 bits are programmable for 5-bit characters. In all cases, the receiver checks only for a high condition at the center of the first stop-bit position, that is, one bit time after the last data bit or after the parity bit, if parity is enabled. If an external 1x clock is used for the transmitter, clearing bit 3 selects one stop bit and setting bit 3 selects two stop bits for transmission.

SB	5 Bits	6–8 Bits		SB	5–8 Bits
0000	1.063	0.563		1000	1.563
0001	1.125	0.625		1001	1.625
0010	1.188	0.688		1010	1.688
0011	1.250	0.750		1011	1.750
0100	1.313	0.813		1100	1.813
0101	1.375	0.875		1101	1.875
0110	1.438	0.938		1110	1.938
0111	1.500	1.000		1111	2.000

21.3.3 UART Status Registers (USR*n*)

The USR*n* registers, shown in Figure 21-6, show the status of the transmitter, the receiver, and the FIFO.

Address: 0x00_0204 (UCSR0)   Access: User read-only
0x00_0244 (UCSR1)
0x00_0284 (UCSR2)

	7	6	5	4	3	2	1	0
R	RB	FE	PE	OE	TXEMP	TXRDY	FFULL	RXRDY
W								
Reset:	0	0	0	0	0	0	0	0
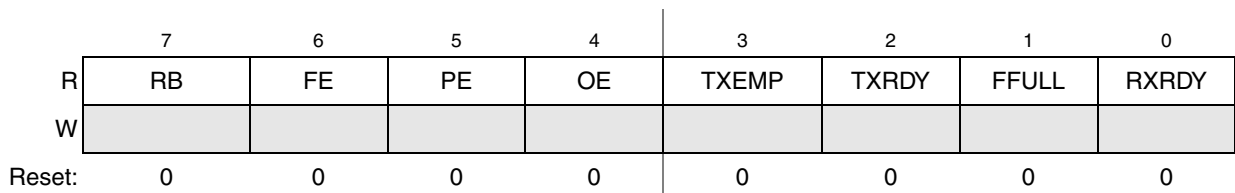
Figure 21-6. UART Status Register (USR*n*)

Table 21-5. USR*n* Field Descriptions

Field	Description
7 RB	Received break. The received break circuit detects breaks that originate in the middle of a received character. However, a break in the middle of a character must persist until the end of the next detected character time. 0 No break was received. 1 An all-zero character of the programmed length was received without a stop bit. Only a single FIFO position is occupied when a break is received. Further entries to the FIFO are inhibited until U*n*RXD returns to the high state for at least one-half bit time, which is equal to two successive edges of the UART clock. RB is valid only when RXRDY = 1.
6 FE	Framing error. 0 No framing error occurred. 1 No stop bit was detected when the corresponding data character in the FIFO was received. The stop-bit check occurs in the middle of the first stop-bit position. FE is valid only when RXRDY = 1.
5 PE	Parity error. Valid only if RXRDY = 1. 0 No parity error occurred. 1 If UMR1*n*[PM] = 0*x* (with parity or force parity), the corresponding character in the FIFO was received with incorrect parity. If UMR1*n*[PM] = 11 (multidrop), PE stores the received address or data (A/D) bit. PE is valid only when RXRDY = 1.
4 OE	Overrun error. Indicates whether an overrun occurs. 0 No overrun occurred. 1 One or more characters in the received data stream have been lost. OE is set upon receipt of a new character when the FIFO is full and a character is already in the shift register waiting for an empty FIFO position. When this occurs, the character in the receiver shift register and its break detect, framing error status, and parity error, if any, are lost. OE is cleared by the RESET ERROR STATUS command in UCR*n*.
3 TEMP	Transmitter empty. 0 The transmit buffer is not empty. Either a character is being shifted out, or the transmitter is disabled. The transmitter is enabled/disabled by programming UCR*n*[TC]. 1 The transmitter has underrun (both the transmitter holding register and transmitter shift registers are empty). This bit is set after transmission of the last stop bit of a character if there are no characters in the transmitter holding register awaiting transmission.
2 TXRDY	Transmitter ready. 0 The CPU loaded the transmitter holding register or the transmitter is disabled. 1 The transmitter holding register is empty and ready for a character. TXRDY is set when a character is sent to the transmitter shift register or when the transmitter is first enabled. If the transmitter is disabled, characters loaded into the transmitter holding register are not sent.
1 FFULL	FIFO full. 0 The FIFO is not full but may hold up to two unread characters. 1 A character was received and the receiver FIFO is now full. Any characters received when the FIFO is full are lost.
0 RXRDY	Receiver ready. 0 The CPU has read the receive buffer and no characters remain in the FIFO after this read. 1 One or more characters were received and are waiting in the receive buffer FIFO.

21.3.4 UART Clock Select Registers (UCSR*n*)

The UCSRs select an external clock on the DTIN input (divided by 1 or 16) or a prescaled internal bus clock as the clocking source for the transmitter and receiver. See Section 21.4.1, "Transmitter/Receiver Clock Source." The transmitter and receiver can use different clock sources. To use the internal bus clock for both, set UCSR*n* to 0xDD.

Address: 0x00_0204 (UCSR0) Access: User write-only
0x00_0244 (UCSR1)
0x00_0284 (UCSR2)

	7	6	5	4	3	2	1	0
R								
W			RCS				TCS	
Reset:	0	0	0	0	0	0	0	0
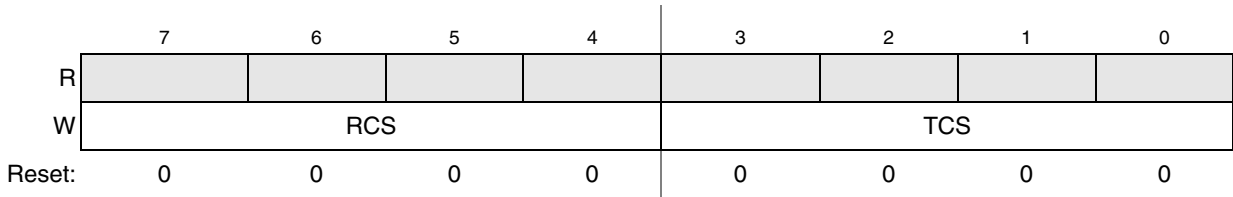
Figure 21-7. UART Clock Select Register (UCSR*n*)

Table 21-6. UCSR*n* Field Descriptions

Field	Description
7–4 RCS	Receiver clock select. Selects the clock source for the receiver channel. 1101 Prescaled internal bus clock ($f_{sys/2}$) 1110 DTIN divided by 16 1111 DTIN
3–0 TCS	Transmitter clock select. Selects the clock source for the transmitter channel. 1101 Prescaled internal bus clock ($f_{sys/2}$) 1110 DTIN divided by 16 1111 DTIN

21.3.5 UART Command Registers (UCR*n*)

The UCRs, shown in Figure 21-8, supply commands to the UART. Only multiple commands that do not conflict can be specified in a single write to a UCR*n*. For example, RESET TRANSMITTER and ENABLE TRANSMITTER cannot be specified in one command.

Address: 0x00_0208 (UCR0) Access: User write-only
0x00_0248 (UCR1)
0x00_0288 (UCR2)

	7	6	5	4	3	2	1	0
R								
W	0		MISC			TC		RC
Reset:	0	0	0	0	0	0	0	0
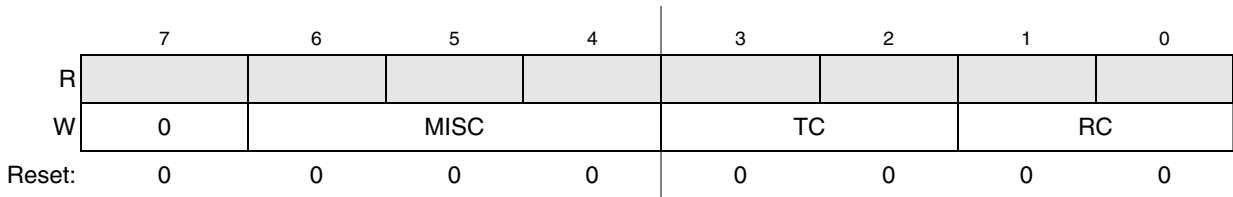
Figure 21-8. UART Command Register (UCR*n*)

Table 21-7 describes UCR*n* fields and commands. Examples in Section 21.4.2, "Transmitter and Receiver Operating Modes," show how these commands are used.

Table 21-7. UCR*n* Field Descriptions

Field	Description
7	Reserved, should be cleared.
6–4 MISC	MISC Field (this field selects a single command) *(see command table below)*

	Command	Description
000	NO COMMAND	—
001	RESET MODE REGISTER POINTER	Causes the mode register pointer to point to UMR1*n*.
010	RESET RECEIVER	Immediately disables the receiver, clears USR*n*[FFULL,RXRDY], and reinitializes the receiver FIFO pointer. No other registers are altered. Because it places the receiver in a known state, use this command instead of RECEIVER DISABLE when reconfiguring the receiver.
011	RESET TRANSMITTER	Immediately disables the transmitter and clears USR*n*[TXEMP,TXRDY]. No other registers are altered. Because it places the transmitter in a known state, use this command instead of TRANSMITTER DISABLE when reconfiguring the transmitter.
100	RESET ERROR STATUS	Clears USR*n*[RB,FE,PE,OE]. Also used in block mode to clear all error bits after a data block is received.
101	RESET BREAK–CHANGE INTERRUPT	Clears the delta break bit, UISR*n*[DB].
110	START BREAK	Forces U*n*TXD low. If the transmitter is empty, the break may be delayed up to one bit time. If the transmitter is active, the break starts when character transmission completes. The break is delayed until any character in the transmitter shift register is sent. Any character in the transmitter holding register is sent after the break. The transmitter must be enabled for the command to be accepted. This command ignores the state of $\overline{\text{U}n\text{CTS}}$.
111	STOP BREAK	Causes U*n*TXD to go high (mark) within two bit times. Any characters in the transmit buffer are sent.

Table 21-7. UCR*n* Field Descriptions (continued)

Field	Description
3–2 TC	TC Field (This field selects a single command)

	Command	Description
00	NO ACTION TAKEN	Causes the transmitter to stay in its current mode: if the transmitter is enabled, it remains enabled; if the transmitter is disabled, it remains disabled.
01	TRANSMITTER ENABLE	Enables operation of the channel's transmitter. USR*n*[TXEMP,TXRDY] are set. If the transmitter is already enabled, this command has no effect.
10	TRANSMITTER DISABLE	Terminates transmitter operation and clears USR*n*[TXEMP,TXRDY]. If a character is being sent when the transmitter is disabled, transmission completes before the transmitter becomes inactive. If the transmitter is already disabled, the command has no effect.
11	—	Reserved, do not use.

Field	Description
1–0 RC	RC (This field selects a single command)

	Command	Description
00	NO ACTION TAKEN	Causes the receiver to stay in its current mode. If the receiver is enabled, it remains enabled; if disabled, it remains disabled.
01	RECEIVER ENABLE	If the UART module is not in multidrop mode (UMR1*n*[PM] $\neq$ 11), RECEIVER ENABLE enables the channel's receiver and forces it into search-for-start-bit state. If the receiver is already enabled, this command has no effect.
10	RECEIVER DISABLE	Disables the receiver immediately. Any character being received is lost. The command does not affect receiver status bits or other control registers. If the UART module is programmed for local loop-back or multidrop mode, the receiver operates even though this command is selected. If the receiver is already disabled, the command has no effect.
11	—	Reserved, do not use.

21.3.6   UART Receive Buffers (URB*n*)

The receive buffers (shown in Figure 21-9) contain one serial shift register and three receiver holding registers, which act as a FIFO. U*n*RXD is connected to the serial shift register. The CPU reads from the top of the FIFO while the receiver shifts and updates from the bottom when the shift register is full (see Figure 21-20). RB contains the character in the receiver.

Address: 0x00_020C (URB0)   Access: User read-only
         0x00_024C (URB1)
         0x00_028C (URB2)

	7	6	5	4	3	2	1	0
R					RB			
W								
Reset:	1	1	1	1	1	1	1	1
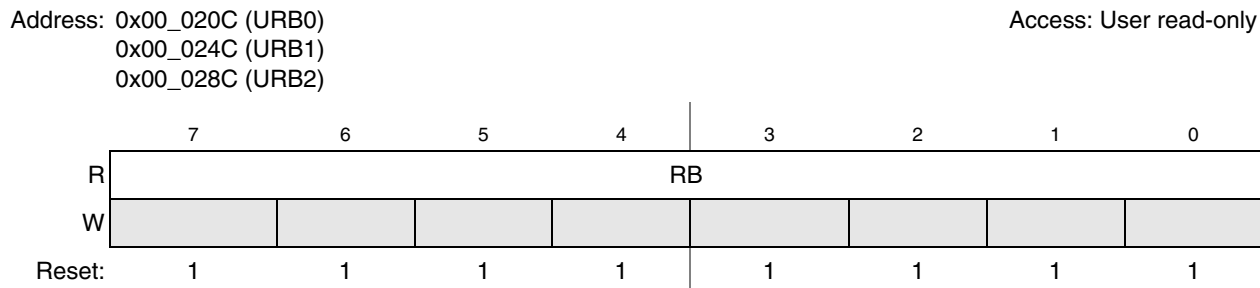
Figure 21-9. UART Receive Buffer (URB*n*)

21.3.7 UART Transmit Buffers (UTB*n*)

The transmit buffers consist of the transmitter holding register and the transmitter shift register. The holding register accepts characters from the bus master if channel's USR*n*[TXRDY] is set. A write to the transmit buffer clears USR*n*[TXRDY], inhibiting any more characters until the shift register can accept more data. When the shift register is empty, it checks if the holding register has a valid character to be sent (TXRDY = 0). If there is a valid character, the shift register loads it and sets USR*n*[TXRDY] again. Writes to the transmit buffer when the channel's TXRDY = 0 and when the transmitter is disabled have no effect on the transmit buffer.

Figure 21-10 shows UTB*n*. TB contains the character in the transmit buffer.

Address: 0x00_020C (UTB0)   Access: User write-only
         0x00_024C (UTB1)
         0x00_028C (UTB2)

	7	6	5	4	3	2	1	0
R								
W					TB			
Reset:	0	0	0	0	0	0	0	0
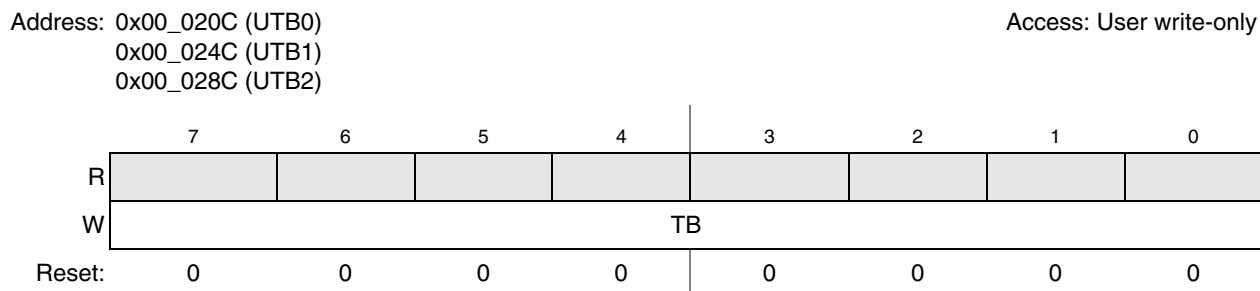
Figure 21-10. UART Transmit Buffer (UTB*n*)

21.3.8 UART Input Port Change Registers (UIPCR*n*)

The UIPCRs, shown in Figure 21-11, hold the current state and the change-of-state for $\overline{UnCTS}$.

Address: 0x00_0210 (UIPCR0)   Access: User read-only
         0x00_0250 (UIPCR1)
         0x00_0290 (UIPCR2)

	7	6	5	4	3	2	1	0
R	0	0	0	COS	0	0	0	CTS
W								
Reset:	0	0	0	0	1	1	1	$\overline{UnCTS}$
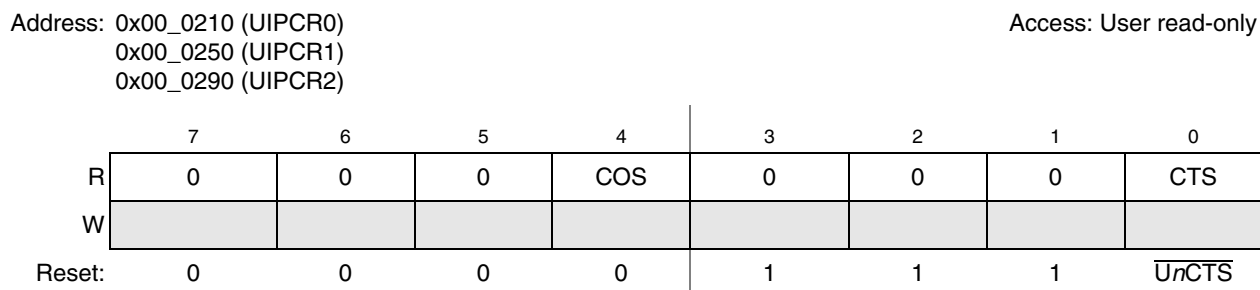
Figure 21-11. UART Input Port Change Register (UIPCR*n*)

Table 21-8. UIPCR*n* Field Descriptions

Field	Description
7–5	Reserved, should be cleared.
4 COS	Change of state (high-to-low or low-to-high transition). 0 No change-of-state since the CPU last read UIPCR*n*. Reading UIPCR*n* clears UISR*n*[COS]. 1 A change-of-state longer than 25–50 μs occurred on the $\overline{U n CTS}$ input. UACR*n* can be programmed to generate an interrupt to the CPU when a change of state is detected.
3–1	Reserved, should be cleared.
0 CTS	Current state of clear-to-send. Starting two serial clock periods after reset, CTS reflects the state of $\overline{U n CTS}$. If $\overline{U n CTS}$ is detected asserted at that time, COS is set, which initiates an interrupt if UACR*n*[IEC] is enabled. 0 The current state of the $\overline{U n CTS}$ input is asserted. 1 The current state of the $\overline{U n CTS}$ input is deasserted.

21.3.9 UART Auxiliary Control Register (UACR*n*)

The UACRs, shown in Figure 21-9, control the input enable.

Address: 0x00_0210 (UACR0)   Access: User write-only
          0x00_0250 (UACR1)
          0x00_0290 (UACR2)

	7	6	5	4	3	2	1	0
R								
W	0	0	0	0	0	0	0	IEC
Reset:	0	0	0	0	0	0	0	0
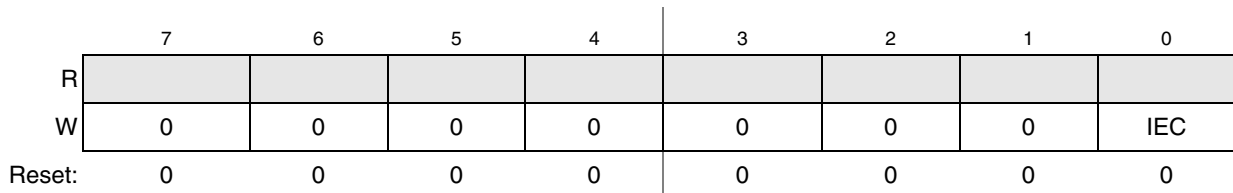
Figure 21-12. UART Auxiliary Control Register (UACR*n*)

Table 21-9. UACR*n* Field Descriptions

Field	Description
7–1	Reserved, should be cleared.
0 IEC	Input enable control. 0 Setting the corresponding UIPCR*n* bit has no effect on UISR*n*[COS]. 1 UISR*n*[COS] is set and an interrupt is generated when the UIPCR*n*[COS] is set by an external transition on the $\overline{U n CTS}$ input (if UIMR*n*[COS] = 1).

21.3.10 UART Interrupt Status/Mask Registers (UISR*n*/UIMR*n*)

The UISRs, shown in Figure 21-13, provide status for all potential interrupt sources. UISR*n* contents are masked by UIMR*n*. If corresponding UISR*n* and UIMR*n* bits are set, the internal interrupt output is asserted. If a UIMR*n* bit is cleared, the state of the corresponding UISR*n* bit has no effect on the output.

The UISR*n* and UIMR*n* registers share the same space in memory. Reading this register provides the user with interrupt status, while writing controls the mask bits.

NOTE

True status is provided in the UISR*n* regardless of UIMR*n* settings. UISR*n* is cleared when the UART module is reset.

Address: 0x00_0214 (UISR0)   Access: User read/write
            0x00_0254 (UISR1)
            0x00_0294 (UISR2)

	7	6	5	4	3	2	1	0
R (UISR*n*)	COS	0	0	0	0	DB	FFULL/ RXRDY	TXRDY
W (UIMR*n*)	COS	0	0	0	0	DB	FFULL/ RXRDY	TXRDY
Reset:	0	0	0	0	0	0	0	0

Figure 21-13. UART Interrupt Status/Mask Registers (UISR*n*/UIMR*n*)

Table 21-10. UISR*n*/UIMR*n* Field Descriptions

Field	Description
7 COS	Change-of-state. 0 UIPCR*n*[COS] is not selected. 1 Change-of-state occurred on U*n*CTS and was programmed in UACR*n*[IEC] to cause an interrupt.
6–3	Reserved, should be cleared.
2 DB	Delta break. 0 No new break-change condition to report. Section 21.3.5, "UART Command Registers (UCRn)," describes the RESET BREAK-CHANGE INTERRUPT command. 1 The receiver detected the beginning or end of a received break.
1 FFULL/ RXRDY	Status of FIFO or receiver, depending on UMR1[FFULL/RXRDY] bit. Duplicate of USR*n*[FIFO] & USR*n*[RXRDY] {{TABLE_BELOW}}
0 TXRDY	Transmitter ready. This bit is the duplication of USR*n*[TXRDY]. 0 The transmitter holding register was loaded by the CPU or the transmitter is disabled. Characters loaded into the transmitter holding register when TXRDY = 0 are not sent. 1 The transmitter holding register is empty and ready to be loaded with a character.

UIMR*n* [FFULL/RXRDY]	UISR*n* [FFULL/RXRDY]	UMR1*n*[FFULL/RXRDY]	
		0 (RXRDY)	1 (FIFO)
0	0	Receiver not ready	FIFO not full
1	0	Receiver not ready	FIFO not full
0	1	Receiver is ready, Do not interrupt	FIFO is full, Do not interrupt
1	1	Receiver is ready, interrupt	FIFO is full, interrupt

Preliminary

21.3.11 UART Baud Rate Generator Registers (UBG1*n*/UBG2*n*)

The UBG1*n* registers hold the MSB, and the UBG2*n* registers hold the LSB of the preload value. UBG1*n* and UBG2*n* concatenate to provide a divider to the internal bus clock for transmitter/receiver operation, as described in Section 21.4.1.2.1, "Internal Bus Clock Baud Rates."

Address: 0x00_0218 (UBG10)   Access: User write-only
         0x00_0258 (UBG11)
         0x00_0298 (UBG12)

	7	6	5	4	3	2	1	0
R								
W				Divider MSB				
Reset:	0	0	0	0	0	0	0	0
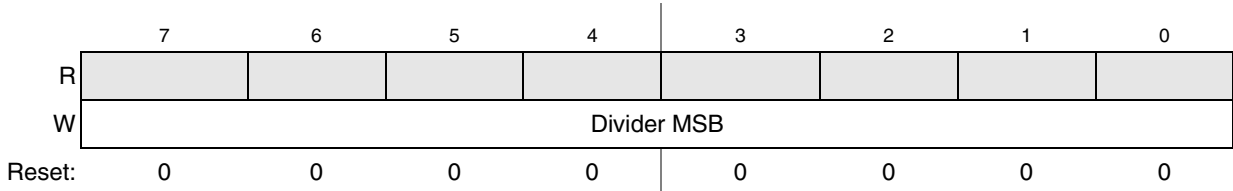
Figure 21-14. UART Baud Rate Generator Register (UBG1*n*)

Address: 0x00_021C (UBG20)   Access: User write-only
         0x00_025C (UBG21)
         0x00_029C (UBG22)

	7	6	5	4	3	2	1	0
R								
W				Divider LSB				
Reset:	0	0	0	0	0	0	0	0
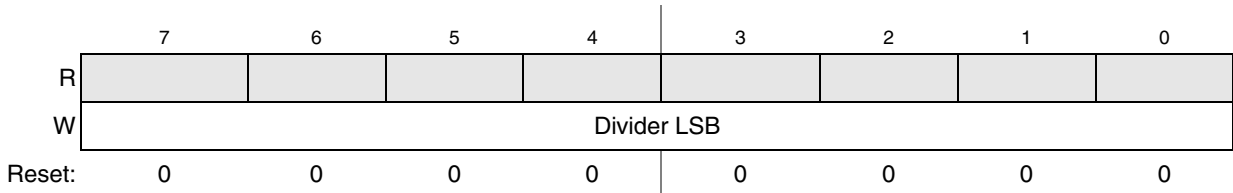
Figure 21-15. UART Baud Rate Generator Register (UBG2*n*)

NOTE

The minimum value that can be loaded on the concatenation of UBG1*n* with UBG2*n* is 0x0002. The UBG2*n* reset value of 0x00 is invalid and must be written to before the UART transmitter or receiver are enabled. Both UBG1*n* and UBG2*n* are write-only and cannot be read by the CPU.

21.3.12 UART Input Port Register (UIP*n*)

The UIP*n* registers, shown in Figure 21-17, show the current state of the $\overline{\text{U}n\text{CTS}}$ input.

Address: 0x00_0234 (UIP0)   Access: User read-only
0x00_0274 (UIP1)
0x00_02B4 (UIP2)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	CTS
W								
Reset:	1	1	1	1	1	1	1	1
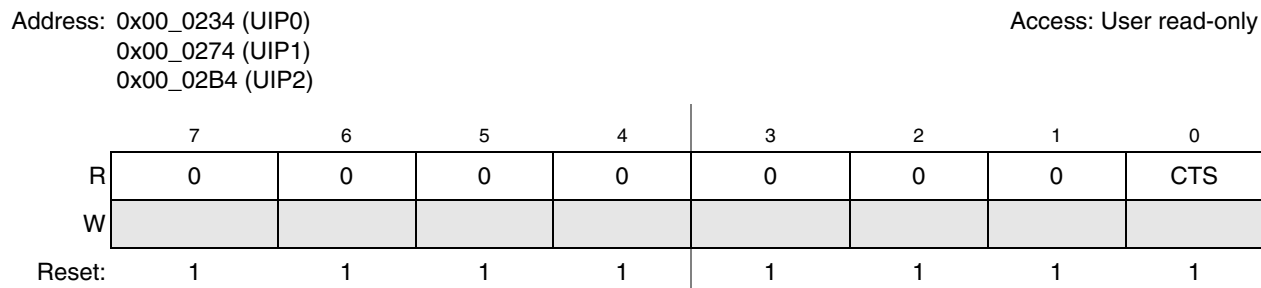
Figure 21-17. UART Input Port Register (UIP*n*)

Table 21-12. UIP*n* Field Descriptions

Field	Description
7–1	Reserved, should be cleared.
0 CTS	Current state of clear-to-send. The $\overline{\text{U}n\text{CTS}}$ value is latched and reflects the state of the input pin when UIP*n* is read. Note: This bit has the same function and value as UIPCR*n*[RTS]. 0 The current state of the $\overline{\text{U}n\text{CTS}}$ input is logic 0. 1 The current state of the $\overline{\text{U}n\text{CTS}}$ input is logic 1.

21.3.13 UART Output Port Command Registers (UOP1*n*/UOP0*n*)

The $\overline{\text{U}n\text{RTS}}$ output can be asserted by writing a 1 to UOP1*n*[RTS] and negated by writing a 1 to UOP0*n*[RTS]. See Figure 21-18.

Address: 0x00_0238 (UOP10)   Access: User write-only
0x00_023C (UOP00)
0x00_0278 (UOP11)
0x00_027C (UOP01)
0x00_02B8 (UOP12)
0x00_02BC (UOP02)

	7	6	5	4	3	2	1	0
R								
W								RTS
Reset:	0	0	0	0	0	0	0	0
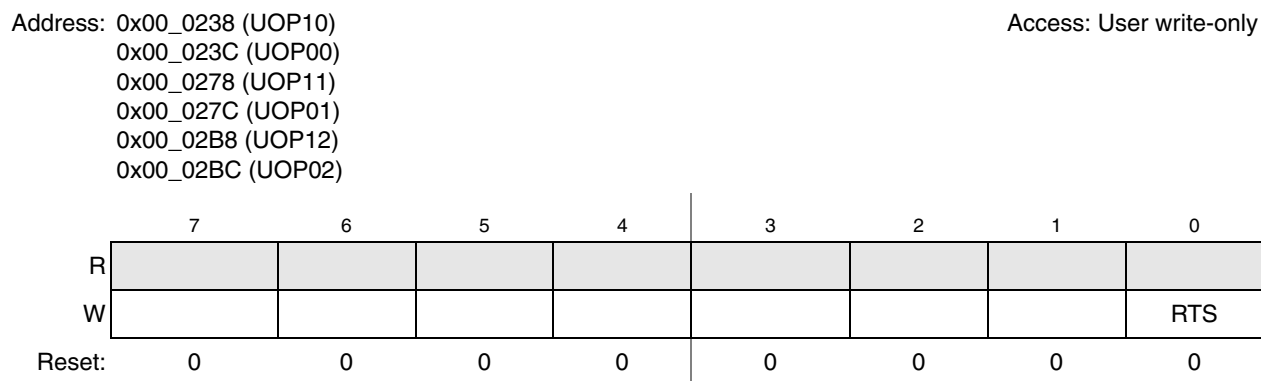
Figure 21-18. UART Output Port Command Registers (UOP1*n*/UOP0*n*)

Table 21-13. UOP1*n*/UOP0*n* Field Descriptions

Field	Description
7–1	Reserved, should be cleared.
0 RTS	Output port output. Controls assertion (UOP1)/negation (UOP0) of $\overline{\text{U}n\text{RTS}}$ output. 0 Not affected. 1 Asserts $\overline{\text{U}n\text{RTS}}$ in UOP1. Negates $\overline{\text{U}n\text{RTS}}$ in UOP0.

MCF5213 Reference Manual, Rev. 1.1

Preliminary

21.4 Functional Description

This section describes operation of the clock source generator, transmitter, and receiver.

21.4.1 Transmitter/Receiver Clock Source

The internal bus clock serves as the basic timing reference for the clock source generator logic, which consists of a clock generator and a programmable 16-bit divider dedicated to each UART. The clock generator might not produce standard baud rates if the internal bus clock is used, so enable the 16-bit divider.

21.4.1.1 Programmable Divider

As Figure 21-19 shows, the UART*n* transmitter and receiver can use the following clock sources:

- An external clock signal on the DT*n*IN pin. When not divided, DT*n*IN provides a synchronous clock mode; when divided by 16, it is asynchronous.
- The internal bus clock supplies an asynchronous clock source that is divided by 32 and then divided by the 16-bit value programmed in UBG1*n* and UBG2*n*. See Section 21.3.11, "UART Baud Rate Generator Registers (UBG1n/UBG2n)."

The choice of DTIN or internal bus clock is programmed in the UCSR.
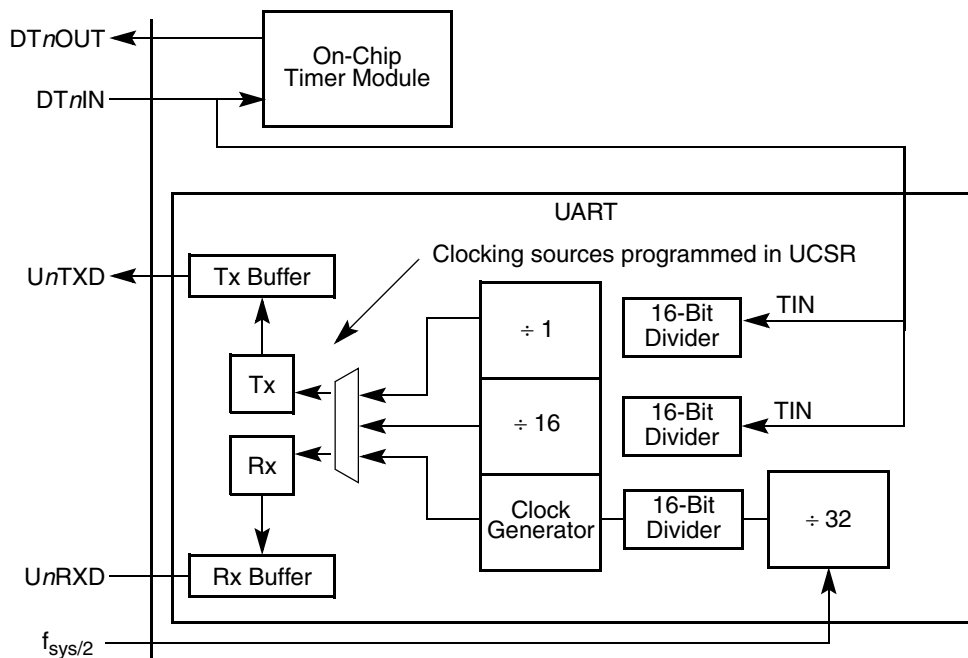
Figure 21-19. Clocking Source Diagram

NOTE

If DT*n*IN is a clocking source for either the timer or UART, that timer module cannot use DT*n*IN for timer input capture.

21.4.1.2 Calculating Baud Rates

The following sections describe how to calculate baud rates.

21.4.1.2.1 Internal Bus Clock Baud Rates

When the internal bus clock is the UART clocking source, it goes through a divide-by-32 prescaler and then passes through the 16-bit divider of the concatenated UBG1n and UBG2n registers. The baud-rate calculation is as follows:

$$\text{Baudrate} = \frac{f_{sys/2}}{[32 \times \text{divider}]} \qquad \textbf{\textit{Eqn. 21-1}}$$

Using a 66 MHz internal bus clock and letting baud rate = 9600, then

$$\text{Divider} = \frac{66\text{MHz}}{[32 \times 9600]} = 215(\text{decimal}) = 0x00D6(\text{hexadecimal}) \qquad \textbf{\textit{Eqn. 21-2}}$$

therefore UBG1n = 0x00 and UBG2n = 0xD6.

21.4.1.2.2 External Clock

An external source clock (DTnIN) can be used as is or divided by 16.

If f_{extc} is the external clock frequency, then the baud rate can be described with this equation:

$$\text{Baudrate} = \frac{f_{extc}}{(16 \text{ or } 1) \times (16\text{-bit divider})} \qquad \textbf{\textit{Eqn. 21-3}}$$

21.4.2 Transmitter and Receiver Operating Modes

Figure 21-20 is a functional block diagram of the transmitter and receiver showing the command and operating registers, which are described generally in the following sections and described in detail in Section 21.3, "Memory Map/Register Definition."
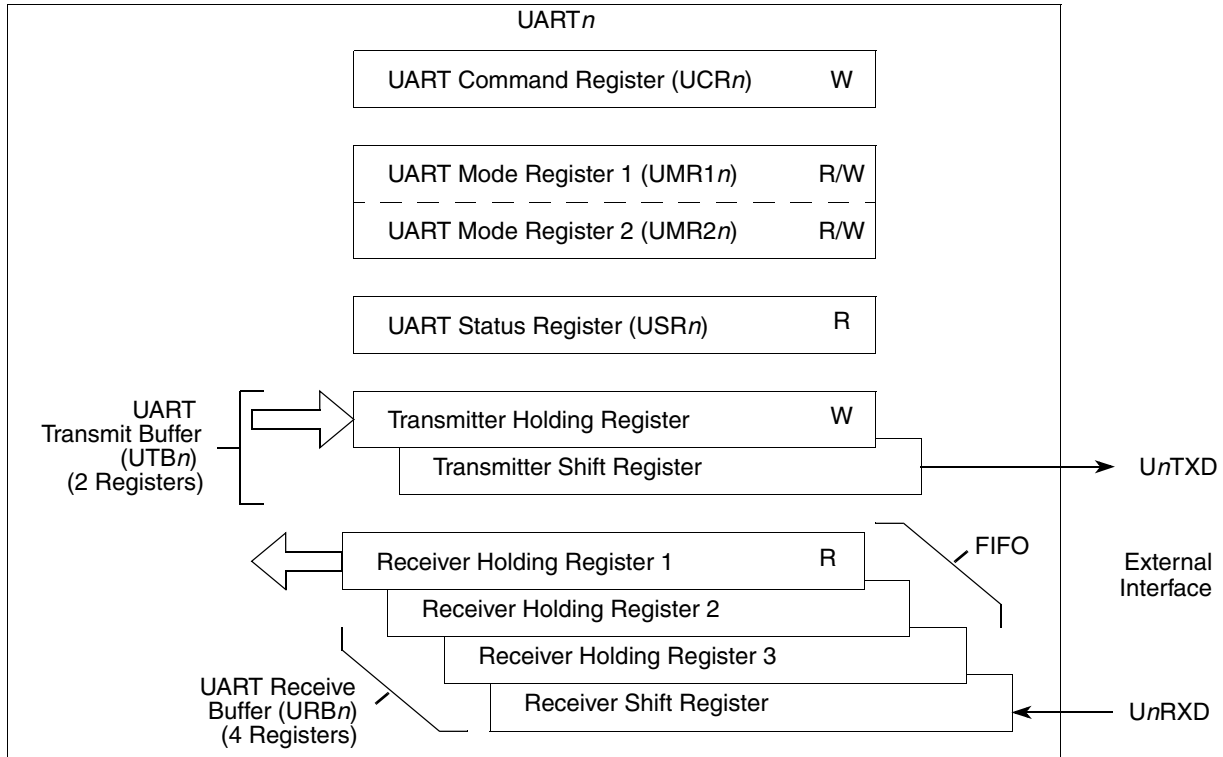
Figure 21-20. Transmitter and Receiver Functional Diagram

21.4.2.1 Transmitter

The transmitter is enabled through the UART command register (UCR*n*). When it is ready to accept a character, the UART sets USR*n*[TXRDY]. The transmitter converts parallel data from the CPU to a serial bit stream on U*n*TXD. It automatically sends a start bit followed by the programmed number of data bits, an optional parity bit, and the programmed number of stop bits. The lsb is sent first. Data is shifted from the transmitter output on the falling edge of the clock source.

After the stop bits are sent, if no new character is in the transmitter holding register, the U*n*TXD output remains high (mark condition) and the transmitter empty bit, USR*n*[TXEMP], is set. Transmission resumes and TXEMP is cleared when the CPU loads a new character into the UART transmit buffer (UTB*n*). If the transmitter receives a disable command, it continues until any character in the transmitter shift register is completely sent.

If the transmitter is reset through a software command, operation stops immediately (see Section 21.3.5, "UART Command Registers (UCRn)"). The transmitter is reenabled through the UCR*n* to resume operation after a disable or software reset.

If the clear-to-send operation is enabled, $\overline{\text{U}n\text{CTS}}$ must be asserted for the character to be transmitted. If $\overline{\text{U}n\text{CTS}}$ is negated in the middle of a transmission, the character in the shift register is sent and U*n*TXD remains in mark state until $\overline{\text{U}n\text{CTS}}$ is reasserted. If the transmitter is forced to send a continuous low condition by issuing a SEND BREAK command, the transmitter ignores the state of $\overline{\text{U}n\text{CTS}}$.

If the transmitter is programmed to automatically negate $\overline{\text{U}n\text{RTS}}$ when a message transmission completes, $\overline{\text{U}n\text{RTS}}$ must be asserted manually before a message is sent. In applications in which the transmitter is disabled after transmission is complete and $\overline{\text{U}n\text{RTS}}$ is appropriately programmed, $\overline{\text{U}n\text{RTS}}$ is negated one bit time after the character in the shift register is completely transmitted. The transmitter must be manually reenabled by reasserting $\overline{\text{U}n\text{RTS}}$ before the next message is to be sent.

Figure 21-21 shows the functional timing information for the transmitter.

^{1} Cn = transmit characters
^{2} W = write
^{3} UMR2n[TXCTS] = 1
^{4} UMR2n[TXRTS] = 1
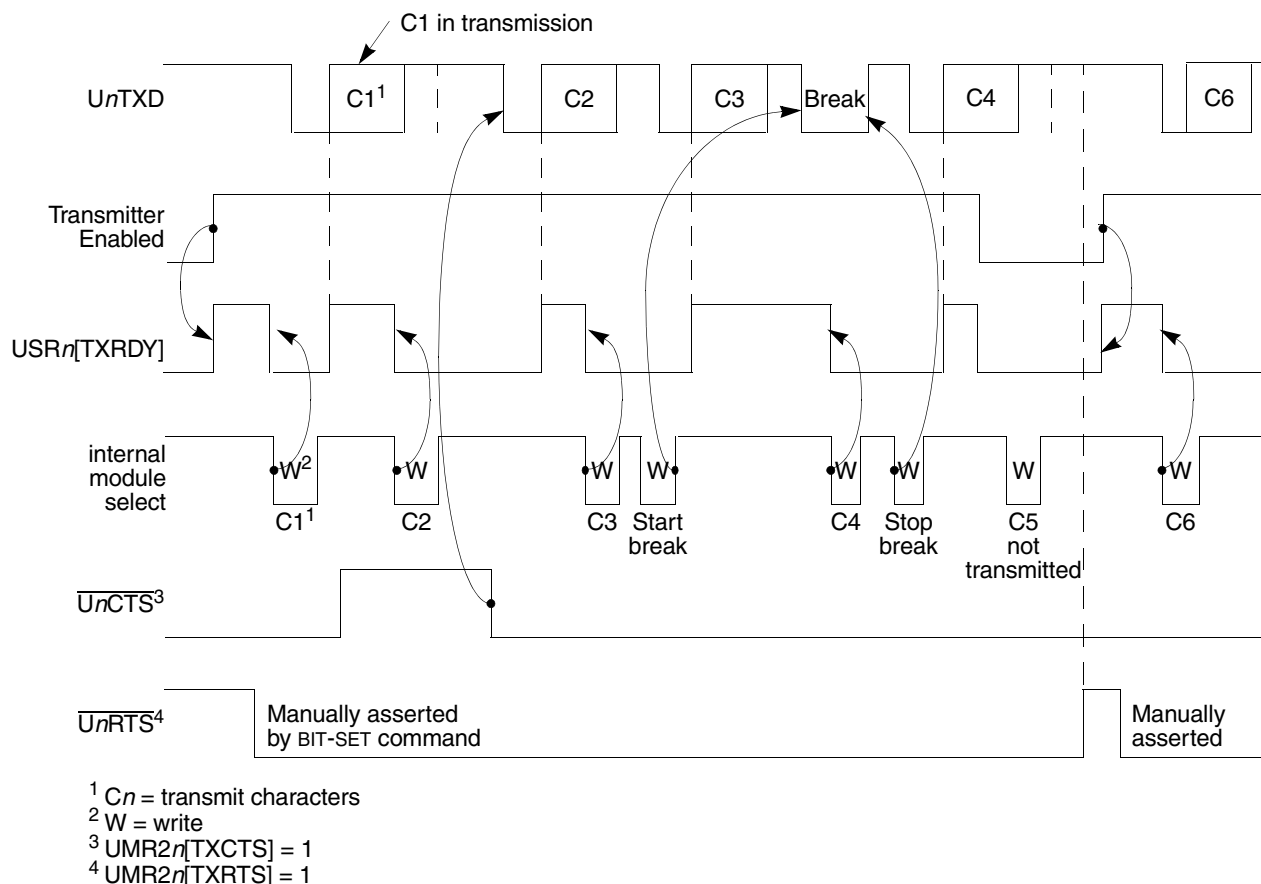
Figure 21-21. Transmitter Timing Diagram

21.4.2.2 Receiver

The receiver is enabled through its UCRn, as described in Section 21.3.5, "UART Command Registers (UCRn)."

When the receiver detects a high-to-low (mark-to-space) transition of the start bit on UnRXD, the state of UnRXD is sampled eight times on the edge of the bit time clock starting one-half clock after the transition (asynchronous operation) or at the next rising edge of the bit time clock (synchronous operation). If UnRXD is sampled high, the start bit is invalid and the search for the valid start bit begins again.

If UnRXD is still low, a valid start bit is assumed and the receiver continues sampling the input at one-bit time intervals, at the theoretical center of the bit, until the proper number of data bits and parity, if any, is assembled and one stop bit is detected. Data on the UnRXD input is sampled on the rising edge of the

programmed clock source. The lsb is received first. The data is then transferred to a receiver holding register and USRn[RXRDY] is set. If the character is less than eight bits, the most significant unused bits in the receiver holding register are cleared.

After the stop bit is detected, the receiver immediately looks for the next start bit. However, if a non-zero character is received without a stop bit (framing error) and UnRXD remains low for one-half of the bit period after the stop bit is sampled, the receiver operates as if a new start bit were detected. Parity error, framing error, overrun error, and received break conditions set the respective PE, FE, OE, RB error and break flags in the USRn at the received character boundary and are valid only if USRn[RXRDY] is set.

If a break condition is detected (UnRXD is low for the entire character including the stop bit), a character of all zeros is loaded into the receiver holding register and USRn[RB,RXRDY] are set. UnRXD must return to a high condition for at least one-half bit time before a search for the next start bit begins.

The receiver detects the beginning of a break in the middle of a character if the break persists through the next character time. If the break begins in the middle of a character, the receiver places the damaged character in the Rx FIFO and sets the corresponding USRn error bits and USRn[RXRDY]. Then, if the break lasts until the next character time, the receiver places an all-zero character into the Rx FIFO and sets USRn[RB,RXRDY].

Figure 21-22 shows receiver functional timing.
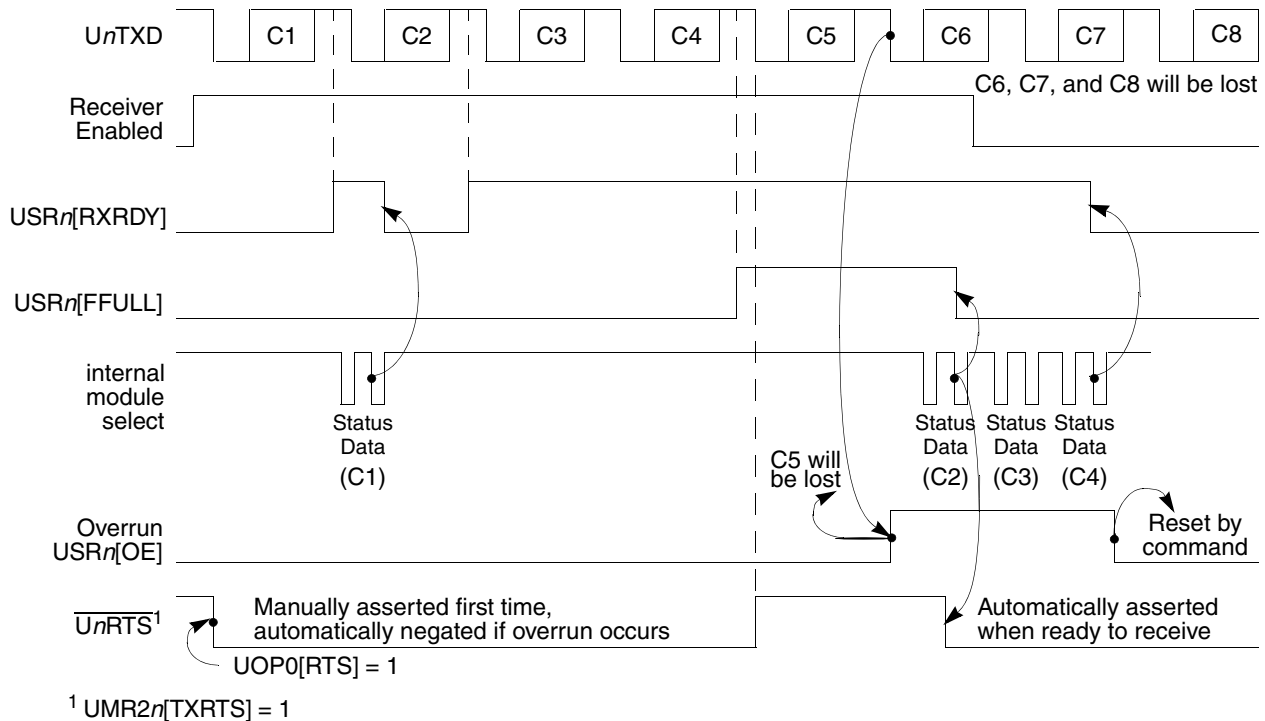
Figure 21-22. Receiver Timing

21.4.2.3 FIFO

The FIFO is used in the UART's receive buffer logic. The FIFO consists of three receiver holding registers. The receive buffer consists of the FIFO and a receiver shift register connected to the UnRXD (see

Figure 21-20). Data is assembled in the receiver shift register and loaded into the top empty receiver holding register position of the FIFO. Thus, data flowing from the receiver to the CPU is quadruple-buffered.

In addition to the data byte, three status bits, parity error (PE), framing error (FE), and received break (RB), are appended to each data character in the FIFO; OE (overrun error) is not appended. By programming the ERR bit in the channel's mode register (UMR1*n*), status is provided in character or block modes.

USR*n*[RXRDY] is set when at least one character is available to be read by the CPU. A read of the receive buffer produces an output of data from the top of the FIFO. After the read cycle, the data at the top of the FIFO and its associated status bits are popped and the receiver shift register can add new data at the bottom of the FIFO. The FIFO-full status bit (FFULL) is set if all three positions are filled with data. Either the RXRDY or FFULL bit can be selected to cause an interrupt and either TXRDY or RXRDY can be used to generate a DMA request.

The two error modes are selected by UMR1*n*[ERR] as follows:

- In character mode (UMR1*n*[ERR] = 0, status is given in the USR*n* for the character at the top of the FIFO.
- In block mode, the USR*n* shows a logical OR of all characters reaching the top of the FIFO since the last RESET ERROR STATUS command. Status is updated as characters reach the top of the FIFO. Block mode offers a data-reception speed advantage where the software overhead of error-checking each character cannot be tolerated. However, errors are not detected until the check is performed at the end of an entire message—the faulting character is not identified.

In either mode, reading the USR*n* does not affect the FIFO. The FIFO is popped only when the receive buffer is read. The USR*n* should be read before reading the receive buffer. If all three receiver holding registers are full, a new character is held in the receiver shift register until space is available. However, if a second new character is received, the contents of the character in the receiver shift register is lost, the FIFOs are unaffected, and USR*n*[OE] is set when the receiver detects the start bit of the new overrunning character.

To support flow control, the receiver can be programmed to automatically negate and assert $\overline{\text{U}n\text{RTS}}$, in which case the receiver automatically negates $\overline{\text{U}n\text{RTS}}$ when a valid start bit is detected and the FIFO is full. The receiver asserts $\overline{\text{U}n\text{RTS}}$ when a FIFO position becomes available; therefore, overrun errors can be prevented by connecting $\overline{\text{U}n\text{RTS}}$ to the $\overline{\text{U}n\text{CTS}}$ input of the transmitting device.

NOTE

The receiver can still read characters in the FIFO if the receiver is disabled. If the receiver is reset, the FIFO, $\overline{\text{U}n\text{RTS}}$ control, all receiver status bits, and interrupts, and DMA requests are reset. No more characters are received until the receiver is reenabled.

21.4.3   Looping Modes

The UART can be configured to operate in various looping modes, as shown in Figure 21-22. These modes are useful for local and remote system diagnostic functions. The modes are described in the following paragraphs and in Section 21.3, "Memory Map/Register Definition."

The UART's transmitter and receiver should be disabled when switching between modes. The selected mode is activated immediately upon mode selection, regardless of whether a character is being received or transmitted.

21.4.3.1   Automatic Echo Mode

In automatic echo mode, shown in Figure 21-23, the UART automatically resends received data bit by bit. The local CPU-to-receiver communication continues normally, but the CPU-to-transmitter link is disabled. In this mode, received data is clocked on the receiver clock and re-sent on U*n*TXD. The receiver must be enabled, but the transmitter need not be.
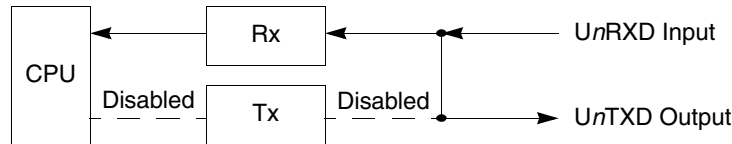
Figure 21-23. Automatic Echo

Because the transmitter is inactive, USR*n*[TXEMP,TXRDY] are inactive and data is sent as it is received. Received parity is checked but not recalculated for transmission. Character framing is also checked, but stop bits are sent as they are received. A received break is echoed as received until the next valid start bit is detected.

21.4.3.2   Local Loop-Back Mode

Figure 21-24 shows how U*n*TXD and U*n*RXD are internally connected in local loop-back mode. This mode is for testing the operation of a local UART module channel by sending data to the transmitter and checking data assembled by the receiver to ensure proper operations.
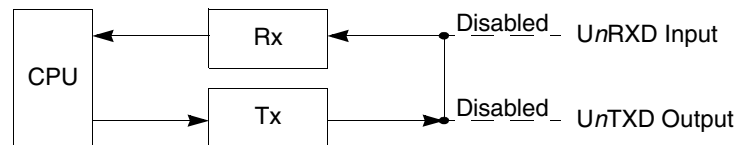
Figure 21-24. Local Loop-Back

Features of this local loop-back mode are as follows:

- Transmitter and CPU-to-receiver communications continue normally in this mode.
- U*n*RXD input data is ignored.
- U*n*TXD is held marking.
- The receiver is clocked by the transmitter clock. The transmitter must be enabled, but the receiver need not be.

21.4.3.3   Remote Loop-Back Mode

In remote loop-back mode, shown in Figure 21-25, the channel automatically transmits received data bit by bit on the U*n*TXD output. The local CPU-to-transmitter link is disabled. This mode is useful in testing

receiver and transmitter operation of a remote channel. For this mode, the transmitter uses the receiver clock.

Because the receiver is not active, received data cannot be read by the CPU and all status conditions are inactive. Received parity is not checked and is not recalculated for transmission. Stop bits are sent as they are received. A received break is echoed as received until the next valid start bit is detected.
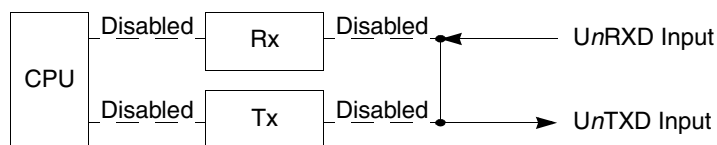
Figure 21-25. Remote Loop-Back

21.4.4 Multidrop Mode

Setting UMR1n[PM] programs the UART to operate in a wake-up mode for multidrop or multiprocessor applications. In this mode, a master can transmit an address character followed by a block of data characters targeted for one of up to 256 slave stations.

Although slave stations have their channel receivers disabled, they continuously monitor the master's data stream. When the master sends an address character, the slave receiver channel notifies its respective CPU by setting USRn[RXRDY] and generating an interrupt (if programmed to do so). Each slave station CPU then compares the received address to its station address and enables its receiver if it wishes to receive the subsequent data characters or block of data from the master station. Slave stations not addressed continue monitoring the data stream. Data fields in the data stream are separated by an address character. After a slave receives a block of data, its CPU disables the receiver and repeats the process. Functional timing information for multidrop mode is shown in Figure 21-26.
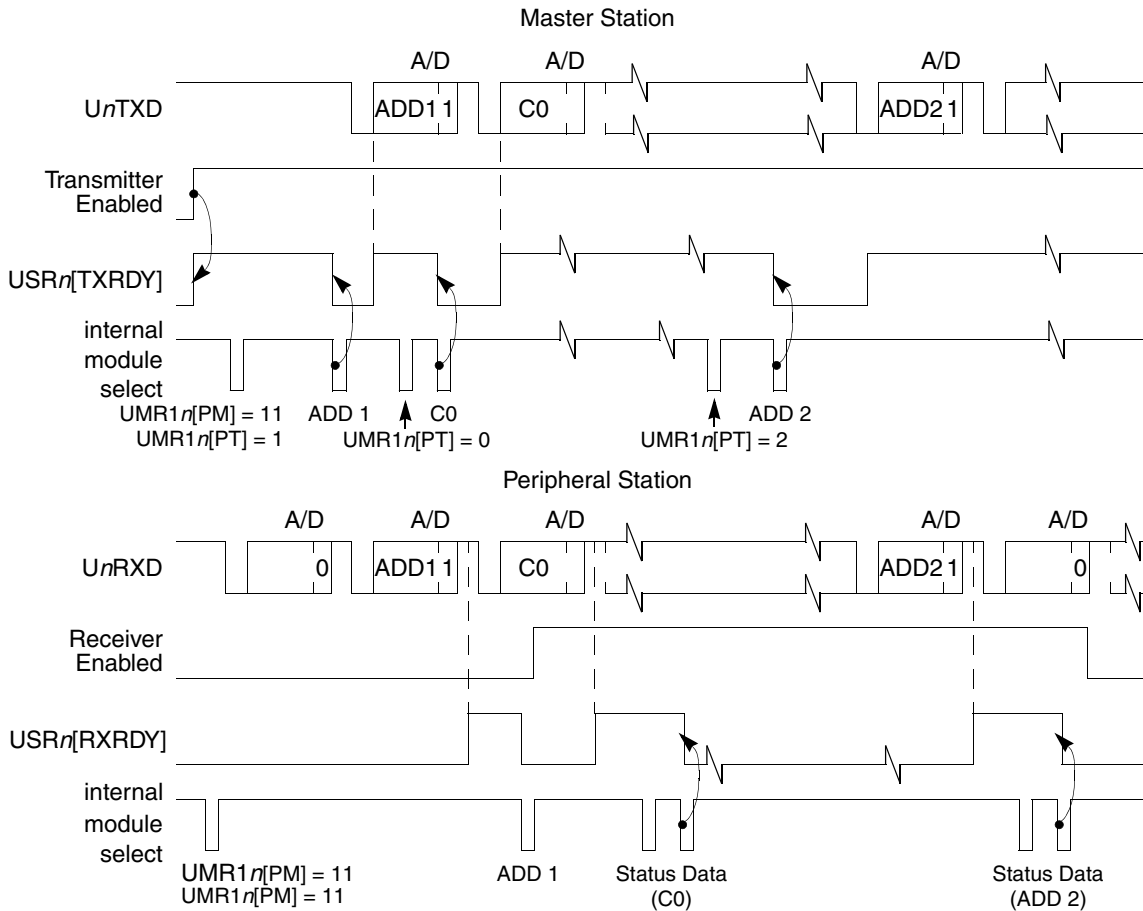
Figure 21-26. Multidrop Mode Timing Diagram

A character sent from the master station consists of a start bit, a programmed number of data bits, an address/data (A/D) bit flag, and a programmed number of stop bits. A/D = 1 indicates an address character; A/D = 0 indicates a data character. The polarity of A/D is selected through UMR1n[PT]. UMR1n should be programmed before enabling the transmitter and loading the corresponding data bits into the transmit buffer.

In multidrop mode, the receiver continuously monitors the received data stream, regardless of whether it is enabled or disabled. If the receiver is disabled, it sets the RXRDY bit and loads the character into the receiver holding register FIFO provided the received A/D bit is a one (address tag). The character is discarded if the received A/D bit is zero (data tag). If the receiver is enabled, all received characters are transferred to the CPU through the receiver holding register during read operations.

In either case, the data bits are loaded into the data portion of the FIFO while the A/D bit is loaded into the status portion of the FIFO normally used for a parity error (USRn[PE]).

Framing error, overrun error, and break detection operate normally. The A/D bit takes the place of the parity bit; therefore, parity is neither calculated nor checked. Messages in this mode may still contain error detection and correction information. One way to provide error detection, if 8-bit characters are not required, is to use software to calculate parity and append it to the 5-, 6-, or 7-bit character.

21.4.5 Bus Operation

This section describes bus operation during read, write, and interrupt acknowledge cycles to the UART module.

21.4.5.1 Read Cycles

The UART module responds to reads with byte data. Reserved registers return zeros.

21.4.5.2 Write Cycles

The UART module accepts write data as bytes only. Write cycles to read-only or reserved registers complete normally without exception processing, but data is ignored.

21.4.6 Programming

The software flowchart, Figure 21-27, consists of the following:

- UART module initialization—These routines consist of SINIT and CHCHK (See Sheet 1 p. 21-30 and Sheet 2 p. 21-31). Before SINIT is called at system initialization, the calling routine allocates 2 words on the system FIFO. On return to the calling routine, SINIT passes UART status data on the FIFO. If SINIT finds no errors, the transmitter and receiver are enabled. SINIT calls CHCHK to perform the checks. When called, SINIT places the UART in local loop-back mode and checks for the following errors:
  — Transmitter never ready
  — Receiver never ready
  — Parity error
  — Incorrect character received
- I/O driver routine—This routine (See Sheet 4 p. 21-33 and Sheet 5 p. 21-34) consists of INCH, the terminal input character routine which gets a character from the receiver, and OUTCH, which sends a character to the transmitter.
- Interrupt handling—Consists of SIRQ (See Sheet 4 p. 21-33), which is executed after the UART module generates an interrupt caused by a change-in-break (beginning of a break). SIRQ then clears the interrupt source, waits for the next change-in-break interrupt (end of break), clears the interrupt source again, then returns from exception processing to the system monitor.

21.4.6.1 Interrupt and DMA Request Initialization

21.4.6.1.1 Setting up the UART to Generate Core Interrupts

The list below gives the steps needed to properly initialize the UART to generate an interrupt request to the core.

1. Initialize ICRx register in the interrupt controller.
2. Unmask appropriate bits in IMR in the interrupt controller.
3. Unmask appropriate bits in the core's status register (SR) to enable interrupts.

4. If TXRDY or RXRDY are being used to generate interrupt requests, then verify that DMAREQC (in the SCM) does not also assign the UART's TXRDY and RXRDY into DMA channels

5. Initialize interrupts in the UART, see Table 21-14.

Table 21-14. UART Interrupts

Register	Bit	Interrupt
UMR1*n*	6	RxIRQ
UIMR*n*	7	Change of State (COS)
UIMR*n*	2	Delta Break
UIMR*n*	1	RxFIFO Full
UIMR*n*	0	TXRDY

21.4.6.1.2 Setting up the UART to Request DMA Service

The UART is capable of generating two different DMA request signals–transmit DMA requests and receive DMA requests.

The transmit DMA request signal is asserted when the TXRDY (transmitter ready) in the UART interrupt status register, UISR*n*[TXRDY], is set. When the transmit DMA request signal is asserted, the DMA can initiate a data copy, reading the next character to be transmitted from memory and writing it into the UART transmit buffer (UTB*n*). This would allow the DMA channel to stream data from memory to the UART for transmission without processor intervention. Once the entire message has been moved into the UART, the DMA would typically generate an end-of-data-transfer interrupt request to the CPU. The resulting interrupt service routine (ISR) could query the UART programming model to determine the end-of-transmission status.

Similarly, the receive DMA request signal is asserted when the FFULL/RXRDY (FIFO full or receive ready) flag in the interrupt status register, UISR*n*[FFULL/RXRDY], is set. When the receive DMA request signal is asserted, the DMA can initiate a data move, reading the appropriate characters from the UART receive buffer (URB*n*) and storing them in memory. This allows the DMA channel to stream data from the UART receive buffer into memory without processor intervention. Once the entire message has been moved from the UART, the DMA would typically generate an end-of-data-transfer interrupt request to the CPU. The resulting interrupt service routine (ISR) should query the UART programming model to determine the end-of-transmission status. In typical applications, the receive DMA request should be configured to use RXRDY directly (and not FFULL) to remove any complications related to retrieving the final characters from the FIFO buffer.

The implementation described in this section allows independent DMA processing of transmit and receive data while still supporting interrupt notification to the processor for CTS change-of-state and delta break error handling.

To configure the UART for DMA requests:

1. Initialize the DMAREQC in the SCM to map the desired UART DMA requests to the desired DMA channels. For example; setting DMAREQC[7:4] to 1000 maps UART0 receive DMA requests to DMA channel 1; setting DMAREQC[11:8] to 1101 maps UART1 transmit DMA requests to DMA channel 2; and so on. It is possible to independently map transmit based and receive based UART DMA requests in the DMAREQC.

2. Disable interrupts using the UIMR register. The appropriate UIMR bits must be cleared so that interrupt requests are disabled for those conditions for which a DMA request is desired. For example; to generate transmit DMA requests from UART1, then UIMR1[TXRDY] should be cleared. This will prevent TXRDY from generating an interrupt request while a transmit DMA request is generated.

3. Configure the GPACR and appropriate PACR registers located in the SCM for DMA access to IPSBAR space.

4. Initialize the DMA channel. The DMA should be configured for cycle steal mode and a source and destination size of one byte. This will cause a single byte to be transferred for each UART DMA request.

Table 21-15 shows the DMA requests.

Table 21-15. UART DMA Requests

Register	Bit	DMA Request
UISR*n*	1	Receive DMA request
UISR*n*	0	Transmit DMA request

21.4.6.2 UART Module Initialization Sequence

Table 21-16 shows the UART module initialization sequence.

Table 21-16. UART Module Initialization Sequence

Register	Setting
UCR*n*	Reset the receiver and transmitter. Reset the mode pointer (MISC[2–0] = 0b001).
UIMR*n*	Enable the desired interrupt sources.
UACR*n*	Initialize the input enable control (IEC bit).
UCSR*n*	Select the receiver and transmitter clock. Use timer as source if required.
UMR1*n*	If preferred, program operation of receiver ready-to-send (RXRTS bit). Select receiver-ready or FIFO-full notification (RXRDY/FFULL bit). Select character or block error mode (ERR bit). Select parity mode and type (PM and PT bits). Select number of bits per character (B/Cx bits).

Table 21-16. UART Module Initialization Sequence (continued)

Register	Setting
UMR2*n*	Select the mode of operation (CMx bits). If preferred, program operation of transmitter ready-to-send (TXRTS). If preferred, program operation of clear-to-send (TXCTS bit). Select stop-bit length (SBx bits).
UCR*n*	Enable transmitter and/or receiver.
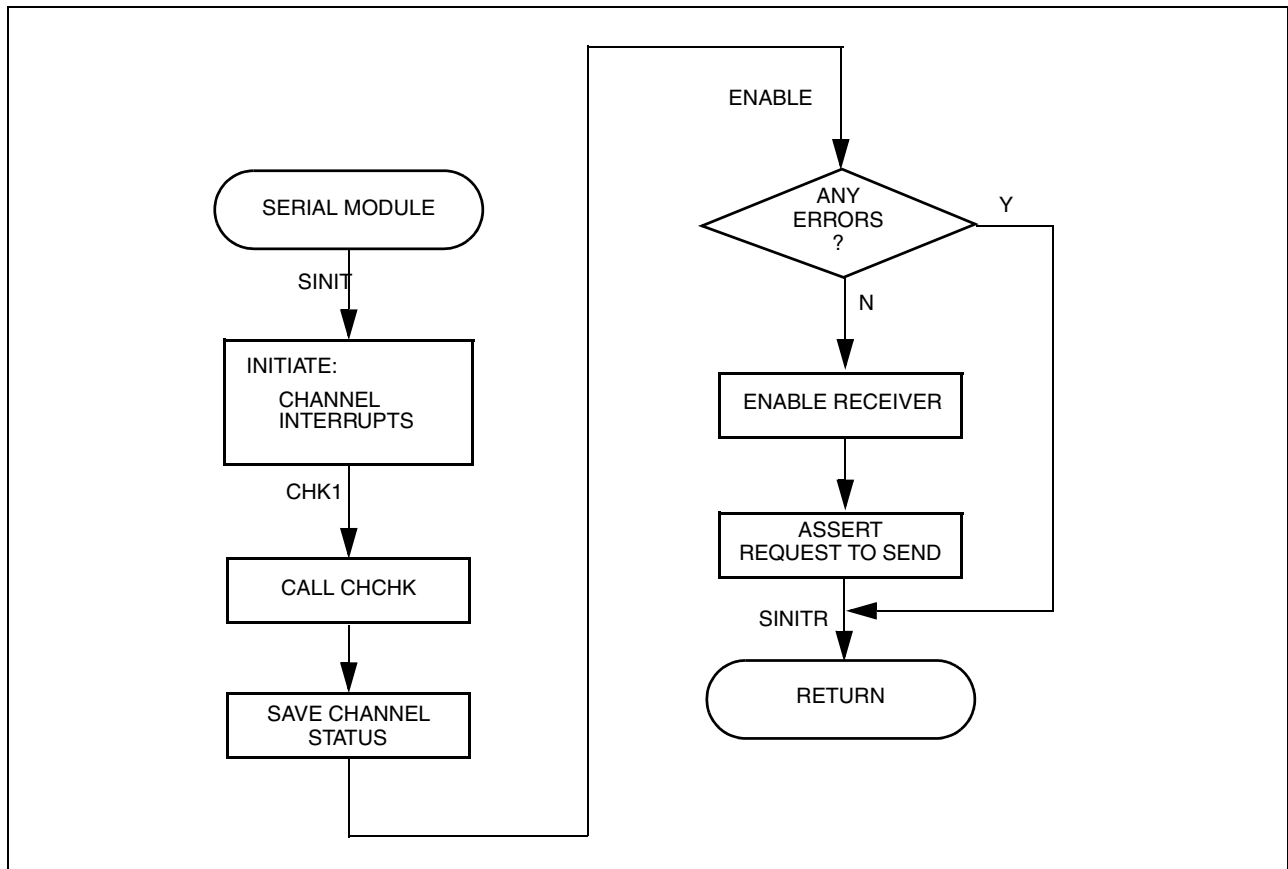
Figure 21-27. UART Mode Programming Flowchart (Sheet 1 of 5)
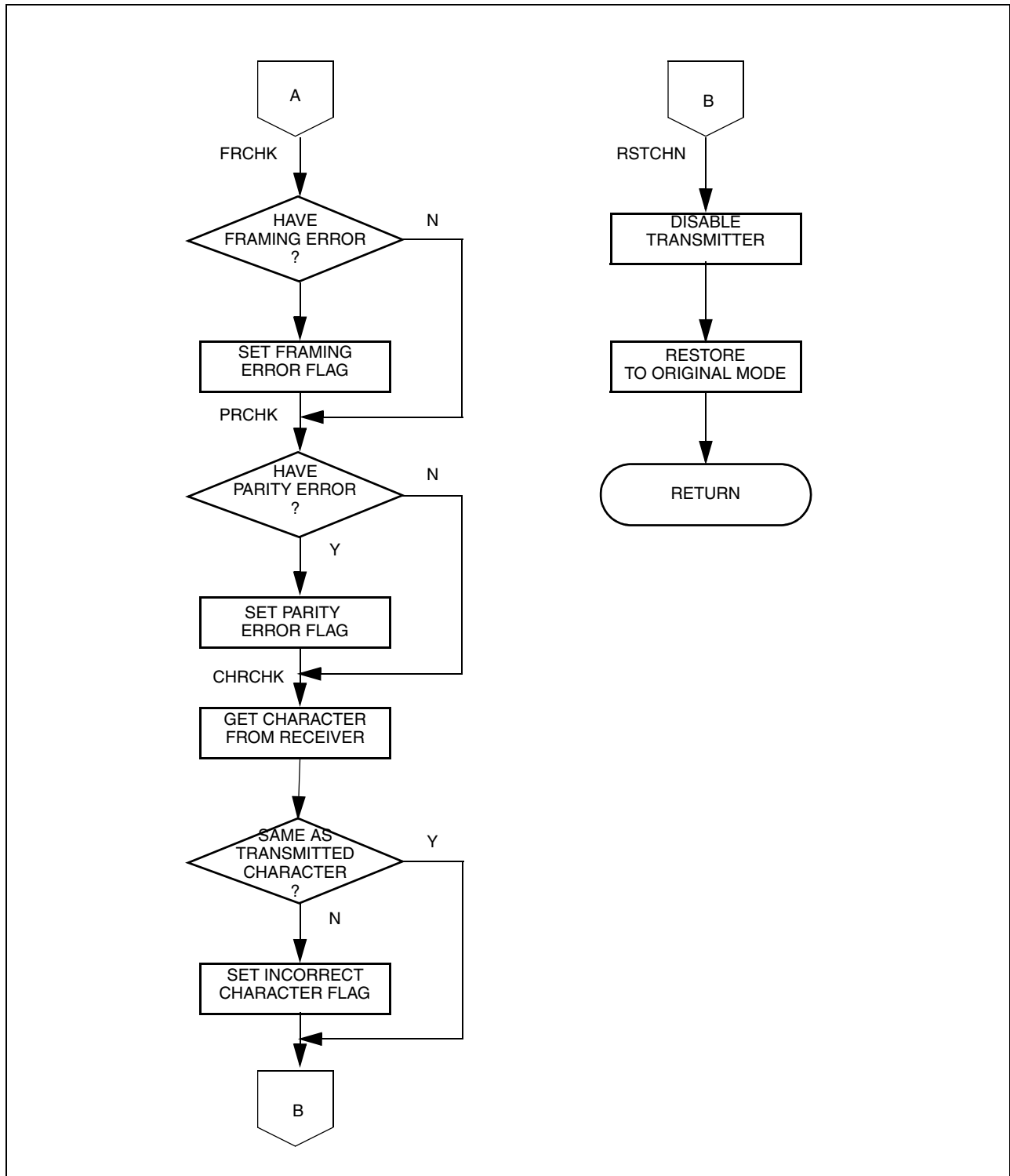
MCF5213 Reference Manual, Rev. 1.1

Preliminary

Figure 21-27. UART Mode Programming Flowchart (Sheet 2 of 5)

Figure 21-27. UART Mode Programming Flowchart (Sheet 3 of 5)

MCF5213 Reference Manual, Rev. 1.1

Preliminary
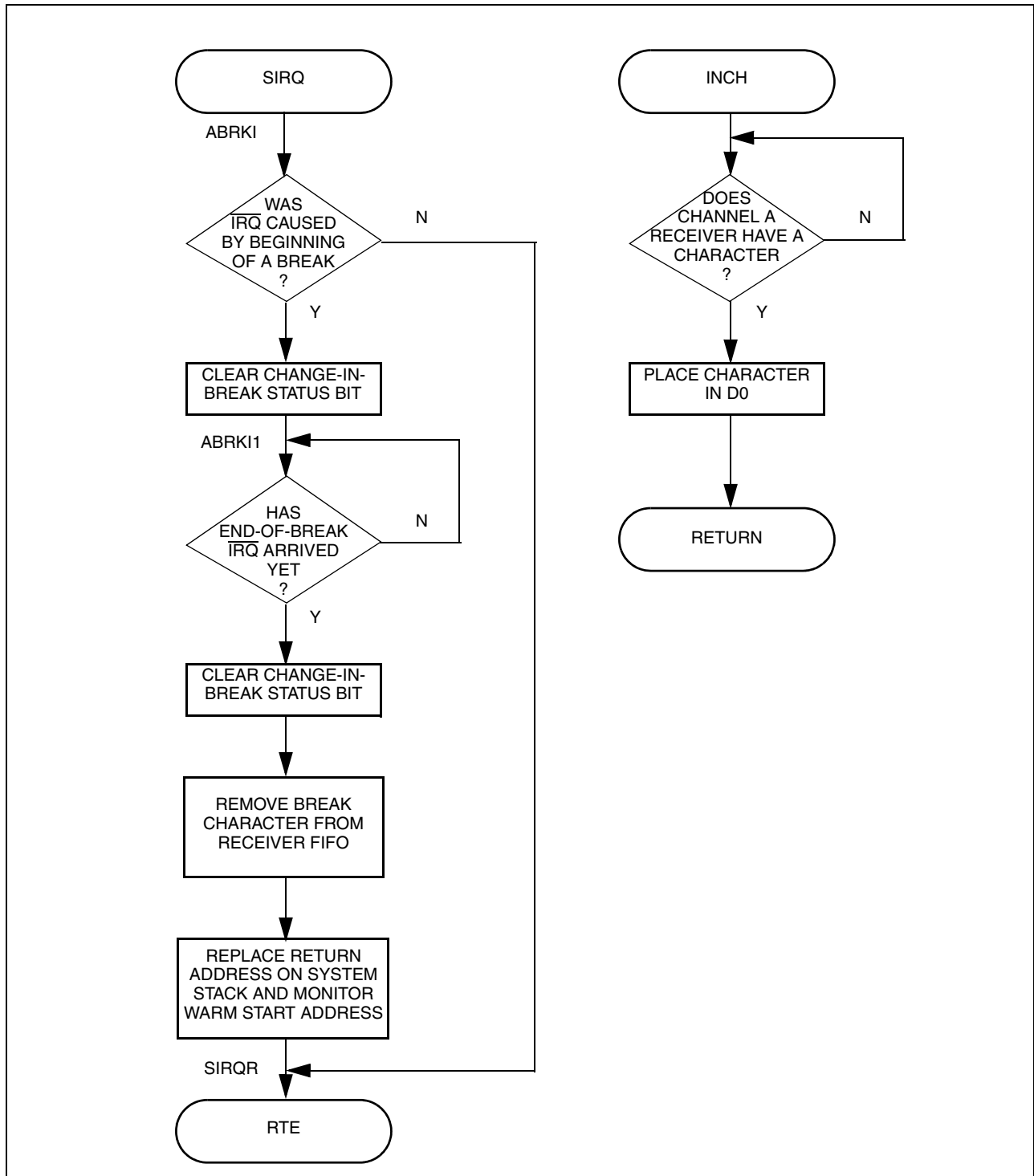
Figure 21-27. UART Mode Programming Flowchart (Sheet 4 of 5)
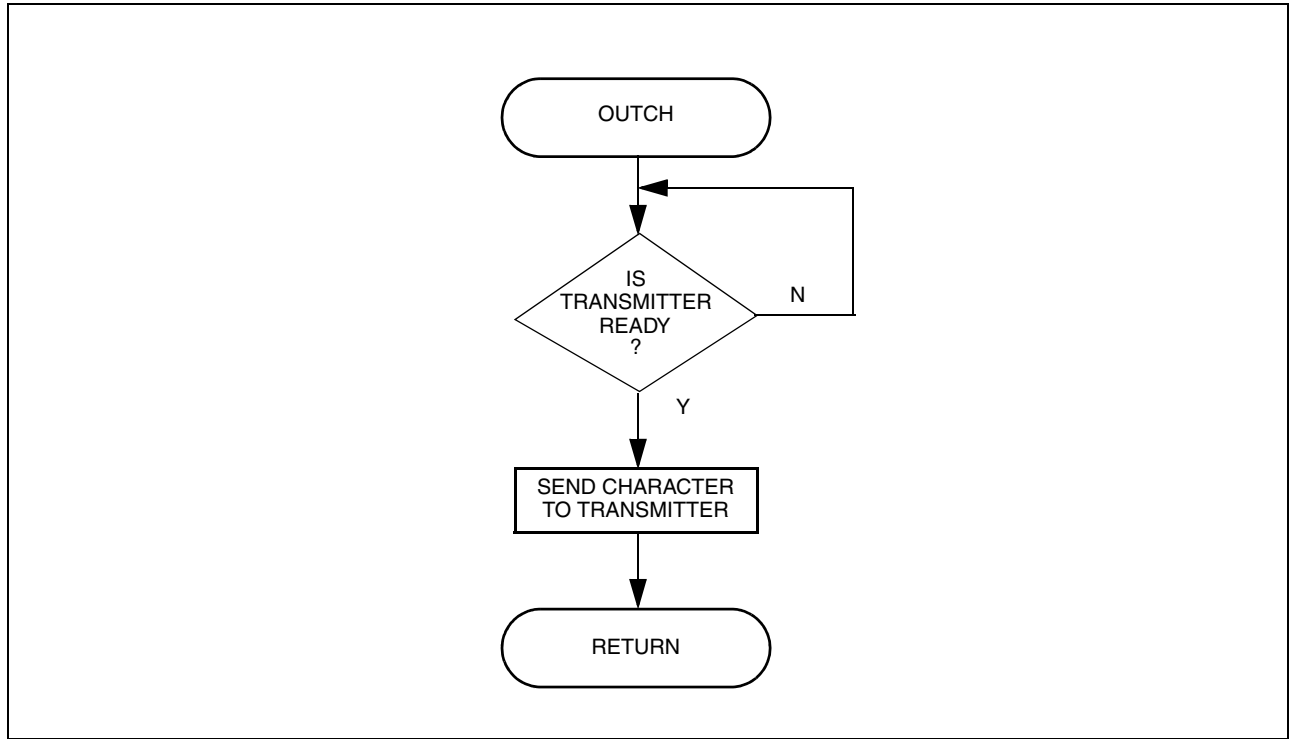
Figure 21-27. UART Mode Programming Flowchart (Sheet 5 of 5)

Chapter 22
I^{2}C Interface

22.1 Introduction

This chapter describes the I^{2}C module, clock synchronization, and I^{2}C programming model registers. It also provides extensive programming examples.

22.2 Overview

I^{2}C is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange, minimizing the interconnection between devices. This bus is suitable for applications that require occasional communication between many devices over a short distance. The flexible I^{2}C bus allows additional devices to be connected to the bus for expansion and system development.

The interface is designed to operate up to 100 Kbps with maximum bus loading and timing. The device is capable of operating at higher baud rates, up to a maximum of the internal bus clock divided by 20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF.

The I^{2}C system is a true multiple-master bus; it uses arbitration and collision detection to prevent data corruption in the event that multiple devices attempt to control the bus simultaneously. This feature supports complex applications with multiprocessor control and can be used for rapid testing and alignment of end products through external connections to an assembly-line computer.

NOTE
The I^{2}C module is designed to be compatible with the Philips I^{2}C bus protocol. For information on system configuration, protocol, and restrictions, see *The I^{2}C Bus Specification, Version 2.1*.

NOTE
The GPIO module must be configured to enable the peripheral function of the appropriate pins (refer to Chapter 13, "General Purpose I/O Module") prior to configuring the I^{2}C module.

22.3 Features

The I^{2}C module has the following key features:

- Compatibility with I^{2}C bus standard version 2.1
- Support for 3.3-V tolerant devices
- Multiple-master operation

- Software-programmable for one of 50 different serial clock frequencies
- Software-selectable acknowledge bit
- Interrupt-driven, byte-by-byte data transfer
- Arbitration-lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- START and STOP signal generation/detection
- Repeated START signal generation
- Acknowledge bit generation/detection
- Bus-busy detection

Figure 22-1 is a block diagram of the I²C module.
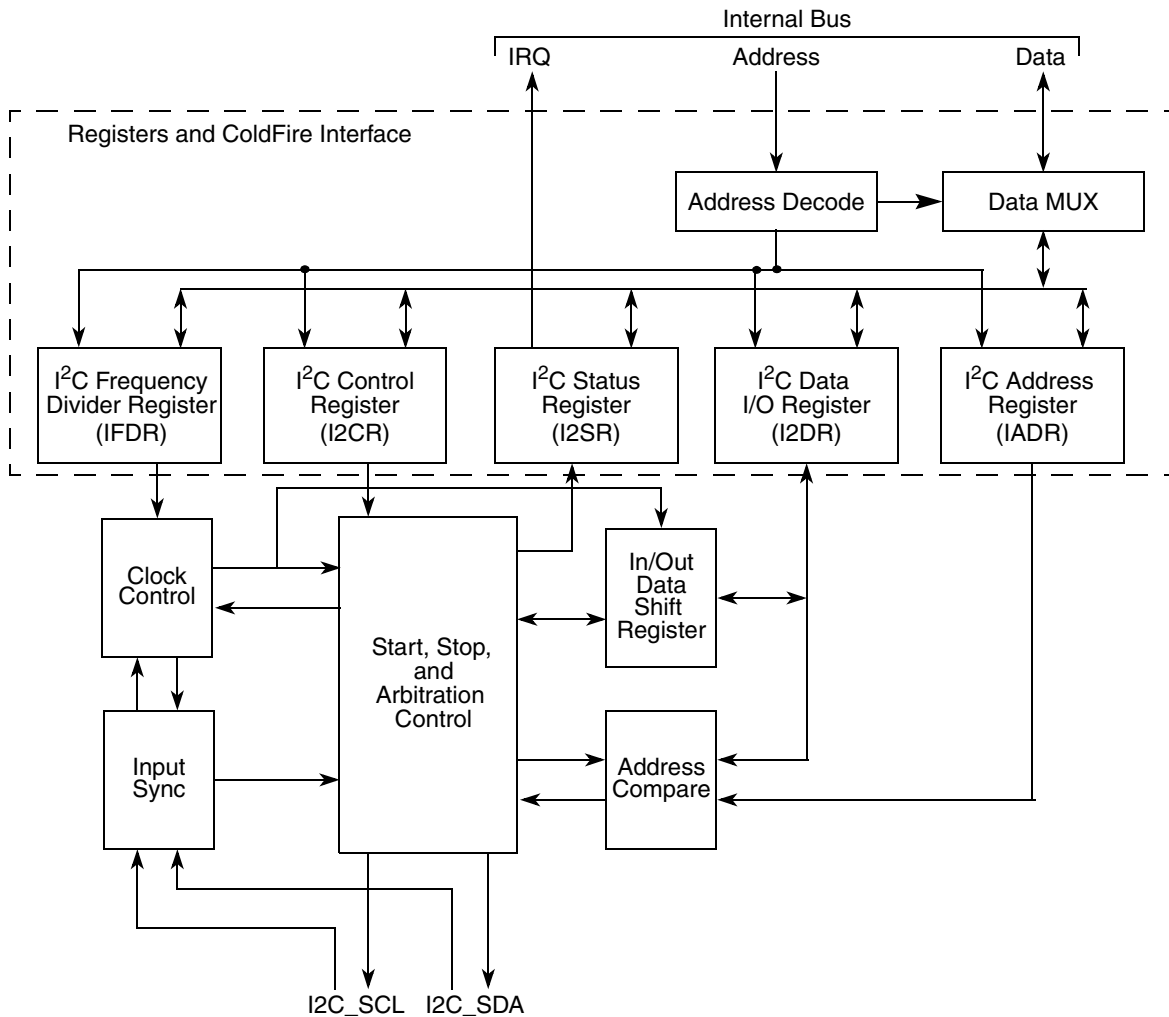
Figure 22-1. I²C Module Block Diagram

Figure 22-1 shows the I²C registers, which are described in Section 22.5, "Memory Map/Register Definition":

- I²C address register (I2ADR)
- I²C frequency divider register (I2FDR)
- I²C control register (I2CR)
- I²C status register (I2SR)
- I²C data I/O register (I2DR)

22.4 I²C System Configuration

The I²C module uses a serial data line (I2C_SDA) and a serial clock line (I2C_SCL) for data transfer. For I²C compliance, all devices connected to these two signals must have open drain or open collector outputs. The logic AND function is exercised on both lines with external pull-up resistors.

Out of reset, the I²C default state is as a slave receiver. Thus, when not programmed to be a master or responding to a slave transmit address, the I²C module should return to the default slave receiver state. See Section 22.6.1, "Initialization Sequence," for exceptions.

Normally, a standard communication is composed of four parts: START signal, slave address transmission, data transfer, and STOP signal. These are discussed in the following sections.

22.4.1 START Signal

When no other device is bus master (both I2C_SCL and I2C_SDA lines are at logic high), a device can initiate communication by sending a START signal (see A in Figure 22-2). A START signal is defined as a high-to-low transition of I2C_SDA while I2C_SCL is high. This signal denotes the beginning of a data transfer (each data transfer can be several bytes long) and awakens all slaves.
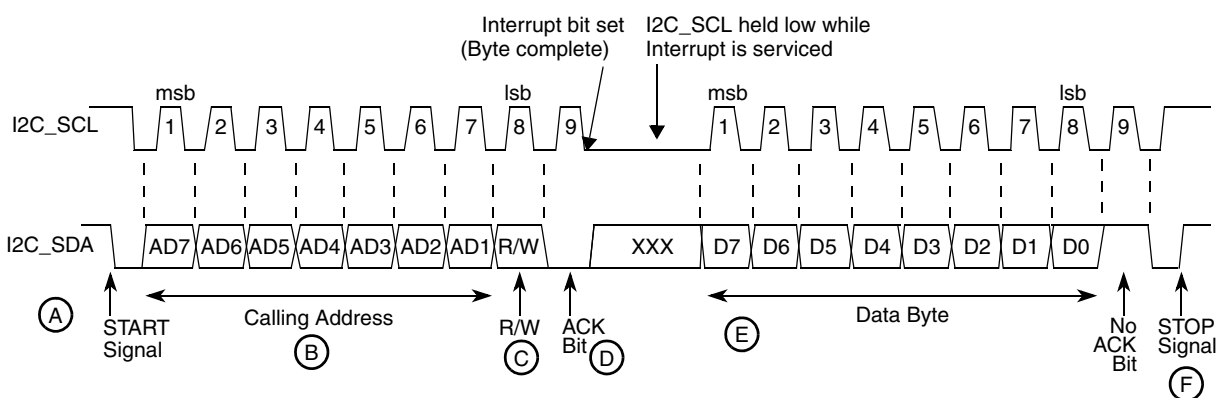
Figure 22-2. I²C Standard Communication Protocol

22.4.2 Slave Address Transmission

The master sends the slave address in the first byte after the START signal (B). After the seven-bit calling address, it sends the R/W bit (C), which tells the slave data transfer direction (0 = write transfer, 1 = read transfer).

Each slave must have a unique address. An I²C master must not transmit its own slave address; it cannot be master and slave at the same time.

The slave whose address matches that sent by the master pulls I2C_SDA low at the ninth serial clock (D) to return an acknowledge bit.

22.4.3 Data Transfer

When successful slave addressing is achieved, the data transfer can proceed (see E in Figure 22-2) on a byte-by-byte basis in the direction specified by the R/W bit sent by the calling master.

Data can be changed only while I2C_SCL is low and must be held stable while I2C_SCL is high, as Figure 22-2 shows. I2C_SCL is pulsed once for each data bit, with the msb being sent first. The receiving device must acknowledge each byte by pulling I2C_SDA low at the ninth clock; therefore, a data byte transfer takes nine clock pulses. See Figure 22-3.
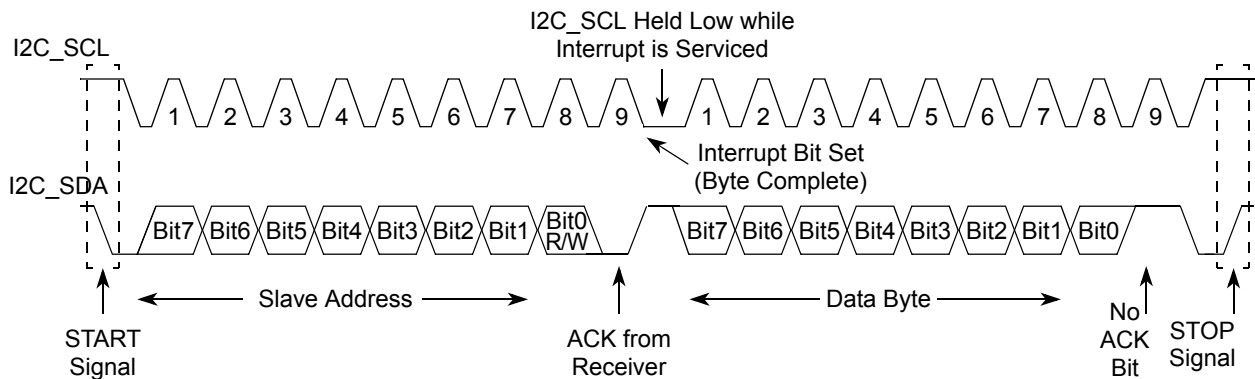
Figure 22-3. Data Transfer

22.4.4 Acknowledge

The transmitter releases the I2C_SDA line high during the acknowledge clock pulse as shown in Figure 22-4. The receiver pulls down the I2C_SDA line during the acknowledge clock pulse so that it remains stable low during the high period of the clock pulse.

If it does not acknowledge the master, the slave receiver must leave I2C_SDA high. The master can then generate a STOP signal to abort the data transfer or generate a START signal (repeated start, shown in Figure 22-5 and discussed in Section 22.4.6, "Repeated START") to start a new calling sequence.
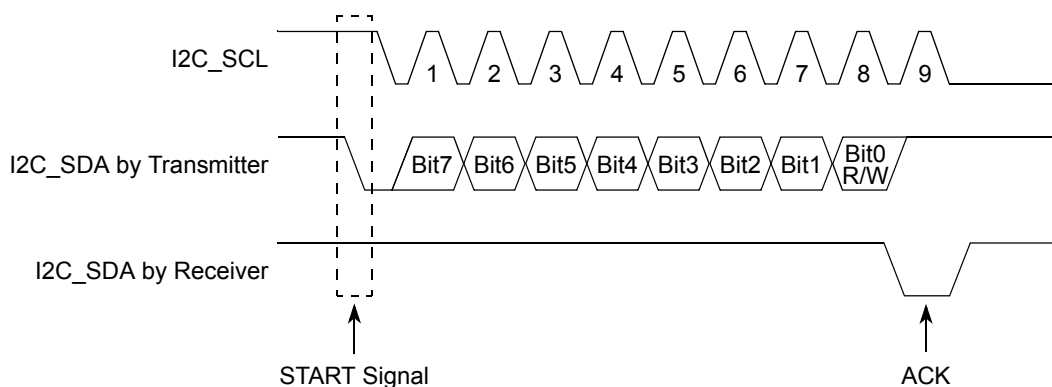
Figure 22-4. Acknowledgement by Receiver

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it means end-of-data to the slave. The slave releases I2C_SDA for the master to generate a STOP or START signal (Figure 22-4).

22.4.5 STOP Signal

The master can terminate communication by generating a STOP signal to free the bus. A STOP signal is defined as a low-to-high transition of I2C_SDA while I2C_SCL is at logical high (see F in Figure 22-2). The master can generate a STOP even if the slave has generated an acknowledgment, at which point the slave must release the bus. The master may also generate a START signal following a calling address, without first generating a STOP signal. Refer to Section 22.4.6, "Repeated START."

22.4.6 Repeated START

A repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication. The master uses a repeated START to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.
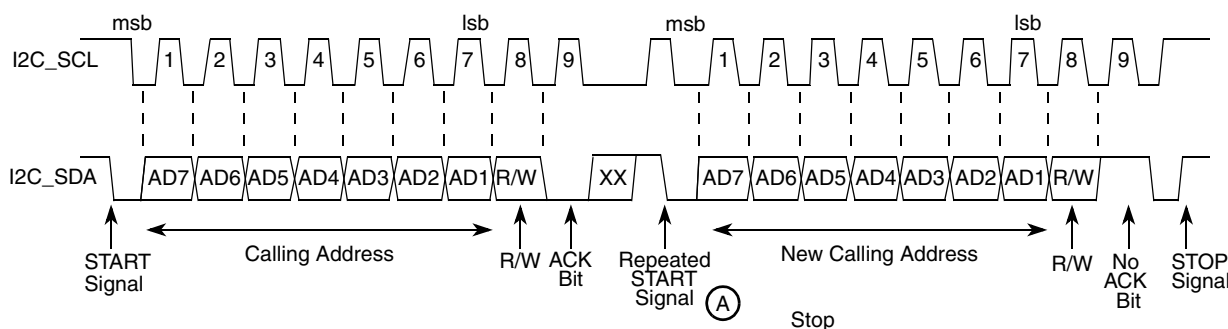
Figure 22-5. Repeated START

Various combinations of read/write formats are then possible:

- The first example in Figure 22-6 is the case of master-transmitter transmitting to slave-receiver. The transfer direction is not changed.

- The second example in Figure 22-6 is the master reading the slave immediately after the first byte. At the moment of the first acknowledge, the master-transmitter becomes a master-receiver and the slave-receiver becomes slave-transmitter.

- In the third example in Figure 22-6, the START condition and slave address are both repeated using the repeated START signal. This is to communicate with same slave in a different mode without releasing the bus. The master transmits data to the slave first, and then the master reads data from slave by reversing the R/W bit.

ST = Start
SP = Stop
A = Acknowledge (I2C_SDA low)
A = Not Acknowledge (I2C_SDA high)
Rept ST = Repeated Start

☐ From Master to Slave

▨ From Slave to Master

Example 1:

ST	7bit Slave Address	0	A	Register Address	A	DATA	A/Ā	SP

R/W̄

Example 2:

ST	7bit Slave Address	1	A	DATA	A	DATA	Ā	SP

R/W̄

Example 3:

ST	7-bit Slave Address	1	A	DATA	A/Ā	Rept ST	7-bit Slave Address	0	A	DATA	A	DATA	A/Ā	SP

R/W̄   R/W̄

Master Transmits to Slave   Master Reads from Slave
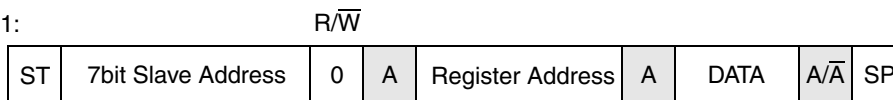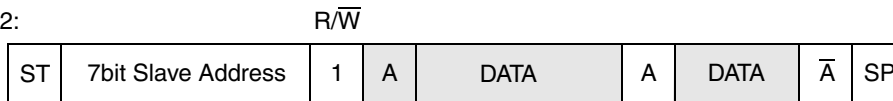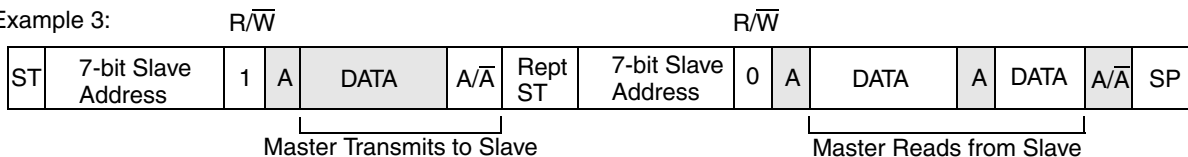
Figure 22-6. Data Transfer, Combined Format

[1] Note: No acknowledge on the last byte

22.4.7 Clock Synchronization and Arbitration

I²C is a true multi-master bus that allows more than one master to be connected to it. If two or more master devices simultaneously request control of the bus, a clock synchronization procedure determines the bus clock. Because wire-AND logic is performed on the I2C_SCL line, a high-to-low transition on the I2C_SCL line affects all the devices connected on the bus. The devices start counting their low period and once a device's clock has gone low, it holds the I2C_SCL line low until the clock high state is reached. However, the change of low to high in this device's clock may not change the state of the I2C_SCL line if another device clock is still within its low period. Therefore, synchronized clock I2C_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 22-8). When all devices concerned have counted off their low period, the synchronized clock (I2C_SCL) line is released and pulled high. At this point, the device clocks and the I2C_SCL line are synchronized, and the devices start counting their high periods. The first device to complete its high period pulls the I2C_SCL line low again.

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 I2C_SDA output (see Figure 22-7). In this case the transition from master to slave mode does not generate a STOP condition. Meanwhile, hardware sets I2SR[IAL] to indicate loss of arbitration.
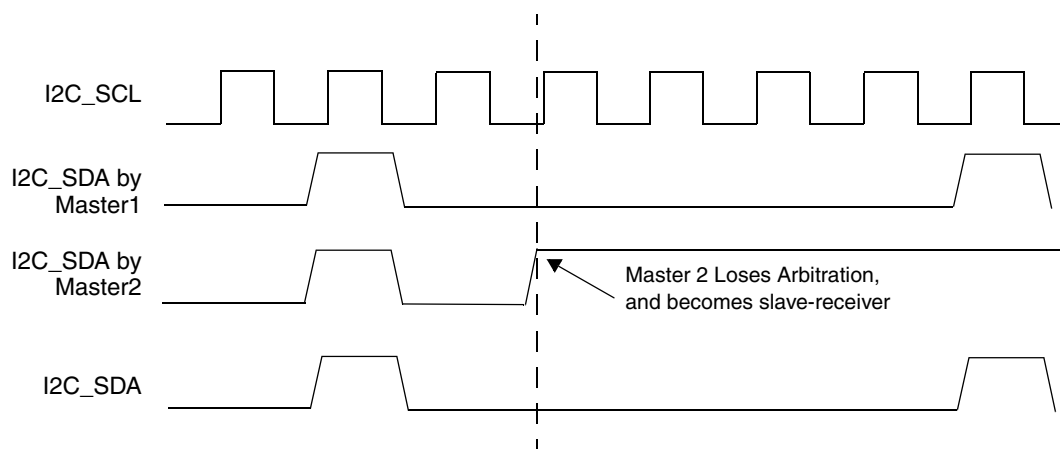
Figure 22-7. Arbitration Procedure
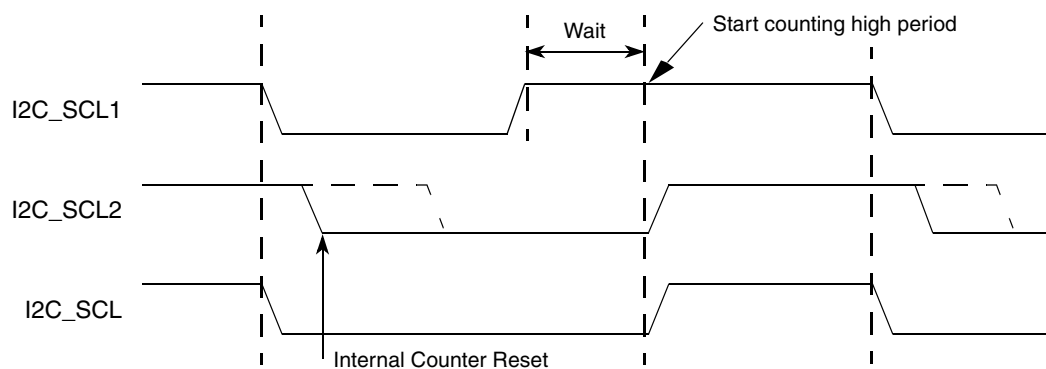
Figure 22-8. Clock Synchronization

22.4.8   Handshaking and Clock Stretching

The clock synchronization mechanism can be used as a handshake in data transfers. Slave devices can hold I2C_SCL low after completing one byte transfer. In such a case, the clock mechanism halts the bus clock and forces the master clock into wait states until the slave releases I2C_SCL.

Slaves may also slow down the transfer bit rate. After the master has driven I2C_SCL low, the slave can drive I2C_SCL low for the required period and then release it. If the slave I2C_SCL low period is longer than the master I2C_SCL low period, the resulting I2C_SCL bus signal low period is stretched.

22.5 Memory Map/Register Definition

Table 22-1 lists the configuration registers used in the I²C interface.

Table 22-1. I²C Module Memory Map

IPSBAR Offset	Register	Access	Reset Value	Section/Page
0x0300	I²C Address Register (I2ADR)	R/W	0x00	22.5.1/22-8
0x0304	I²C Frequency Divider Register (I2FDR)	R/W	0x00	22.5.2/22-9
0x0308	I²C Control Register (I2CR)	R/W	0x00	22.5.3/22-10
0x030C	I²C Status Register (I2SR)	R/W	0x81	22.5.4/22-11
0x0310	I²C Data I/O Register (I2DR)	R/W	0x00	22.5.5/22-12

22.5.1 I²C Address Register (I2ADR)

The I2ADR holds the address the I²C responds to when addressed as a slave. Note that it is not the address sent on the bus during the address transfer when the module is performing a master transfer.

Address: 0x0300 (I2ADR)   Access: User read/write

	7	6	5	4	3	2	1	0
R				ADR				0
W								
Reset:	0	0	0	0	0	0	0	0
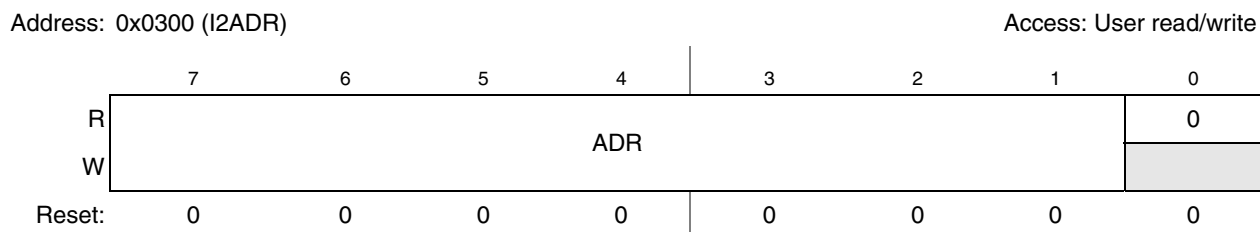
Figure 22-9. I²C Address Register (I2ADR)

Table 22-2. I2ADR Field Descriptions

Field	Description
7–1 ADR	Slave address. Contains the specific slave address to be used by the I²C module. Slave mode is the default I²C mode for an address match on the bus.
0	Reserved, should be cleared.

22.5.2 I²C Frequency Divider Register (I2FDR)

The I2FDR, shown in Figure 22-10, provides a programmable prescaler to configure the I²C clock for bit-rate selection.

Address: 0x0304 (I2FDR) Access: User read/write

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

Figure 22-10. I²C Frequency Divider Register (I2FDR)

Table 22-3. I2FDR Field Descriptions

Field	Description
7–6	Reserved, should be cleared.
5–0 IC	I²C clock rate. Prescales the clock for bit-rate selection. The serial bit clock frequency is equal to the internal bus clock divided by the divider shown below. Due to potentially slow I2C_SCL and I2C_SDA rise and fall times, bus signals are sampled at the prescaler frequency.

IC	Divider	IC	Divider	IC	Divider	IC	Divider
0x00	28	0x10	288	0x20	20	0x30	160
0x01	30	0x11	320	0x21	22	0x31	192
0x02	34	0x12	384	0x22	24	0x32	224
0x03	40	0x13	480	0x23	26	0x33	256
0x04	44	0x14	576	0x24	28	0x34	320
0x05	48	0x15	640	0x25	32	0x35	384
0x06	56	0x16	768	0x26	36	0x36	448
0x07	68	0x17	960	0x27	40	0x37	512
0x08	80	0x18	1152	0x28	48	0x38	640
0x09	88	0x19	1280	0x29	56	0x39	768
0x0A	104	0x1A	1536	0x2A	64	0x3A	896
0x0B	128	0x1B	1920	0x2B	72	0x3B	1024
0x0C	144	0x1C	2304	0x2C	80	0x3C	1280
0x0D	160	0x1D	2560	0x2D	96	0x3D	1536
0x0E	192	0x1E	3072	0x2E	112	0x3E	1792
0x0F	240	0x1F	3840	0x2F	128	0x3F	2048

MCF5213 Reference Manual, Rev. 1.1

Preliminary

22.5.3 I²C Control Register (I2CR)

The I2CR is used to enable the I²C module and the I²C interrupt. It also contains bits that govern operation as a slave or a master.

Address: 0x0308 (I2CR)   Access: User read/write

	7	6	5	4	3	2	1	0
R	IEN	IIEN	MSTA	MTX	TXAK	RSTA	0	0
W								
Reset:	0	0	0	0	0	0	0	0
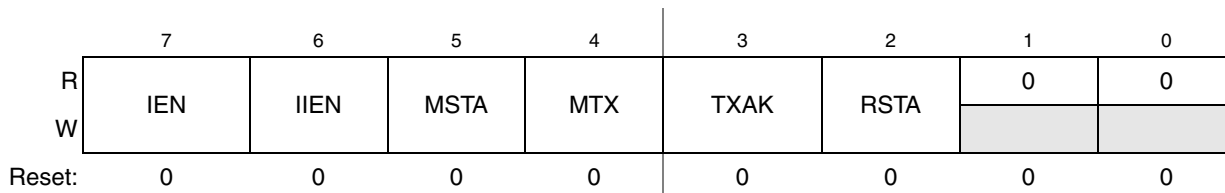
Figure 22-11. I²C Control Register (I2CR)

Table 22-4. I2CR Field Descriptions

Field	Description
7 IEN	I²C enable. Controls the software reset of the entire I²C module. If the module is enabled in the middle of a byte transfer, slave mode ignores the current bus transfer and starts operating when the next START condition is detected. Master mode is not aware that the bus is busy; so initiating a start cycle may corrupt the current bus cycle, ultimately causing either the current master or the I²C module to lose arbitration, after which bus operation returns to normal. 0 The I²C module is disabled, but registers can still be accessed. 1 The I²C module is enabled. This bit must be set before any other I2CR bits have any effect.
6 IIEN	I²C interrupt enable. 0 I²C module interrupts are disabled, but currently pending interrupt condition is not cleared. 1 I²C module interrupts are enabled. An I²C interrupt occurs if I2SR[IIF] is also set.
5 MSTA	Master/slave mode select bit. If the master loses arbitration, MSTA is cleared without generating a STOP signal. 0 Slave mode. Changing MSTA from 1 to 0 generates a STOP and selects slave mode. 1 Master mode. Changing MSTA from 0 to 1 signals a START on the bus and selects master mode.
4 MTX	Transmit/receive mode select bit. Selects the direction of master and slave transfers. 0 Receive 1 Transmit. When the device is addressed as a slave, software should set MTX according to I2SR[SRW]. In master mode, MTX should be set according to the type of transfer required. Therefore, when the MCU addresses a slave device, MTX is always 1.
3 TXAK	Transmit acknowledge enable. Specifies the value driven onto I2C_SDA during acknowledge cycles for both master and slave receivers. Note that writing TXAK applies only when the I²C bus is a receiver. 0 An acknowledge signal is sent to the bus at the ninth clock bit after receiving one byte of data. 1 No acknowledge signal response is sent (that is, acknowledge bit = 1).
2 RSTA	Repeat start. Always read as 0. Attempting a repeat start without bus mastership causes loss of arbitration. 0 No repeat start 1 Generates a repeated START condition.
1–0	Reserved, should be cleared.

22.5.4 I²C Status Register (I2SR)

This I2SR contains bits that indicate transaction direction and status.

Address: 0x030C (I2SR)   Access: User read/write

	7	6	5	4	3	2	1	0
R	ICF	IAAS	IBB	IAL	0	SRW	IIF	RXAK
W								
Reset:	1	0	0	0	0	0	0	1
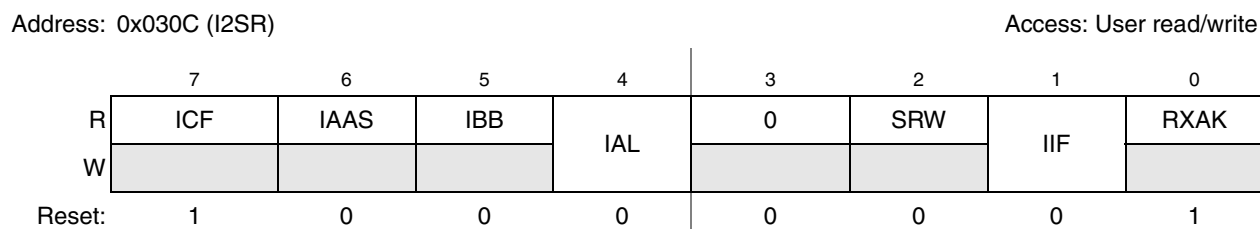
Figure 22-12. I²C Status Register (I2SR)

Table 22-5. I2SR Field Descriptions

Field	Description
7 ICF	I²C Data transferring bit. While one byte of data is transferred, ICF is cleared. 0 Transfer in progress 1 Transfer complete. Set by the falling edge of the ninth clock of a byte transfer.
6 IAAS	I²C addressed as a slave bit. The CPU is interrupted if I2CR[IIEN] is set. Next, the CPU must check SRW and set its TX/RX mode accordingly. Writing to I2CR clears this bit. 0 Not addressed. 1 Addressed as a slave. Set when its own address (IADR) matches the calling address.
5 IBB	I²C bus busy bit. Indicates the status of the bus. 0 Bus is idle. If a STOP signal is detected, IBB is cleared. 1 Bus is busy. When START is detected, IBB is set.
4 IAL	I²C arbitration lost. Set by hardware in the following circumstances. (IAL must be cleared by software by writing zero to it.) • I2C_SDA sampled low when the master drives high during an address or data-transmit cycle. • I2C_SDA sampled low when the master drives 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.
3	Reserved, should be cleared.
2 SRW	Slave read/write. When IAAS is set, SRW indicates the value of the R/W command bit of the calling address sent from the master. SRW is valid only when a complete transfer has occurred, no other transfers have been initiated, and the I²C module is a slave and has an address match. 0 Slave receive, master writing to slave. 1 Slave transmit, master reading from slave.
1 IIF	I²C interrupt. Must be cleared by software by writing a zero in the interrupt routine. 0 No I²C interrupt pending 1 An interrupt is pending, which causes a processor interrupt request (if IIEN = 1). Set when one of the following occurs: • Complete one byte transfer (set at the falling edge of the ninth clock) • Reception of a calling address that matches its own specific address in slave-receive mode • Arbitration lost
0 RXAK	Received acknowledge. The value of I2C_SDA during the acknowledge bit of a bus cycle. 0 An acknowledge signal was received after the completion of 8-bit data transmission on the bus 1 No acknowledge signal was detected at the ninth clock.

22.5.5   I²C Data I/O Register (I2DR)

In master-receive mode, reading the I2DR allows a read to occur and for the next data byte to be received. In slave mode, the same function is available once the I²C has received its slave address.

Address: 0x0310 (I2DR)   Access: User read/write

	7	6	5	4	3	2	1	0
R				DATA				
W								
Reset:	0	0	0	0	0	0	0	0
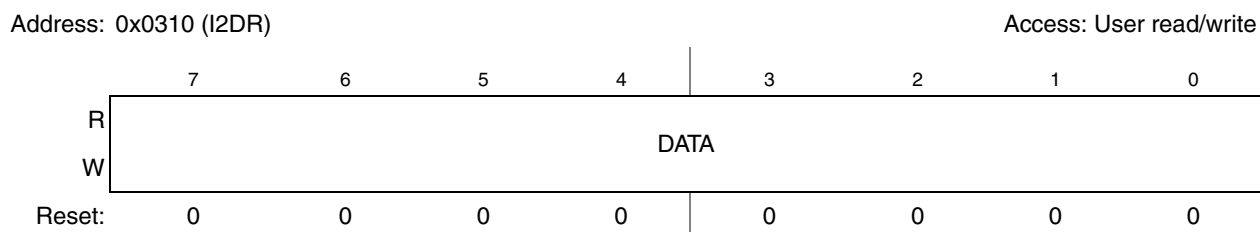
Figure 22-13. I²C Data I/O Register (I2DR)

Table 22-6. I2DR Field Description

Field	Description
7–0 DATA	I²C data. In master transmit mode, when data is written to this register, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates the reception of the next byte of data. In slave mode, the same functions are available after an address match has occurred. **Note:** In master transmit mode, the first byte of data written to I2DR following assertion of I2CR[MSTA] is used for the address transfer and should comprise the calling address (in position D7–D1) concatenated with the required R/W bit (in position D0). This bit (D0) is not automatically appended by the hardware, software must provide the appropriate R/W bit. **Note:** I2CR[MSTA] generates a start when a master does not already own the bus. I2CR[RSTA] generates a start (restart) without the master first issuing a stop (i.e., the master already owns the bus). In order to start the read of data, a dummy read to this register starts the read process from the slave. The next read of the I2DR register contains the actual data.

22.6   I²C Programming Examples

The following examples show programming for initialization, signaling START, post-transfer software response, signaling STOP, and generating a repeated START.

22.6.1   Initialization Sequence

Before the interface can transfer serial data, registers must be initialized, as follows:

1. Set I2FDR[IC] to obtain I2C_SCL frequency from the system bus clock. See
2. Update the I2ADR to define its slave address.
3. Set I2CR[IEN] to enable the I²C bus interface system.
4. Modify the I2CR to select or deselect master/slave mode, transmit/receive mode, and interrupt-enable or not.

NOTE

If I2SR[IBB] is set when the I²C bus module is enabled, execute the following pseudocode sequence before proceeding with normal initialization code. This issues a STOP command to the slave device, placing it in idle state as if it were just power-cycled on.

```
I2CR = 0x0
I2CR = 0xA0
dummy read of I2DR
I2SR = 0x0
I2CR = 0x0
```

22.6.2   Generation of START

After completion of the initialization procedure, serial data can be transmitted by selecting the master transmitter mode. On a multiple-master bus system, I2SR[IBB] must be tested to determine whether the serial bus is free. If the bus is free (IBB = 0), the START signal and the first byte (the slave address) can be sent. The data written to the data register comprises the address of the desired slave and the lsb indicates the transfer direction.

The free time between a STOP and the next START condition is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the I2C_SCL period, it may be necessary to wait until the I2C is busy after writing the calling address to the I2DR before proceeding with the following instructions.

The following example signals START and transmits the first byte of data (slave address):

```
CHFLAG   MOVE.B I2SR,-(A0)            ;Check I2SR[MBB]
         BTST.B #5, (A0)+

         BNE.S CHFLAG                 ;If I2SR[MBB] = 1, wait until it is clear
TXSTART  MOVE.B I2CR,-(A0)            ;Set transmit mode
         BSET.B #4,(A0)
         MOVE.B (A0)+, I2CR
         MOVE.B I2CR, -(A0)           ;Set master mode
         BSET.B #5, (A0)              ;Generate START condition
         MOVE.B (A0)+, I2CR
         MOVE.B CALLING,-(A0)         ;Transmit the calling address, D0=R/W
         MOVE.B (A0)+, I2DR
IFREE    MOVE.B I2SR,-(A0)            ;Check I2SR[MBB]
                                      ;If it is clear, wait until it is set.
         BTST.B #5, (A0)+
         BEQ.S IFREE;
```

22.6.3   Post-Transfer Software Response

Sending or receiving a byte sets the I2SR[ICF], which indicates one byte communication is finished. I2SR[IIF] is also set. An interrupt is generated if the interrupt function is enabled during initialization by setting I2CR[IIEN]. Software must first clear I2SR[IIF] in the interrupt routine. I2SR[ICF] is cleared either by reading from I2DR in receive mode or by writing to I2DR in transmit mode.

Software can service the I2C I/O in the main program by monitoring the IIF bit if the interrupt function is disabled. Polling should monitor IIF rather than ICF because that operation is different when arbitration is lost.

When an interrupt occurs at the end of the address cycle, the master is always in transmit mode; that is, the address is sent. If master receive mode is required, I2CR[MTX] should be toggled.

During slave-mode address cycles (I2SR[IAAS] = 1), I2SR[SRW] is read to determine the direction of the next transfer. MTX is programmed accordingly. For slave-mode data cycles (IAAS = 0), SRW is invalid. MTX should be read to determine the current transfer direction.

The following is an example of a software response by a master transmitter in the interrupt routine (see Figure 22-14).

```
I2SR     LEA.L I2SR,-(A7)        ;Load effective address
         BCLR.B #1,(A7)+         ;Clear the IIF flag
         MOVE.B I2CR,-(A7)       ;Push the address on stack,
         BTST.B #5,(A7)+         ;check the MSTA flag
         BEQ.S SLAVE             ;Branch if slave mode
         MOVE.B I2CR,-(A7)       ;Push the address on stack
         BTST.B #4,(A7)+         ;check the mode flag
         BEQ.S RECEIVE           ;Branch if in receive mode
         MOVE.B I2SR,-(A7)       ;Push the address on stack,
         BTST.B #0,(A7)+         ;check ACK from receiver
         BNE.B END               ;If no ACK, end of transmission
TRANSMIT MOVE.B DATABUF,-(A7)    ;Stack data byte
         MOVE.B (A7)+, I2DR      ;Transmit next byte of data
```

22.6.4 Generation of STOP

A data transfer ends when the master signals a STOP, which can occur after all data is sent, as in the following example.

```
MASTX    MOVE.B I2SR, -(A7)      ;If no ACK, branch to end
         BTST.B #0,(A7)+
         BNE.B END
         MOVE.B TXCNT,D0         ;Get value from the transmitting counter
         BEQ.S END               ;If no more data, branch to end
         MOVE.B DATABUF,-(A7)    ;Transmit next byte of data
         MOVE.B (A7)+,I2DR
         MOVE.B TXCNT,D0         ;Decrease the TXCNT
         SUBQ.L #1,D0
         MOVE.B D0,TXCNT
         BRA.S EMASTX;Exit
END      LEA.L I2CR,-(A7)        ;Generate a STOP condition
         BCLR.B #5,(A7)+
         EMASTX RTE              ;Return from interrupt
```

For a master receiver to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last data byte. This is done by setting I2CR[TXAK] before reading the next-to-last byte. Before the last byte is read, a STOP signal must be generated, as in the following example.

```
MASR     MOVE.B RXCNT,D0         ;Decrease RXCNT
         SUBQ.L #1,D0
         MOVE.B D0,RXCNT
         BEQ.S ENMASR            ;Last byte to be read
```

```
            MOVE.B  RXCNT,D1              ;Check second-to-last byte to be read
            EXTB.L  D1
            SUBI.L  #1,D1;
            BNE.S   NXMAR                 ;Not last one or second last
            LAMAR BSET.B #3,I2CR          ;Disable ACK
            BRA     NXMAR
ENMASR      BCLR.B  #5,I2CR               ;Last one, generate STOP signal
NXMAR       MOVE.B  I2DR,RXBUF            ;Read data and store RTE
```

22.6.5   Generation of Repeated START

After the data transfer, if the master still wants the bus, it can signal another START followed by another slave address without signaling a STOP, as in the following example.

```
RESTART  MOVE.B  I2CR,-(A7)             ;Repeat START (RESTART)
         BSET.B  #2, (A7)
         MOVE.B  (A7)+, I2CR
         MOVE.B  CALLING,-(A7)          ;Transmit the calling address, D0=R/W-
         MOVE.B  CALLING,-(A7)

         MOVE.B  (A7)+, I2DR
```

22.6.6   Slave Mode

In the slave interrupt service routine, software should poll the I2SR[IAAS] bit to determine if the controller has received its slave address. If IAAS is set, software should set the transmit/receive mode select bit (I2CR[MTX]) according to the I2SR[SRW]. Writing to the I2CR clears the IAAS automatically. The only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred; interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer can now be initiated by writing information to I2DR for slave transmits, or read from I2DR in slave-receive mode. A dummy read of I2DR in slave/receive mode releases I2C_SCL, allowing the master to send data.

In the slave transmitter routine, I2SR[RXAK] must be tested before sending the next byte of data. Setting RXAK means an end-of-data signal from the master receiver, after which software must switch it from transmitter to receiver mode. Reading I2DR then releases I2C_SCL so that the master can generate a STOP signal.

22.6.7   Arbitration Lost

If several devices try to engage the bus at the same time, one becomes master. Hardware immediately switches devices that lose arbitration to slave receive mode. Data output to I2C_SDA stops, but I2C_SCL is still generated until the end of the byte during which arbitration is lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with I2SR[IAL] = 1 and I2CR[MSTA] = 0.

If a device that is not a master tries to transmit or execute a START, hardware will inhibit the transmission, clear MSTA without signaling a STOP, generate an interrupt to the CPU, and set IAL to indicate a failed attempt to engage the bus. When considering these cases, the slave service routine should first test IAL and software should clear it if it is set.
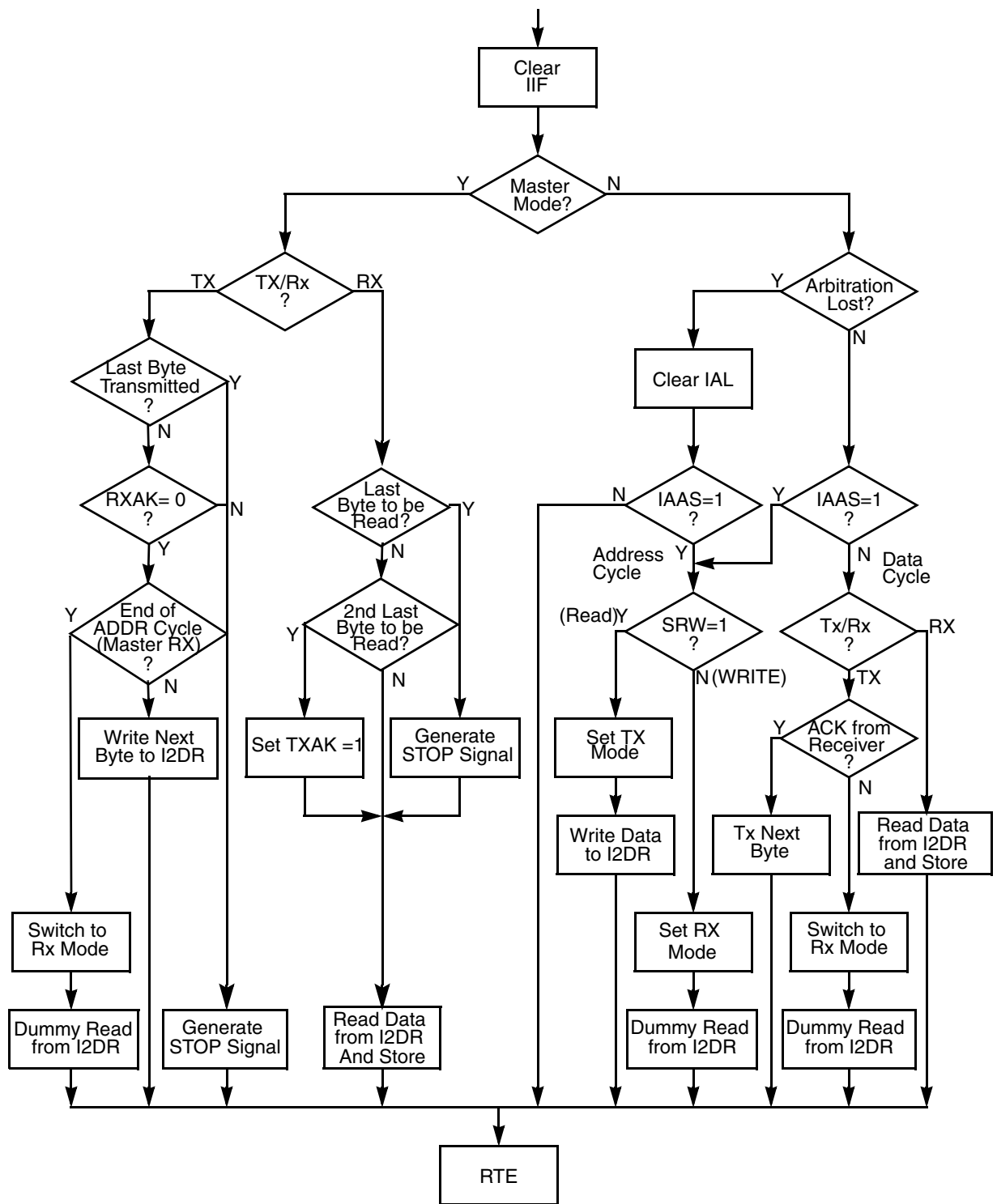
Figure 22-14. Flow-Chart of Typical I²C Interrupt Routine

Chapter 23
Analog-to-Digital Converter (ADC)

23.1   Introduction

The Analog-to-Digital Converter (ADC) consists of two separate and complete ADCs, each with their own sample and hold circuits. The converters share a common voltage reference and common digital control module.

23.2   Features

The ADC's characteristics include:

- 12-bit resolution
- Maximum ADC clock frequency of 5.33MHz, 187.5ns period
- Sampling rate up to 1.78 million samples per second[1]
- Single conversion time of 8.5 ADC clock cycles (8.5×187.5ns = 1.595µs)
- Additional conversion time of 6 ADC clock cycles (6×187.5ns = 1.126µs)
- Eight conversions in 26.5 ADC clocks (26.5×187.5ns = 4.972µs) using Simultaneous mode
- Ability to simultaneously sample and hold two inputs
- Ability to sequentially scan and store up to eight measurements
- Internal multiplex to select two of eight inputs
- Power savings modes allow automatic shutdown/startup of all or part of ADC
- Those inputs not selected tolerate injected/sourced current without affecting ADC performance, supporting operation in noisy industrial environments.
- Optional interrupts at the end of a scan, if an out-of-range limit is exceeded (high or low), or at zero crossing
- Optional sample correction by subtracting a pre-programmed offset value
- Signed or unsigned result
- Single ended or differential inputs for all input pins with support for an arbitrary mix of input types

23.3   Block Diagram

The ADC function, shown in **Figure 23-1,** consists of two four-channel input select functions, interfacing with two independent Sample and Hold (S/H) circuits, which feed two 12-bit ADCs. The two converters store their results in a buffer, awaiting further processing.

---

1. Once in Loop mode, the time between each conversion is six ADC Clock cycles (1.125 µs). Using Simultaneous conversion two samples are captured in 1.126 µs, providing an overall sample rate of 1,776,667 samples per second.

MCF5213 Reference Manual, Rev. 1.1

Preliminary
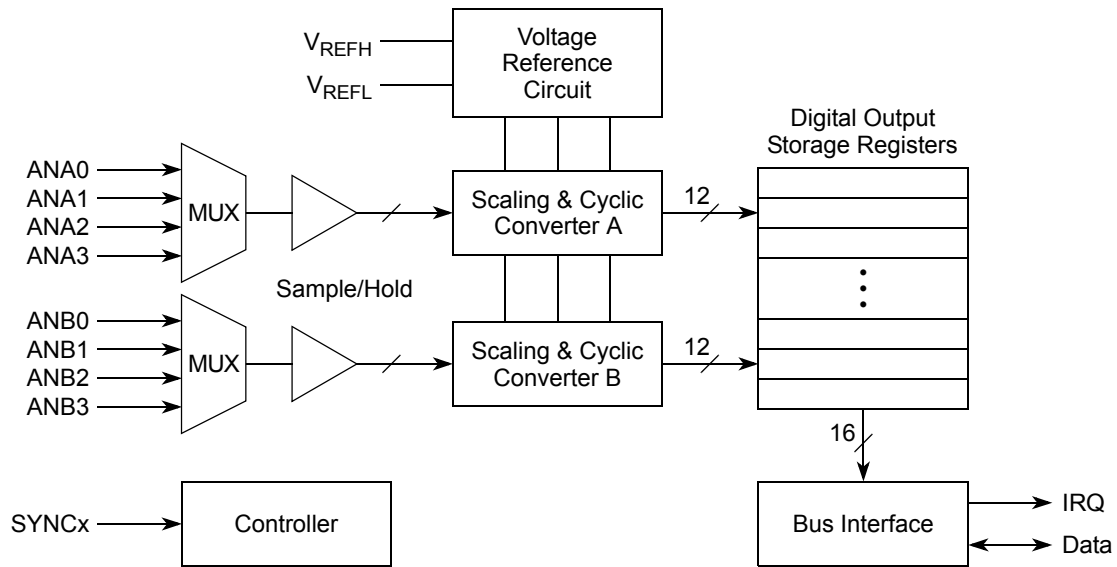
Figure 23-1. Dual ADC Block Diagram

23.4 Functional Description

The ADC's conversion process is either initiated by a sync signal from one of two input pins (SYNCx) or by writing 1 to a START*n* bit.

Starting a single conversion actually begins a sequence of conversions, or a scan of up to eight single ended or differential samples one at a time in sequential scan mode. The operation of the module in sequential scan mode is shown in Figure 23-2.
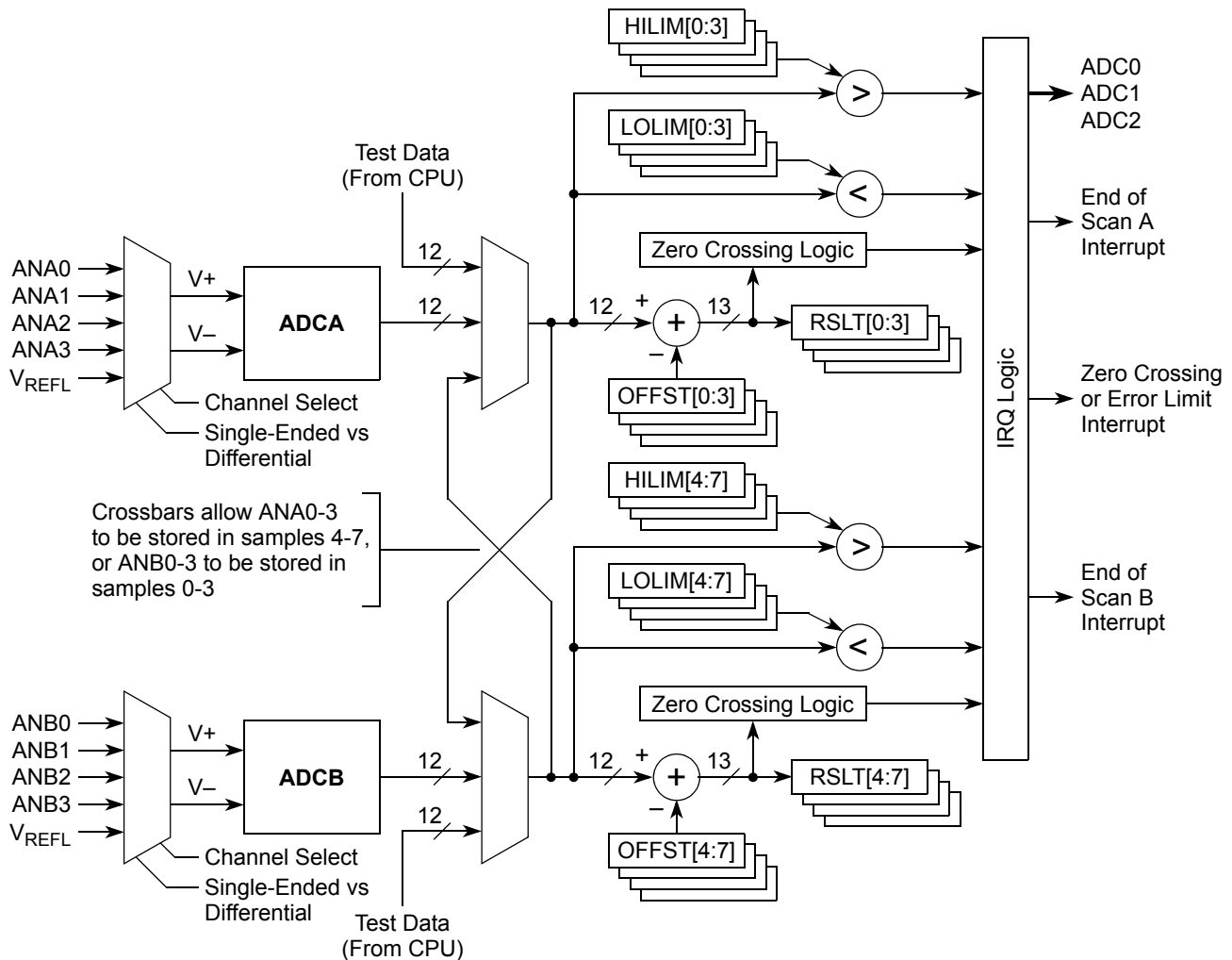
Figure 23-2. Sequential Mode Operation of the ADC

Scan sequence is determined by defining eight sample slots in CLST1/2 registers, processed in order SAMPLE0-7 during sequential scan or in order SAMPLE0-3 by Converter A and in order SAMPLE4-7 by Converter B in parallel Scan. SAMPLE slots may be disabled using the SDIS register.

The following pairs of analog inputs can be configured as a differential pair ANA0-1, ANA2-3, ANB0-1, and ANB2-3. When configured as a differential pair, a reference to either member of the differential pair by a sample slot results in a differential measurement using that differential pair.

Parallel scan can be simultaneous or non-simultaneous. During Simultaneous scan, the scans in the two converters are done simultaneously always resulting in simultaneous pairs of conversions, one by Converter A and one by Converter B. The two converters share the same start, stop, sync, end-of-scan interrupt enable control, and interrupt. Scanning in both converters is terminated when either converter encounters a disabled sample. In non-simultaneous scan, the parallel scans in the two converters are achieved independently. The two converters have their own start, stop, sync, end-of-scan interrupt enable controls, and end-of-scan interrupts. Scanning in either converter terminates only when that converter encounters a disabled sample in its part of SDIS register (DS0-DS3 for A, DS4-DS7 for B).
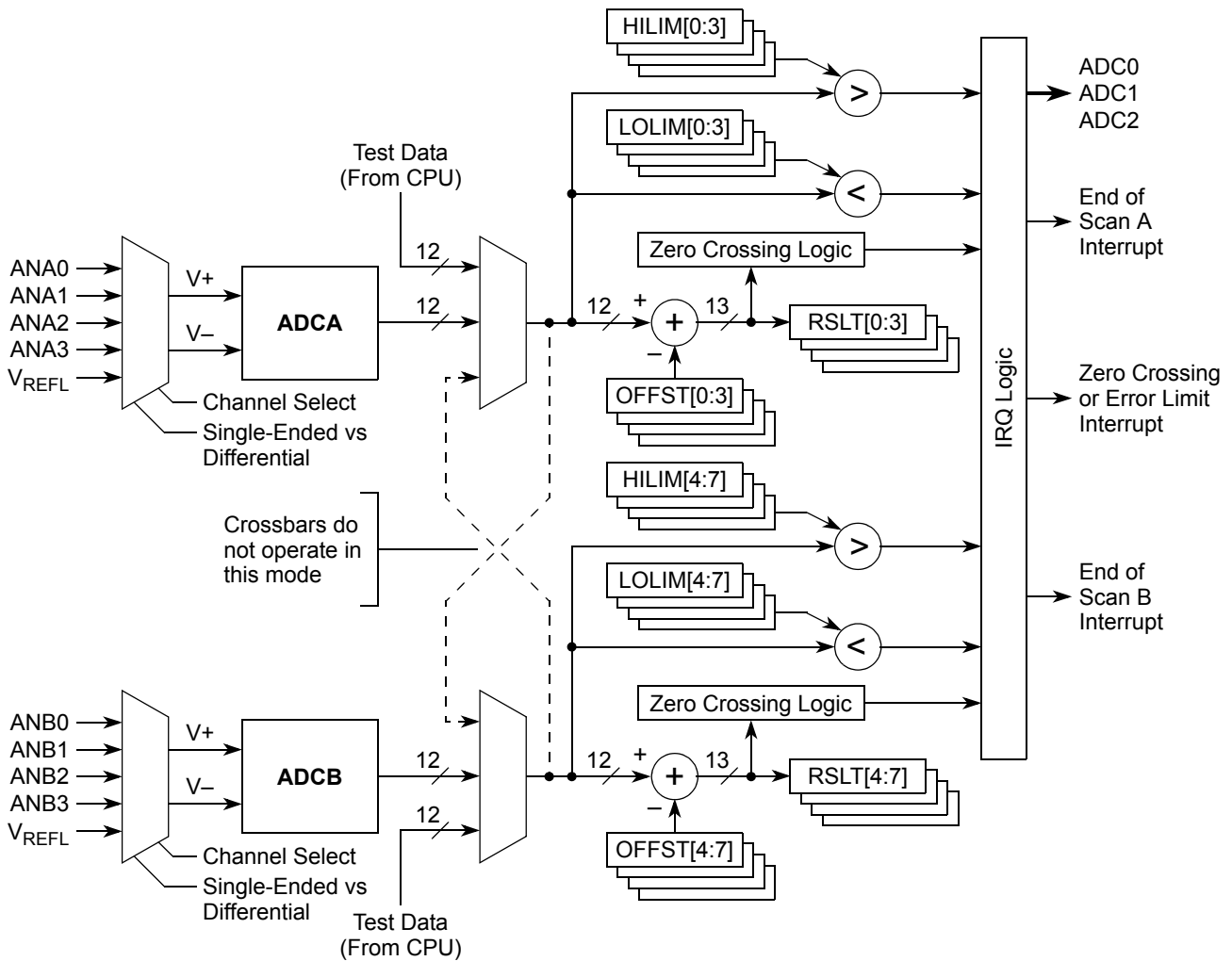
Figure 23-3. Parallel Mode Operation of the ADC

The ADC can be configured to perform a single scan and halt, perform a scan whenever triggered, or perform the scan sequence repeatedly until manually stopped. The single scan (Once mode) differs from the Triggered mode only in that SYNC input signals must be re-armed after each using a Once mode scan and subsequent SYNC inputs are ignored until the SYNC input is re-armed. This arming can occur anytime after the SYNC pulse occurs, even while the scan it initiated is still in process.

Optional interrupts can be generated at the end of a scan sequence. Interrupts are available simply to indicate the scan ended, that a sample was out of range, or at several different zero crossing conditions. *Out-of-Range* is determined by the High and Low Limit registers.

To understand the operation of the ADC it is important to understand the features and limitations of each of the functional parts.

23.4.1   Input MUX Function

The input MUX function is shown in Figure 23-4. The Channel Select and Single Ended vs. Differential switches are indirectly controlled based on settings within the LIST1, LIST2, SDIS registers and the CHNCFG field of the CTRL1 register.

1. MUXing for Sequential Mode, Single-Ended Conversions—During each conversion cycle (sample), any one input of the two muxes can be directed to any RSLT*n* register.

2. MUXing for Sequential Mode, Differential Conversions—During any conversion cycle (sample), either member of a differential pair may be referenced as a SAMPLE, resulting in a differential measurement on that pair being stored in the corresponding RSLT*n* register.

3. MUXing for Parallel Mode, Single-Ended Conversions—During any conversion cycle (sample), any of ANA0-ANA3 can be directed to an RSLT0-3 Result register and any of ANB0-ANB3 can be directed to the RSLT4-7.

4. MUXing for Parallel Mode, Differential Conversions—During any conversion cycle (sample), either member of differential pair ANA0/1 or either member of differential pair ANA2/3 can be referenced as a SAMPLE, resulting in a differential measurement of that pair being stored in one of the RSLT0-3 registers. Likewise either member of differential pair ANB0/1 or either member of differential pair ANB2/3 can be referenced as a SAMPLE, resulting in a differential measurement of that pair being stored in one of the RSLT4-7 registers.

Details of switch operation is shown in Table 23-2. Internally, all measurements are performed differentially. During single ended measurements, V_{REFL} is used as the negative (-) input voltage while the selected analog input is used as the positive (+) input.

Table 23-2. Analog MUX Controls for Each Conversion Mode

Conversion Mode	Channel Select Switches	Single Ended Differential Switches
Sequential, Single Ended	The two 1-of-4 select muxes can be set for the appropriate input line.	The lower switch selects V_{REFL} for the V- input of the A/D. The upper switch is always closed so that any of the four inputs can get to the V+ A/D input.
Sequential, Differential	The channel select switches are turned on in pairs, providing a dual 1-of-2 select function, such that either of the two differential channels can be routed to the A/D input.	The upper switch is open and the bottom switch selects the differential channel for the V- input of the A/D.
Parallel, Single Ended	The two 1-of-4 select muxes can be set for the appropriate input line.	The lower switch is selects V_{REFL} for the V- input of the A/D. The upper switch is always closed so that any of the four inputs can get to the V+ A/D input.
Parallel, Differential	The channel select switches are turned on in pairs, providing a dual 1-of-2 select function, such that either of the two differential channels can be routed to the A/D input.	The upper and lower switches are open and the middle switch is closed, providing the differential channel to the differential input of the A/D.

MUX Configuration for Single-Ended vs **MUX Configuration for Differential**
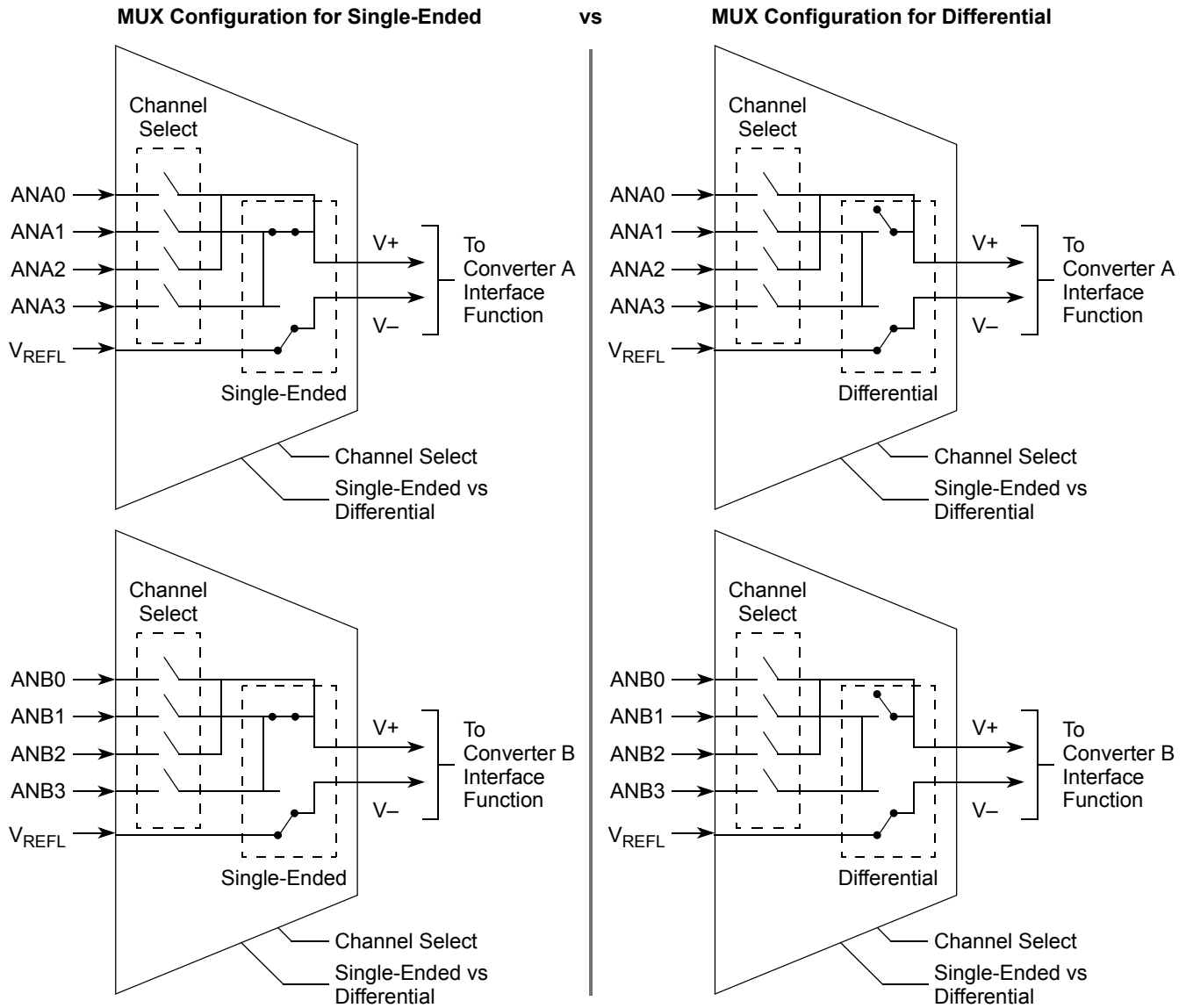
Figure 23-4. Input Select Mux

23.4.2   ADC Sample Conversion

The ADC consists of a cyclic, algorithmic architecture using two recursive sub-ranging sections (RSD#1 and RSD#2), shown in Figure 23-5. Each sub-ranging section resolves a single bit for each conversion clock, resulting in an overall conversion rate of two bits per clock cycle. Each sub-ranging section is designed to run at a maximum clock speed of 5.33MHz. Thus a complete 12-bit conversion takes six ADC clocks (1.125ms), not including sample or post processing time.
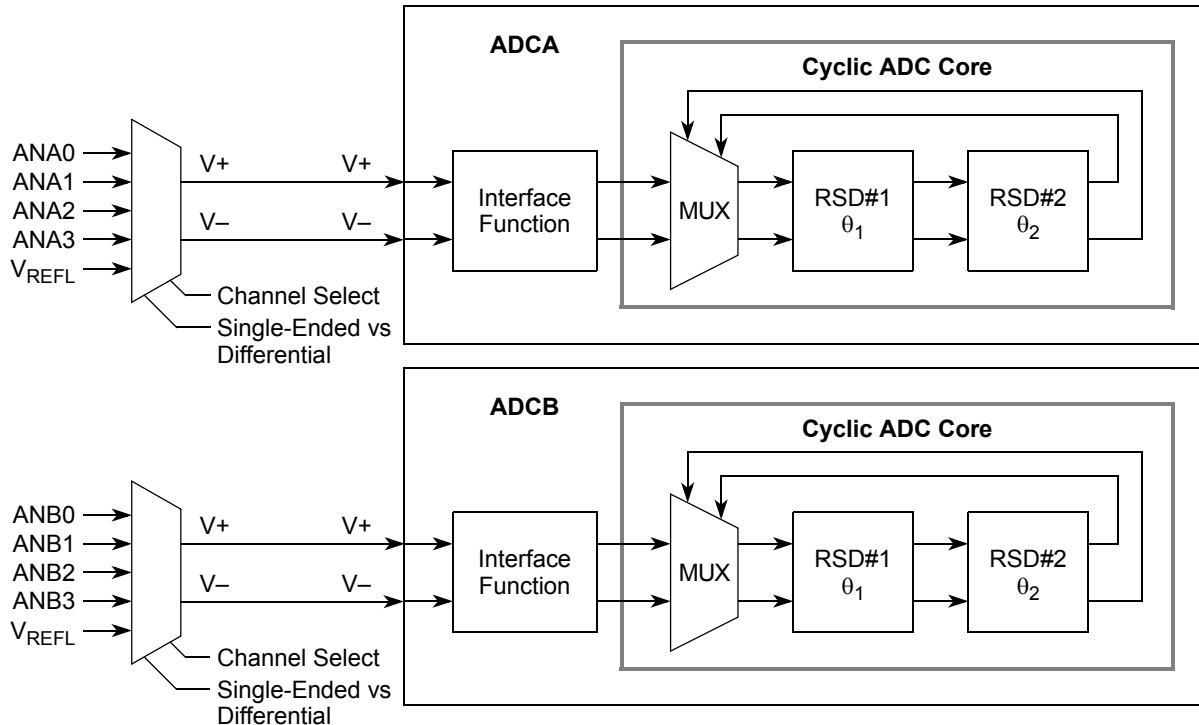
Figure 23-5. Cyclic ADC – Top Level Block Diagram

The ADC has two input modes. The input mode for a given sample is determined by the CHNCFG field of the CTRL1 register. The two input modes are:

1. Single-Ended Mode (CHNCFG bit=0)—In Single-Ended mode, input mux of the ADC selects one of the analog inputs and directs it to the plus terminal of the A/D core. The minus terminal of the A/D core is connected to the V_{REFL} reference during this mode. The ADC measures the voltage of the selected analog input and compares it against the (V_{REFH} - V_{REFL}) reference voltage range.

2. Differential Mode (CHNCFG bit = 1)—In Differential mode, the ADC measures the voltage difference between two analog inputs and compares that against the (V_{REFH} - V_{REFL}) voltage range. The input is selected as an input pair: ANA0/1, ANA2/3, ANB0/1 or ANB2/3. In this mode, the plus terminal of the A/D core is connected to the even analog input while the minus terminal is connected to the odd analog input.

A mix and match combination of single-ended and differential configurations may exist. For example:

- ANA0 and ANA1 differential, ANA2 and ANA3 single-ended
- ANB0 and ANB1 differential, and ANB2 and ANB3 single-ended

23.4.2.1   Single-Ended Samples

The ADC module performs a ratio metric conversion. For single ended measurements, the digital result is proportional to the ratio of the analog input to the reference voltage in the following formula:

$$SingleEndedValue = round(\frac{V_{IN} - V_{REFLO}}{V_{REFH} - V_{REFLO}} \times 4095) \times 8$$

VIN = Applied voltage at the input pin

VREFH and VREFL = Voltage at the external reference pins on the device (typically VREFH = VSSA and VREFL = VDDA)

Note: The 12-bit result is rounded to the nearest LSB.

Note: The ADC is a 12-bit function with 4096 possible states. However, the 12 bits have been left shifted three bits on the 16-bit data bus so its magnitude, as read from the data bus, is now 32760.

23.4.2.2   Differential Samples

For differential measurements, the digital result is proportional to the ratio of the difference in the inputs to the difference in the reference voltages (V_{REFH} and V_{REFL}). Figure 23-6 shows typical configurations for differential inputs.

When converting differential measurements, the following formula is useful:

$$DifferentialValue = round(\frac{V_{IN1} - V_{IN2}}{V_{REFH} - V_{REFLO}} \times 4095) \times 8$$

V_{IN} = Applied voltage at the input pin

V_{REFH} and V_{REFL} = Voltage at the external reference pins on the device (typically $V_{REFH} = V_{SSA}$ and $V_{REFL} = V_{DDA}$)

Note: The 12-bit result is rounded to the nearest LSB.

Note: The ADC is a 12-bit function with 4096 possible states. However, the 12 bits have been left shifted three bits on the 16-bit data bus so its magnitude, as read from the data bus, is now 32760.

Differential buffer will center about mid-point

NOTE: Normally, V_{REFL} is set to $V_{SSA} = 0V$

Center tap held at $(V_{REFH} + V_{REFL})\ /2$
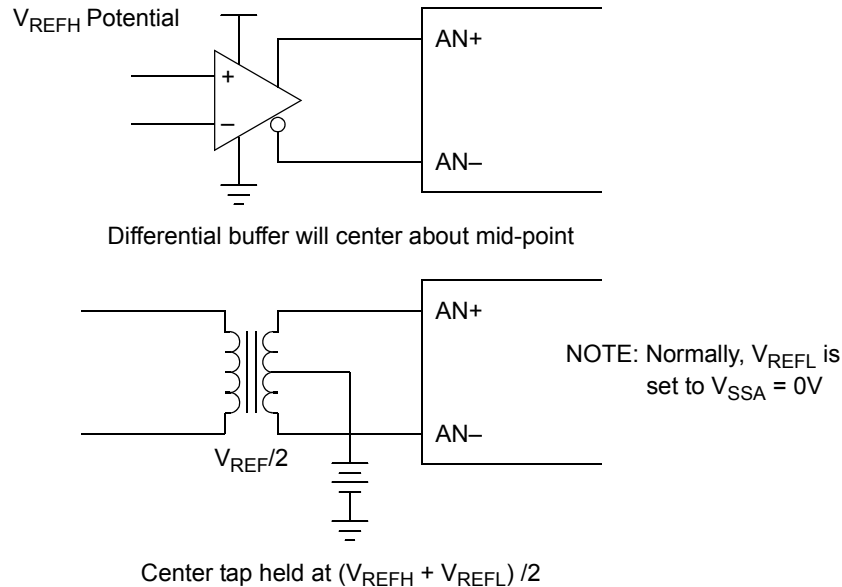
Figure 23-6. Typical Connections for Differential Measurements

23.4.3 ADC Data Processing

As shown in **Figure 23-7**, the raw result of the ADC conversion process is sent to an adder for offset correction. The adder subtracts the OFFST*n* register value from each sample and the result is then stored in the corresponding Result Register (RSLT*n*). Concurrent to this the raw ADC value is checked for limit violations and the RSLT*n* values are checked for zero-crossing. Appropriate interrupts are asserted, if enabled.

The sign of the result is calculated from the ADC unsigned result minus the respective offset register. If the offset register is programmed with a value of zero, the result register value is unsigned and equals the cyclic converter unsigned result. The range of the Result (RSLT) register is $0000–$7FF8, assuming the Offset (OFFST) register is set to zero.

The processor can write to the result registers whenever the ADC is in Stop mode or powered down. The data from this write operation is treated as if it came from the ADC analog core; so the limit checking, zero crossing, and the Offset registers' function as if in Normal mode. For example, if the ADC is stopped and the processor writes to RSLT5 register, the data written to the RSLT5 register is muxed to the ADC digital logic inputs, processed, and stored into RSLT5 as if the analog core had provided the data. This test data must be left justified by three bits (as shown in the RSLT register definition) and does not include the sign bit. The sign bit (SEXT) is calculated during subtraction of the corresponding OFFST*n* offset value.
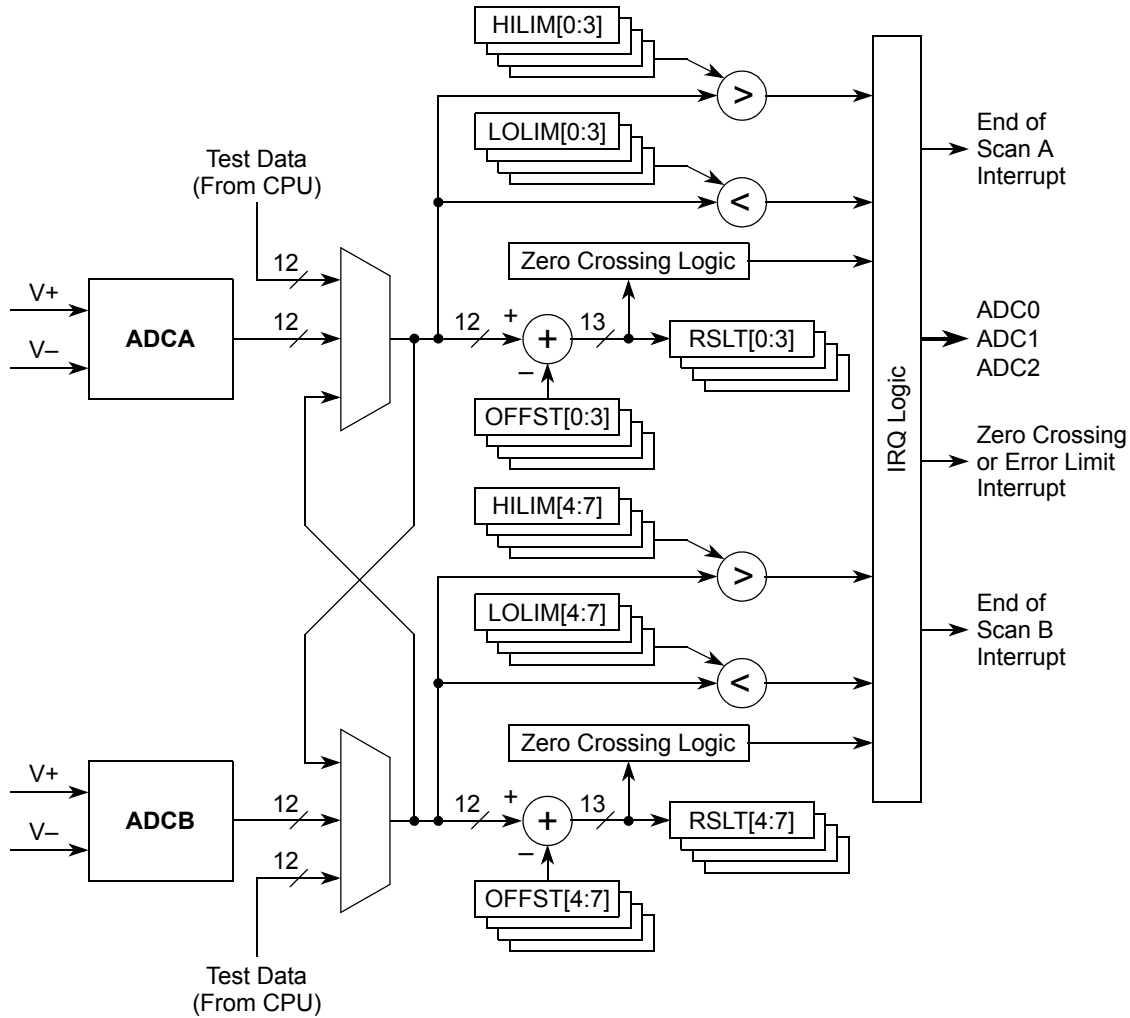
Figure 23-7. Result Register Data Manipulation

23.4.4   Sequential vs. Parallel Sampling

All scan modes make use of the eight SAMPLE slots in the CLST1 and CLST2 registers. These slots are used to define which single-ended input or differential input pair is measured at each step in a scan sequence. The SDIS register is used to disable unneeded slots.

Differential measurements are made on input pairs ANA0/1, ANA2/3, ANB0/1, and ANB2/3 using the CHNCFG field of the CTRL1 register. A single ended measurement will be made if a SAMPLE slot refers to an input not configured as a member of a differential pair by CHNCFG. A differential measurement will be made if a SAMPLE slot refers to either member of a differential pair. Refer to the CHNCFG field description in the CTRL1 register for details of differential and single ended measurement.

Scan modes are either Sequential or Parallel, as defined by the SMODE field of the CTRL1 register.   In sequential scans, up to eight SAMPLE slots are sampled one at a time in the order SAMPLE 0-7. Each SAMPLE slot may refer to any of the eight analog inputs (ANA0-3 and ANB0-3), thus the same input may

be referenced by more than one SAMPLE slot. Scanning is initiated when the START0 bit is written as 1 or, if the SYNC0 bit is 1, when the SYNC0 input goes high. A scan ends when the first disabled sample slot is encountered in the SDIS register. Completion of the scan triggers the EOSI0 interrupt if the interrupt is enabled by the EOSIE0 bit. The START0 bit and SYNC0 input are ignored while a scan is in process. Scanning stops and cannot be initiated when the STOP0 bit is set.

Parallel scans differ in that Converter A collects up to four samples (SAMPLE 0-3) in parallel to Converter B collecting up to four samples (SAMPLE 4-7). SAMPLEs 0-3 may only reference inputs ANA0-3 and SAMPLEs 4-7 may only reference inputs ANB0-3. Within these constraints, any sample may reference any pin and the same input may be referenced by more than one sample slot.

By default (when SIMULT=1), Parallel scans of the converters are initiated together when the START0 bit is written as 1 or, if the SYNC0 bit is 1, when the SYNC0 input goes high. The scan in both converters terminates when either converter encounters a disabled sample slot in SDIS. Completion of a scan triggers the EOSI0 interrupt provided the EOSIE0 interrupt enable is set. Samples are always taken simultaneously in both the A and B converters. Setting the STOP0 bit stops and prevents the initiation of scanning in both converters.

Setting SIMULT=0 (non-simultaneous mode) causes parallel scanning to operate independently in the A and B converter. Each converter has its own set of STARTn, STOPn, SYNCn, and EOSIEn control bits, SYNCn input, EOSIn interrupt, and CIPn status indicators ($n = 0$ for converter A, $n = 1$ for converter B). Though still operating in parallel, the scans in the A and B converter start and stop independently according to their own controls and may be simultaneous, phase shifted, or asynchronous depending on when scans are initiated on the respective converters. The A and B converter may be of different length (still up to a maximum of four) and each converter's scan completes when a disabled sample is encountered in that converters sample list only. STOP0 only stops the A converter and STOP1 only stops the B converter. Looping scan modes repeat independently, with the A converter capturing SAMPLE 0-3 and B converter capturing SAMPLE 4-7. In Loop modes, each converter independently restarts its scan after capturing its samples.

23.4.5   Scan Sequencing

Scan modes break down into three types based on how they repeat. These types are Once, Triggered, or Loop. Be certain to read Section 23.4.4, "Sequential vs. Parallel Sampling" to understand the operation of sequential and parallel scan modes before proceeding further.

During a Once mode scan a single sequential or parallel scan is executed. Once scan modes differ from triggered scan modes in that they must be re-armed after each use. While all scan modes ignore sync pulses occurring while a scan is in process, Once scan modes will continue to ignore sync pulses even after the scan completes until re-armed. Re-arming, however, can occur any time including during the scan by writing to a CTRLn register. If operating in a Sequential mode or simultaneous parallel write to the CTRL1 register. If operating in a non-simultaneous Parallel mode, re-arm Converter A by writing to the CTRL1 register and Converter B by writing to the CTRL2 register.

Triggered scan modes are identical to the corresponding Once scan modes except that re-arming of sync inputs is not necessary.

Loop scan modes automatically restart a scan as soon as the previous scan completes. In the Loop Sequential mode up to eight samples are captured in each loop and the next scan starts immediately after the completion of the previous scan. In Loop Parallel scan modes, both Converters restart together if SIMULT=1 and restart independently if SIMULT=0. All subsequent start and sync pulses will be ignored after the scan begins. Scanning can only be terminated by setting a STOP*n* bit. Use STOP0 in the CTRL1 register if operating in a Sequential or simultaneous Parallel mode. If operating in a non-simultaneous Parallel mode use STOP0 to stop Converter A and STOP1 in the CTRL2 register to stop Converter B.

23.4.6 Power Management

The five supported power modes are described below. They are presented in order from highest to lowest power utilization at the expense of increased conversion latency and/or startup delay. Please see the Clocks section (Figure 23.4.7) for details of the various clocks referenced below.

23.4.6.1 Power Management Modes

1. NORMAL POWER MODE
   This mode operates when:
   — At least one ADC converter is powered up (PD0 or PD1=0 in the PWR register);
   — Both Auto Power-Down and Auto Standby modes are disabled (APD=0, ASB=0 in ADCPOWER);
   — The ADC's clock is enabled (ADC=1 in the SIM module's SIM_PCE register).

In this mode the ADC uses the Conversion Clock as the ADC clock source both when active or idle. To minimize conversion latency it is recommended the conversion clock be configured to 5.33MHz. No startup delay (defined by PUDELAY in the PWR register) is imposed.

2. AUTO POWER-DOWN MODE
   This mode operates when:
   — At least one ADC converter is powered up (PD0 or PD1=0 in the PWR register);
   — Auto Power-Down mode enabled (APD=1 in the PWR register);
   — The ADC's clock is enabled (ADC=1 in the SIM module's SIM_PCE register).

Auto Power-Down and Standby modes can be used together by setting APD=1 in the above configuration. This hybrid mode converts at an ADC clock rate of 100kHz using Standby Current mode when active and gates off the ADC clock and powers down the converters when idle. A startup delay of PUDELAY ADC clock cycles execute at the start of all scans while the ADC engages the conversion clock and the ADC powers up, stabilizing in the Standby Current mode. This provides the lowest possible power configuration for ADC operation.

3. AUTO STANDBY MODE
   This mode operates when:
   — At least one ADC converter is powered up (PD0 or PD1=0 in the PWR register);
   — Auto Power-Down is disabled (APD=0 in the PWR register);
   — Auto Standby is enabled (ASB=1 in the PWR register);
   — The ADC's clock is enabled (ADC=1 in the SIM module's SIM_PCE register);

— Either the relaxation oscillator must be enabled for 8MHz operation or the external oscillator clock must be running at 8MHz in this mode.

In Auto Standby Mode, the ADC uses the Conversion Clock when active and the100kHz Standby Clock when idle. The standby (low current) state automatically engages when the ADC is idle. It is recommended that the conversion clock be configured at or near 5.33MHz to minimize conversion latency. The ADC will execute a startup delay of PUDELAY ADC clocks at the start of all scans, allowing the ADC to switch to the Conversion Clock and to revert from Standby to Normal Current mode.

It is recommended the conversion clock be configured at or near 5.33MHz to minimize conversion latency when active. In this mode, the ADC uses the conversion clock when active and gates off the Conversion Clock and powers down the converters when idle. A startup delay of PUDELAY ADC clocks is executed at the start of all scans, allowing the ADC to stabilize when switching to normal current mode from a completely powered off condition. This mode uses less power than Normal and more power than Auto Standby. It requires more startup latency (than Auto Standby) when leaving the idle state to start a scan (higher PUDELAY value).

4. POWER-DOWN MODE
   This mode operates when:

   — Both ADC converters are powered down (PD0=PD1=1 in the PWR register);
   — The ADC's clock is disabled (ADC=0 in the SIM module's SIM_PCE register).

   In this configuration, the clock trees to the ADC and all of its analog components are shut down and the ADC uses no power.

23.4.6.2 Power Management Details

The ADC voltage reference and converters are powered down (PDn=1 in the PWR register) on reset. Individual converters can be manually powered down when not in use (PD0=1 or PD1=1) and the voltage reference can be automatically powered down when no converter is in use (PD2=1), or manually powered up when no converters are powered (PD2=0). When the ADC voltage reference is powered down, output reference voltages are set to Low (V_{SSA}).

A delay of PUDELAY ADC clock cycles is imposed whenever PD0 or PD1 are cleared to power-up a converter and whenever the ADC goes from an idle (neither converter has a scan in process) to an active state when not operating in Normal Power mode. The ADC is active when at least one converter has a scan in process. A device recommends the use of two PUDELAY values, a large value for full power-up and a smaller value for going from standby current levels to full power-up. The following paragraphs provide an explanation of how to use PUDELAY when starting the ADC up or changing modes.

When starting up in Normal mode, first set PUDELAY to the large power-up value. Next, clear the PD0 and or PD1 bits to power-up the required converters. Poll the status bits (PSTS*n* in the PWR register) until all required converters are powered up. Following polling, start scan operations. The value in PUDELAY will provide a power-up delay before scans begin. Since Normal mode does not use PUDELAY at start of scans, no further delays will be imposed.

When starting up using Auto Standby mode, first use the Normal mode startup procedure. Before starting scan operations, set PUDELAY to the smaller value, and then set ASB in the PWR register. Auto Standby

mode will automatically reduce current levels until active and then impose a PUDELAY wait to allow current levels to rise from standby to normal levels.

When starting up using Auto Power-Down mode, first use the Normal mode startup procedure. Before starting scan operations, set PUDELAY to the large power-up value. Next, set APD in the PWR register. Finally, clear the PD0 and or PD1 bits for the required converters. Converters remain powered off until scanning goes active at which time the large PUDELAY executes as the ADC goes from powered down to fully powered at the start of the scan.

In Auto Power-Down mode, when the ADC goes from idle to active, a converter is only powered up if it is required for the scan, as determined by the CLST1, CLST2, and SDIS registers.

It is recommended to power-off both converters (PD0=PD1=1 in the PWR register) when re-configuring clocking or power controls to avoid generating bad samples, ensuring proper delays are applied when powering up or starting scans.

Attempts to start a scan during the PUDELAY time-out will be ignored until the appropriate PSTSn bits are cleared in the PWR register.

Any attempt to use a converter when powered down, or with the voltage reference disabled, results in invalid results. It is possible to read ADC result registers after converter power down to see results calculated before power-down. However, a new scan sequence must be started with a SYNCn pulse or a write to the STARTn bit before new results will be available.

23.4.6.3   ADC STOP Mode of Operation

Any conversion sequence in progress can be stopped by setting the relevant STOPn bit. Any further sync pulses, or writes to the STARTn bit, are ignored until the STOPn bit is cleared. Once in this stop mode, the results registers can be modified by writes from the processor. Any write to RSLTn registers in the ADC Stop mode is treated as if the analog core supplied the data, so limit checking, zero crossing, and associated interrupts can occur if enabled.

23.4.7   ADC Clock

23.4.7.1   General

The ADC has two external clock inputs used to drive two clock domains within the ADC module.

Table 23-3. ADC Clock Summary

Clock input	Source	Characteristics
Peripheral Clock (=System Clock)	1/2 Core clock	Maximum rate is PLL output divided by 2 if PLL enabled. When PLL disabled, max rate is oscillator clock divided by 2.
ADC 8MHz Clock	Relaxation Oscillator (8Mhz), Crystal Oscillator (1-16MHz), or external Oscillator	Provides 8Mhz for auto standby power saving mode.

23.4.7.2 Description of Clock Operation

As shown in Figure 23-8, the Conversion Clock is the primary source for the ADC clock and is always selected as the ADC clock when conversions are in process. The DIV value in the CTRL2 register should be configured so the Conversion Clock frequency falls between 100kHz and 5.33MHz. Operating the ADC at out-of-spec clock frequencies will degrade conversion accuracy. Similarly, modifying the parameters affect clock rates or power modes while the regulators are powered up (PD0=0 or PD1=0) will also degrade conversion accuracy.

The Conversion Clock ADC uses for sampling is calculated using the IPBus Clock and the clock divisor bits within the CTRL2 register. Please see Section 23.5.1, "Control 1 Register (CTRL1)" or Section 23.5.2, "Control 2 Register (CTRL2) Under Sequential Scan Modes". The ADC clock is active 100 percent of the time while in Loop modes, or if power management is set to Normal. It is also active during all ADC power-up for a period of time determined by the PUDELAY field in the Power (PWR) Register. After the power-up delay times out, the ADC clock continues until the completion of the ADC*n* scan when operating in Auto Standby or Auto Power-Down modes.
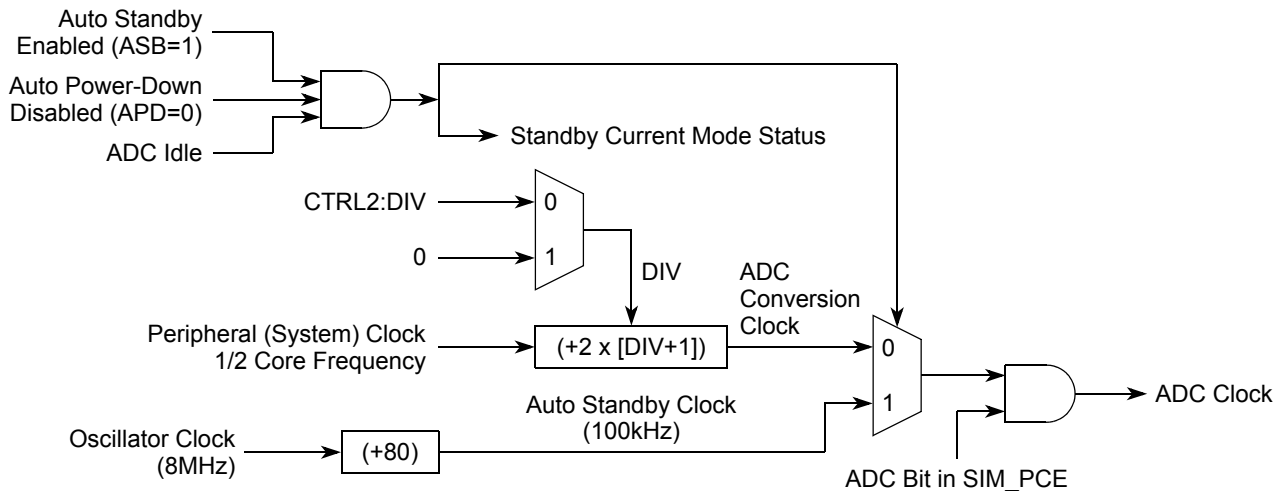
Figure 23-8. ADC Clock Generation

The oscillator clock feeds a 80:1 divider, generating the Auto Standby clock. The Auto Standby clock is selected as the ADC clock during the Auto Standby Power mode when both converters are idle. The Auto Standby Power mode requires an 8MHz oscillator clock from the relaxation oscillator, crystal oscillator or external oscillator.

23.4.7.3 ADC Clock Resynchronization at Start of Scan

At the fastest ADC speed, each ADC clock period is 6 System Clock periods long. When asserting the start of a scan, either by writing to a START*n* bit or by a SYNC*n* signal, the ADC clock is re-synchronized to align it to the system clock. This allows the commanded scan to begin as soon as possible rather than wait up to five additional system clocks for the start of the next ADC clock period. This is shown in Figure 23-9 for both Sequential and Simultaneous Parallel modes of operation. In these modes both ADC operate off of the same start signal.

In a Parallel scan mode when SIMULT=0 both ADCs operate using independent START*n* bits and SYNC*n* signals. As shown in Figure 23-10, the first scan started will be re-synchronized to the system clock but the second scan may wait up to five additional system clocks before starting. Also, please note that which converter is synchronized to the system clock depends on which convert first starts to use the ADC. The case shown has ADCA synchronized, but one could easily imagine the case where the ADCA start comes after instead of before the ADCB start. In this case ADCAs start would be delayed up to five additional system clock periods instead of ADCBs.

If there is a known timing relationship between ADCA and ADCB when operating in a non-simultaneous Parallel mode then the application can control which ADC starts first and gets the re-synchronized clock. The application can also control the delay to starting the second ADC scan so that its start signal aligns with the ADC clock and the start of the second ADC is not delayed.
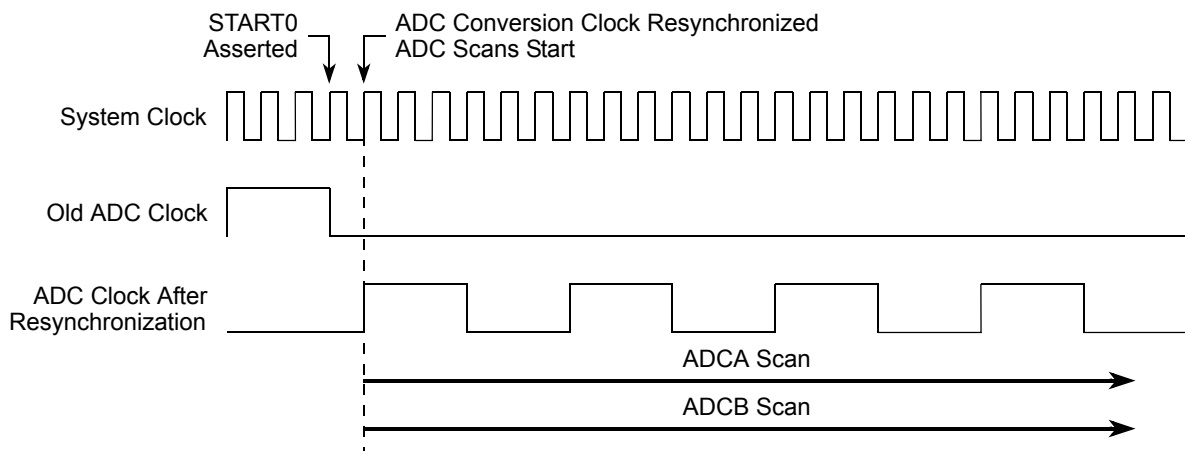
**Figure 23-9. ADC Clock Resynchronization for
Sequential and Simultaneous Parallel Modes**
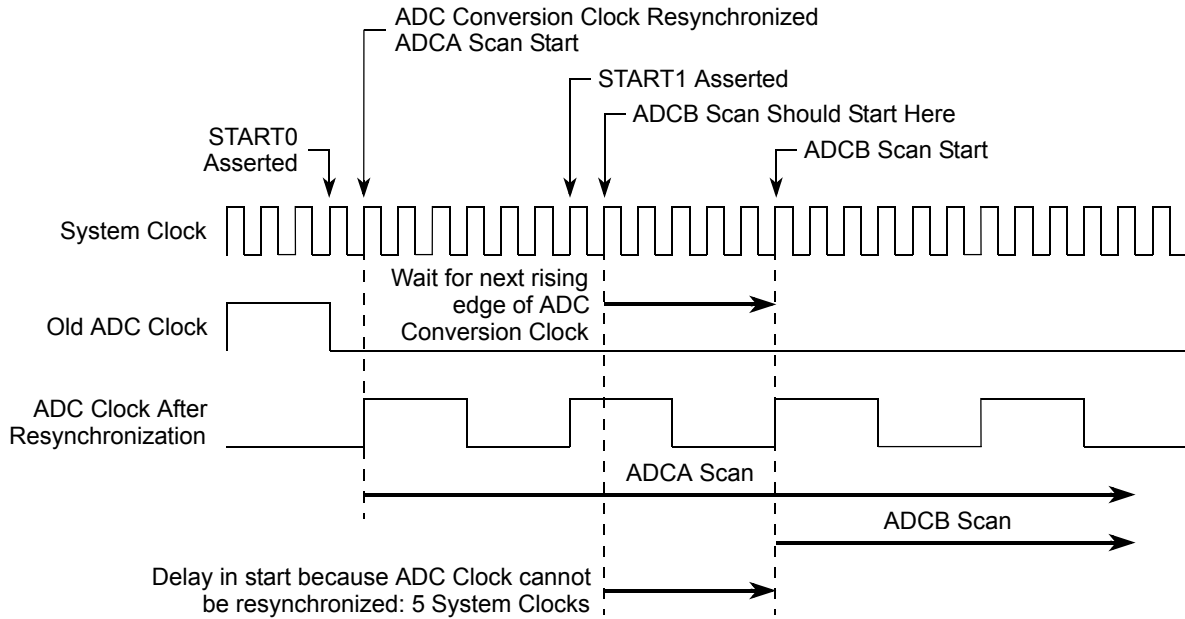
**Figure 23-10. ADC Clock Resynchronization for
Non-Simultaneous Parallel Modes**

23.4.8   Voltage Reference Pins V_{REFH} & V_{REFL}

The voltage difference between V_{REFH} and V_{REFL} provides the reference voltage that all analog inputs are measured against. The reference voltage should be provided from a low noise filtered source capable of providing up to 1mA of reference current.
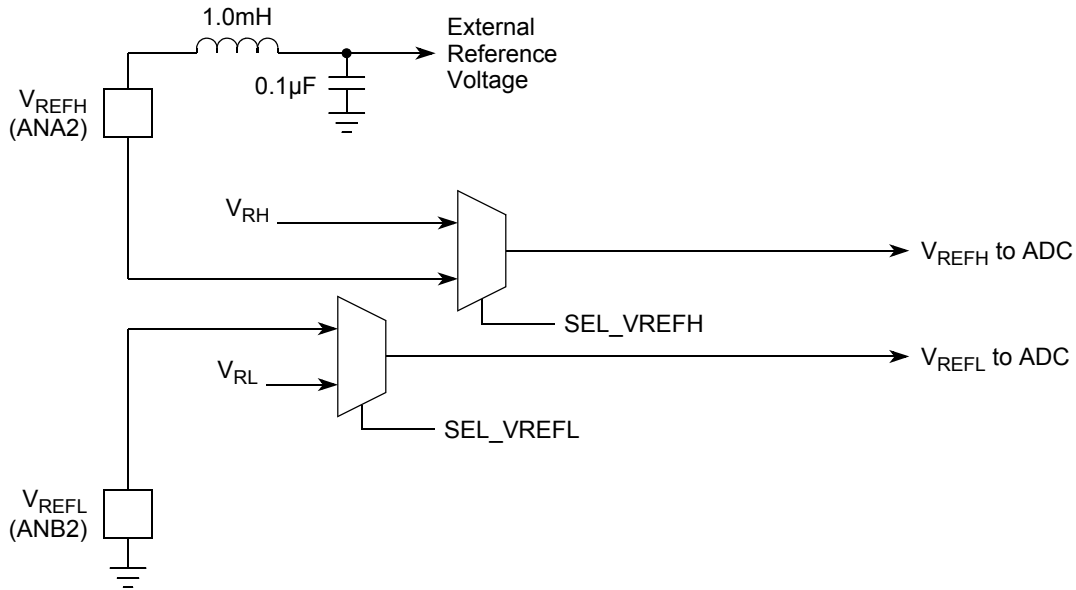
Figure 23-11. ADC Voltage Reference Circuit

When tying V_{REFH} to the same potential as V_{DDA} relative measurements are being made with respect to the amplitude of V_{DDA}. It is imperative special precautions be taken assuring the voltage applied to V_{REFH} be as noise free as possible. Any noise residing on the V_{REFH} voltage is directly transferred to the digital result.

Figure 23-11 illustrates the internal workings of the ADC voltage reference circuit. V_{REFH} must be noise filtered; a minimum configuration is shown in the figure.

23.4.9 Supply Pins V_{DDA} and V_{SSA}

Dedicated power supply pins are provided for the purposes of reducing noise coupling and to improve accuracy. The power provided to these pins is suggested to come from a low noise filtered source. Uncoupling capacitors ought to be connected between V_{DDA} and V_{SSA}.

23.5 Register Definitions

Table 23-4. ADC Memory Map

Device	Peripheral	Base Address
MCF5213	ADC	$00F080

A register address is the sum of a base address and an address offset. The base address is defined at the device level and the address offset is defined at the module level.

Table 23-5 lists the ADC registers in ascending address order, including their acronyms and address offset of each register. ADC uses ADC*n*_BASE plus the given offset depending on the ADC being used. The ADC peripheral has 43 registers.

Table 23-5. ADC Register Summary

IPSBAR Offset	Acronym	Register Name	Access Type	Location
0x0019_0000	CTRL1	Control Register 1	Read/Write	**Section 23.5.1**
0x0019_0001	CTRL2	Control Register 2	Read/Write	**Section 23.5.2** **Section 23.5.3**
0x0019_0002	ZXCTRL	Zero Crossing Control Register	Read/Write	**Section 23.5.4**
0x0019_0003	CLST1	Channel List Register 1	Read/Write	**Section 23.5.5**
0x0019_0004	CLST2	Channel List Register 2	Read/Write	
0x0019_0005	CSDIS	Sample Disable Register	Read/Write	**Section 23.5.6**
0x0019_0006	CSTAT	Status Register	Read/Write	**Section 23.5.7**
0x0019_0007	LIMSTAT	Limit Status Register	Read/Write	**Section 23.5.8**
0x0019_0008	ZXSTAT	Zero Crossing Status Register	Read/Write	**Section 23.5.9**
0x0019_0009–10	RSLT0-7	Result Registers 0-7	Read/Write	**Section 23.5.10**
0x0019_00011–18	LOLIMT0-7	Low Limit Registers 0-7	Read/Write	**Section 23.5.11**
0x0019_00019–20	HILIMT0-7	High Limit Registers 0-7	Read/Write	
0x0019_00021–28	OFFST0-7	Offset Registers 0-7	Read/Write	**Section 23.5.12**
0x0019_00029	PWR	Power Control Register	Read/Write	**Section 23.5.13**
0x0019_0002A	VREF	Voltage Reference Register	Read/Write	**Section 23.5.14**

Bits of each of the 43 registers are summarized in **Figure 23-12.** Details of each follow.

Add. Offset	Register Acronym		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$0	CTRL1	R	0		0	SYNC 0	EOSIE 0	ZCIE	LLMTI E	HLMTI E	CHNCFG				0	SMODE		
		W		STOP0	START 0													
$1	CTRL2 Simultaneous Mode	R	0	0	0	0	0	0	0	0	0	0	0	DIV				
		W																
	CTRL2 Parallel Mode	R	0		0	SYNC 1	EOSIE 1	0	0	0	0	0	SIMULT	DIV				
		W		STOP1	START 1													
$2	ZXCTRL	R	ZCE7		ZCE6		ZCE5		ZCE4		ZCE3		ZCE2		ZCE1		ZCE0	
		W																
$3	CLST1	R	0	SAMPLE3			0	SAMPLE2			0	SAMPLE1			0	SAMPLE0		
		W																
$4	CLST2	R	0	SAMPLE7			0	SAMPLE6			0	SAMPLE5			0	SAMPLE4		
		W																
$5	SDIS	R	0	0	0	0	0	0	0	0	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
		W																

Figure 23-12. ADC Register Map Summary

$6	STAT	R	CIP0	CIP1	0	EOSI1	EOSI0	ZCI	LLMTI	HLMT	RDY7	RDY6	RDY5	RDY4	RDY3	RDY2	RDY1	RDY0
		W																
$7	LIMSTAT	R	HLS7	HLS6	HLS5	HLS4	HLS3	HLS2	HLS1	HLS0	LLS7	LLS6	LLS5	LLS4	LLS3	LLS2	LLS1	LLS0
		W																
$8	ZXSTAT	R	0	0	0	0	0	0	0	0	ZCS7	ZCS6	ZCS5	ZCS4	ZCS3	ZCS2	ZCS1	ZCS0
		W																
$9-$10	RSLTn	R	SEXT	RSLT												0	0	0
		W		TEST_DATA														
$11-$18	LOLIMn	R	0	LLMT												0	0	0
		W																
$19-$20	HILIMn	R	0	HLMT												0	0	0
		W																
$21-$28	OFFSTn	R	0	OFFSET												0	0	0
		W																
$29	PWR	R	ASB	0	0	PSTS2	PSTS1	PSTS0	PUDELAY						APD	PD2	PD1	PD0
		W																
$2A	VREF	R	SEL_VREFH	SEL_VREFL	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																

R	0	Read as 0
W		Reserved

Figure 23-12. ADC Register Map Summary (continued)

23.5.1   Control 1 Register (CTRL1)

Bits 14, 13, 12, and 11 in CTRL1 control all types of scans except parallel scans in the B converter when SIMULT=0 in the CTRL2 register. SIMULT=0 bits 14, 13, 12, and 11 in CTRL are used to control Converter B scans in parallel scan modes while the equivalent bits in CR1 are used for Converter A.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—	STOP0	START0	SYNC0	EOSIE0	ZCIE	LLMTIE	HLMTIE	CHNCFG				—	SMODE		
Reset	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1
R/W	—	R/W	W	R/W									—	R/W		
Address	IPSBAR + 0x19_0000															

Figure 23-13. Control 1 (CTRL1) Register

MCF5213 Reference Manual, Rev. 1.1

23.5.1.1   Reserved—Bit 15

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

23.5.1.2   STOP 0 (STOP0)—Bit 14

When STOP0 is asserted, the current scan is stopped and no further scans can start. Any further SYNC0 input pulses (see SYNC0 bit 12) or writes to the START0 bit are ignored until the STOP0 bit is cleared. After the ADC is in Stop mode, the result registers can be modified by the processor. Any changes to the result registers in Stop mode are treated as if the analog core supplied the data. Therefore, limit checking, zero crossing, and associated interrupts can occur if enabled. *This is not the same as the device's STOP mode.*

- 0 = Normal operation
- 1 = Stop mode

23.5.1.3   Start Conversion (START0)—Bit 13

A scan is started by writing 1 to the START0 bit. This is a *write-only* bit. Writing 1 to the START0 bit again will be ignored until the end of the current scan.

- 0 = No action
- 1 = Start command is issued

The ADC must be in a stable power configuration prior to writing the START bit.   Refer to the functional description of power modes for further details.

23.5.1.4   Synchronization 0 Enable (SYNC0)—Bit 12

A conversion may be initiated by asserting a positive edge on the SYNC0 input. Any subsequent SYNC0 input pulses while the scan remains in process are ignored.

- 0 = Scan is initiated by a write to START0 bit only
- 1 = Use a SYNC0 input pulse or START0 bit to initiate a scan

The ADC must be in a stable power mode prior to SYNC0 input assertion. Refer to the functional description of power modes for further details.

In *OnCE* scan modes, only the first SYNC0 input pulse is honored. Subsequent SYNC0 input pulses are ignored until SYNC0 input is re-armed by writing to the CTRL1 register, usually by simply rewriting 1 to SYNC0. This is achieved at any time, even during the execution of the scan.

23.5.1.5   End Of Scan Interrupt Enable 0 (EOSIE0)—Bit 11

This bit enables an EOSI0 interrupt to be generated upon completion of the scan. For looping scan modes, the interrupt will trigger after the completion of each iteration of the loop.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

Preliminary

23.5.1.6   Zero Crossing Interrupt Enable (ZCIE)—Bit 10

This bit enables the zero crossing interrupt if the current result value has a sign change from the previous result as configured by the ZXCTRL register.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

23.5.1.7   Low Limit Interrupt Enable (LLMTIE)—Bit 9

This bit enables the Low Limit exceeded interrupt when the current result value is less than the Low Limit register value. The raw result value is compared to the LOLIM register, bits LLMT[11:0], before the Offset register value is subtracted.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

23.5.1.8   High Limit Interrupt Enable (HLMTIE)—Bit 8

This bit enables the High Limit exceeded interrupt if the current result value is greater than the High Limit register value. The raw result value is compared to the High Limit (HILIM) register, bits HLMT[11:0], before the Offset register value is subtracted.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

23.5.1.9   Channel Configure (CHNCFG)—Bits 7–4

The inputs can be configured for either single-ended or differential conversions.

Table 23-6. CHNCFG Bit Settings

Bit Settings	Inputs	Description
xxx1	ANA0–ANA1	Configured as differential pair (ANA0 is + and ANA1 is −)
xxx0		Both configured as single ended inputs
xx1x	ANA2–ANA3	Configured as differential pair (ANA2 is + and ANA3 is −)
xx0x		Both configured as single ended inputs
x1xx	ANB0–ANB1	Configured as differential pair (ANB0 is + and ANB1 is −)
x0xx		Both configured as single ended inputs
1xxx	ANB2–ANB3	Configured as differential pair (ANB2 is + and ANB3 is −)
0xxx		Both configured as single ended inputs

Differential measurements return the max value $32760 (= 4095 \times 8)$ when the plus (+) input is V_{REFH} and the minus (−) input is V_{REFL}, return 0 when the plus (+) input is at V_{REFL} and the minus (−) input is at V_{REFL}, and scale linearly between based on the voltage difference between the two signals. Single ended

measurements return the max value 32760 when the input is at V_{REFH}, return 0 when the input is at V_{REFL}, and scale linearly between based on the amount by which the input exceeds V_{REFL}.

23.5.1.10 Scan Mode Control (SMODE)—Bits 2-0

SMODE controls the Scan mode of the ADC module. All scan modes make use of the eight sample slots defined by the CLST1 and CLST2 registers. A scan is the process of stepping through these sample slots, converting the analog input indicated by that slot, and storing the result. Un-required slots may be disabled by writing 1 to the appropriate bits of the SDIS register.

Input pairs ANA0-1, ANA2-3, ANB0-1, and ANB0-1 may be configured as differential pairs using the CHNCFG field. When a slot in CLST*n* refers to either member of a differential pair a differential measurement on that pair will be made, otherwise a single ended measurement will be taken on that input. The details of differential and single ended measurement are described in the description of the CHNCFG field.

SMODE determines whether the slots are used to perform a sequential scan up to eight samples or two parallel scans up to four samples. SMODE controls how these scans are initiated and terminated. It also controls whether the scans are performed once or repetitively. For more details, please see **Section Figure 23-3., "Parallel Mode Operation of the ADC".**

Parallel scans may be simultaneous (SIMULT=1) or non-simultaneous. During simultaneous parallel scans A and B converters scan synchronously using one set of shared controls (CTRL1 register). During non-simultaneous (SIMULT=0) scans the A and B converters operate asynchronously with each converter using its own independent set of controls (CTRL1 for A and CTRL2 for B). Refer to the SIMULT bit description for further details.

NOTE

> The SIMULT bit only applies to parallel operating modes and is ignored during sequential operating modes.

- 000 = Once Sequential
  Upon START, or an enabled sync signal, samples are taken one at a time starting with SAMPLE0 until a first disabled sample is encountered. If no disabled sample is encountered in SDIS register, conversion concludes after SAMPLE7. If the scan is initiated by a sync signal only one scan will be completed until the converter is rearmed by writing to the CTRL1 register.

- 001 = Once Parallel
  Upon START, or an armed and enabled sync signal, converter A will capture samples 0-3 and Converter B will capture samples 4-7. By default (SIMULT=1), samples are taken simultaneously (synchronously) and scanning stops when either converter encounters a disabled sample or both converters complete all four samples. When SIMULT=0, samples are taken asynchronously and scanning stops when each converter encounters a disabled sample in its part of the SDIS register or completes all four samples. If the scan is initiated by a sync signal only one scan will be completed till the converter is rearmed by writing to the CTRL1 register. (When SIMULT=0 the B converter must be re-armed separately by writing to the CTRL2 register.)

- 010 = Loop Sequential
  Upon an initial start or enabled sync pulse, up to eight samples are taken one at a time until a

disabled sample is encountered. The process repeats until the STOP0 bit is set. While a Loop mode is running, any additional start commands or sync pulses are ignored. If Auto Standby (POWER:ASB=1) or Auto Power-Down (POWER:APD=1) is the selected Power mode control, the power-up delay defined by PUDELAY will be applied only on the first conversion.

- 011= Loop Parallel
  Upon an initial start or enabled sync pulse, converter A will capture Samples0-3 and Converter B will capture Samples4-7. Each time a converter completes its current scan, it immediately restarts its scan sequence. This continues until a STOPn bit is asserted. While a loop is running, any additional start commands or sync pulses are ignored. By default (SIMULT=1), samples are taken simultaneously (synchronously) and scanning stops when either converter encounters a disabled sample or both converters complete all four samples. When SIMULT=0, samples are taken asynchronously and scanning stops when each converter encounters a disabled sample in its part of the SDIS register or completes all four samples. If Auto Standby or Auto Power-Down is the selected power mode control, the power-up delay defined by PUDELAY will be applied only on the first conversion.

- 100 = Triggered Sequential
  Upon START, or an enabled sync signal, samples are taken one at a time starting with SAMPLE0, until a first disabled sample is encountered. If no disabled sample is encountered, conversion concludes after SAMPLE7. If external sync is enabled new scans will be started for each sync pulse that is non-overlapping with a current scan in progress.

- 101 = Triggered Parallel (default)
  Upon START, or an enabled sync signal, Converter A will convert Samples0-3 and Converter B will convert Samples4-7 in parallel. By default (SIMULT=1), samples are taken simultaneously (synchronously) and scanning stops when either converter encounters a disabled sample or both converters complete all four samples. When SIMULT=0, samples are taken asynchronously and scanning stops when each converter encounters a disabled sample in its part of the SDIS register or completes all four samples. If external sync is enabled (SYNC0=1) new scans will be started for each sync pulse as long as the ADC has completed the previous scan (STAT:CIPn=0).

- 110 = Reserved use
- 111 = Reserved use

23.5.2 Control 2 Register (CTRL2) Under Sequential Scan Modes

Operating mode dependencies occur when the ADC's scan mode (SMODE in the CTRL1 register) is set to Once Sequential, Loop Sequential, or Triggered Sequential bits 15−5 are reserved. Only the DIV field is available.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	RESERVED											DIV				
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
R/W	R/W															
Address	IPSBAR + 0x19_0001															
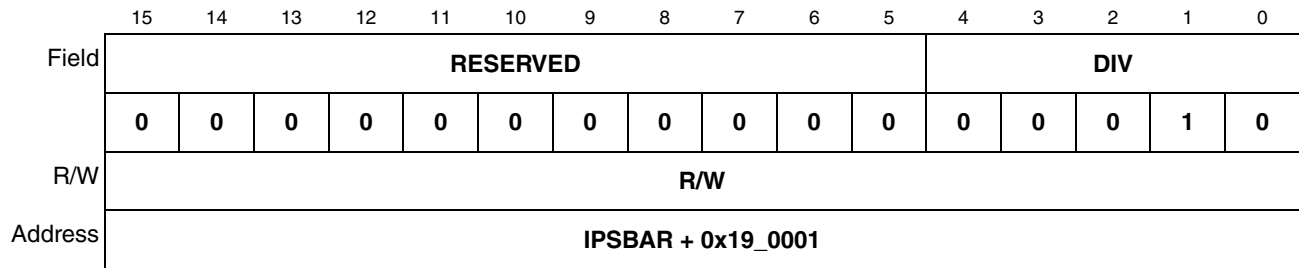
Figure 23-14. Control 2 (CTRL2) Register Under Sequential Scan Modes

23.5.2.1 Reserved—Bits 15–5

This bit field is reserved and should not be modified by writing.

23.5.2.2 Clock Divisor Select (DIV)—Bits 4–0

The divider circuit generates the ADC clock by dividing the system clock by $2 \times (DIV[4:0]+1)$. A DIV value must be chosen so the ADC clock does not exceed 5.33MHz. The following table shows ADC clock frequency based on the value of DIV for these various OCCS configurations.

Table 23-7. ADC Clock Frequency for Various Conversion Clock Sources

DIV	Divisor	ROSC Standby 400Khz	ROSC Normal 8Mhz	PLL 64 Mhz	External CLK
		200kHz Sys Clock	4MHz Sys Clock	32MHz Sys Clock	CLK/2 Sys Clock
0_0000	2	100K	2.00M	16.0M	CLK/4
0_0001	4	100K	1.00M	8.00M	CLK/8
0_0010	6	100K	500K	5.33M	CLK/12
0_0011	8	100K	250K	4.00M	CLK/16
0_0100	10	100K	125K	3.20M	CLK/20
—	—	—	—	—	—
—	—	—	—	—	—
1_1111	64	100K	62.5K	500K	CLK/128

23.5.3 Control 2 Register (CTRL2) Under Parallel Scan Modes

Operating mode dependencies of this register occur when the ADC's Scan mode (SMODE in the CTRL1 register) is set to Once Parallel, Loop Parallel, or Triggered Parallel bits 14–11, and 5 are no longer reserved. These bits are used to control the operation of Converter B.

By default, SIMULT=1 and Converter B operates together with Converter A. In this case bits 14, 13, 12, and 11 in CTRL2 do not affect Converter B operation. When SIMULT=0 and SMODE is a parallel scan bits 14–11 in CTRL2 along with the SYNC1 input are used to control the Converter B scan. In this case EOSIE1 enables the EOSI1 interrupt, signaling the end of a B converter scan. Also, the CIP1 bit in the STAT register is used to indicate a Converter B scan is active.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—	STOP1	START1	SYNC1	EOSIE1	0	0	0	0	0	SIMULT			DIV		
	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0
R/W	—	R/W	W	R/W				—						R/W		
Address	IPSBAR + 0x19_0001															

Figure 23-15. Control 2 (CTRL2) Register Under Parallel Scan Modes

23.5.3.1   Reserved—Bit 15

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

23.5.3.2   Stop (STOP1)—Bit 14

During parallel scan modes when SIMULT=0, setting STOP1 stops parallel scans in the B Converter and prevents new ones from starting. Any further SYNC1 input pulses (please see SYNC1 bit) or writes to the START1 bit are ignored until the STOP1 bit is cleared. After the ADC is in Stop mode, the B Converter Results registers can be modified by the processor. Any changes to the Result registers in Stop mode are treated as if the analog core supplied the data. Therefore, limit checking, zero crossing, and associated interrupts can occur if enabled. *This is not the same as the device's STOP mode*.

- 0 = Normal operation
- 1 = Stop command issued

23.5.3.3   Start Conversion (START1)—Bit 13

During parallel scan modes when SIMULT=0, a B converter parallel scan is started by writing 1 to the START1 bit. This is a *write-only* bit. Writing 1 to the START1 bit again will be ignored until the end of the current scan.

- 0 = No action
- 1 = Start a B Converter parallel scan

The ADC must be in a stable power configuration prior to writing the start bit.   Refer to the functional description of power modes for further details.

23.5.3.4 SYNC1 Enable (SYNC1)—Bit 12

During parallel scan modes when SIMULT=0, setting SYNC1 to 1 permits a B Converter parallel scan to be start by asserting the SYNC1 input for at least one ADC clock cycle. Any additional SYNC1 input pulses will be ignored until the end of the scan.

- 0 = B Converter parallel scan is initiated by a write to START1 bit only
- 1 = Use a SYNC1 input pulse or START1 bit to initiate a B Converter parallel scan

The ADC must be in a stable power mode prior to SYNC1 input assertion. Please refer to the functional description of power modes for further details.

In *Once* scan modes, only a first SYNC1 input pulse is honored. Subsequent SYNC1 input pulses are ignored until the SYNC1 input is re-armed by writing to the CTRL2 Register, usually by simply rewriting 1 to SYNC1. This can be done at any time, including while the scan remains in process.

23.5.3.5 Reserved—Bits 10–6

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

23.5.3.6 End Of Scan Interrupt Enable 1 (EOSIE1)—Bit 11

During parallel scan modes when SIMULT=0, this bit enables an EOSI1 interrupt to be generated upon completion of a B Converter parallel scan. For looping Scan mode, the interrupt will trigger upon the completion of each iteration of the loop.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

23.5.3.7 Simultaneous Mode (SIMULT)—Bit 5

This bit only affects parallel scan modes.

When SIMULT=1 (default value) parallel scans operate in Simultaneous mode. The scans in the A and B converter operate simultaneously and always result in pairs of simultaneous conversions in the A and B converter. START0, STOP0, SYNC0, and EOSIE0 control bits and the SYNC0 input are used to start and stop scans in both converters simultaneously. A scan ends in both converters when either converter encounters a disabled sample slot. When the parallel scan completes, the EOSI0 triggers if EOSIEN0 is set. The CIP0 status bit indicates that a parallel scan is in process.

When SIMULT=0, parallel scans in the A and B converters operate independently. The B Converter has its own independent set of the above controls (START1, STOP1, SYNC1, EOSIE1, SYNC1) designed to control its operation and report its status. Each converter's scan continues until its sample list is exhausted (four samples) or a disabled sample its part of SDIS is encountered. For looping parallel scan mode, each converter starts its next iteration as soon as the previous iteration in that converter is complete and continues until the STOP bit for that converter is asserted.

- 0 = Parallel scans done independently
- 1 = Parallel scans done simultaneously (default)

23.5.3.8 Clock Divisor Select (DIV)—Bits 4–0

The divider circuit generates the ADC clock by dividing the system clock by $2 \times (DIV[4:0]+1)$. A DIV value must be chosen so the ADC clock does not exceed 5.33Mhz. Table 23-8 shows ADC clock frequency based on the value of DIV for these various OCCS configurations.

Table 23-8. ADC Clock Frequency for Various Conversion Clock Sources

DIV	Divisor	ROSC Standby 400Khz	ROSC Normal 8Mhz	PLL 64 Mhz	External CLK
		200kHz Sys Clock	4MHz Sys Clock	32MHz Sys Clock	CLK/2 Sys Clock
0_0000	2	100K	2.00M	16.0M	CLK/4
0_0001	4	100K	1.00M	8.00M	CLK/8
0_0010	6	100K	500K	5.33M	CLK/12
0_0011	8	100K	250K	4.00M	CLK/16
0_0100	10	100K	125K	3.20M	CLK/20
—	—	—	—	—	—
—	—	—	—	—	—
1_1111	64	100K	62.5K	500K	CLK/128

23.5.4 Zero Crossing Control Register (ZXCTRL)

The ADC Zero Crossing Control (ZXCTRL) register provides the ability to monitor the selected channels and determine the direction of zero crossing triggering the optional interrupt. Zero crossing logic monitors only the sign change between current and previous sample. ZCE0 bit monitors the sample stored in RSLT0, ZCE1 bit monitors RSLT1, ZCE7 bit monitors RSLT7. When the Zero Crossing is disabled for a selected result register, sign changes are not monitored or updated in the ZXSTAT register.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	ZCE7		ZCE6		ZCE5		ZCE4		ZCE3		ZCE2		ZCE1		ZCE0	
Reset	0000_0000_0000_0000															
R/W	R/W															
Address	IPSBAR + 0x19_0002															

Figure 23-16. Zero Crossing Control (ZXCTRL) Register

23.5.4.1 Zero Crossing Enable *n* (ZCE*n*)—Bits 15–0

For each channel, *n*, setting the ZCE*n* field allows detection of the indicated zero crossing condition, provided the corresponding offset register (OFFST*n*) has a value *offset*, $0 < offset < 0x7FF8$.

- 00 = Zero crossing disabled

- 01 = Zero crossing enabled for positive to negative sign change
- 10 = Zero crossing enabled for negative to positive sign change
- 11 = Zero crossing enabled for any sign change

23.5.5 Channel List 1 and 2 Registers (CLST1 and CLST2)

The Channel List Register contains an ordered list of the analog input channels to be converted when the next scan is initiated. If all samples are enabled in the SDIS Register, a sequential scan of inputs proceeds in order of SAMPLE0 through SAMPLE7. If one of the Parallel Sampling modes is selected instead, the Converter A sampling order is SAMPLE0-3 and the Converter B sampling order is SAMPLE4-7.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field		SAMPLE3				SAMPLE2				SAMPLE1				SAMPLE0		
Reset	0011_0010_0001_0000															
R/W	R	R/W			R	R/W			R	R/W			R	R/W		
Address	IPSBAR + 0x19_0003															

Figure 23-17. Channel List 1 (CLST1) Register

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field		SAMPLE7				SAMPLE6				SAMPLE5				SAMPLE4		
Reset	0111_0110_0101_0100															
R/W	R	R/W			R	R/W			R	R/W			R	R/W		
Address	IPSBAR + 0x19_0004															

Figure 23-18. Channel List 2 (CLST2) Register

23.5.5.1 Reserved—Bits 15, 11, 7 and 3

These bits are reserved or are not implemented. They are read as 0 and cannot be modified by writing.

23.5.5.2 SAMPLE *n* (SAMPLE4)—Bits 2, 1, and 0

The value of the SAMPLE*n* field is used to select the input channel to be sampled.

Table 23-9. ADC Input Conversion for Sample Bits

SAMPLE*n*[2:0]			ADC Input Pins Selected	
Sequential Mode	Parallel Mode			
n=0,1,2,...,7	*n*=0,1,2,3 (Conv. A)	*n*=4,5,6,7 (Conv. B)	Single Ended	Differential

Table 23-9. ADC Input Conversion for Sample Bits

000	000		ANA0	ANA0+, ANA1−
001	001		ANA1	
010	010		ANA2	ANA2+, ANA3−
011	011		ANA3	
100		100	ANB0	ANB0+, ANB1−
101		101	ANB1	
110		110	ANB2	ANB2+, ANB3−
111		111	ANB3	

In sequential modes, the sample slots are converted in order from SAMPLE0 to SAMPLE7. Analog input pins can be sampled in any order, including sampling the same input pin more than once.

In Parallel modes, Converter A processes sample slots SAMPLE0 through SAMPLE3 while Converter B processes sample slots SAMPLE4 through SAMPLE7. Since Converter A only has access to analog inputs ANA0 through ANA3, sample slots SAMPLE0-3 should only contain binary values between 000 and 011. Likewise, since Converter B only has access to analog inputs ANB0 through ANB3, sample slots SAMPLE4-7 should only contain binary values between 100 and 111. No damage will occur if this constraint is violated but results are undefined.

When inputs are configured as differential pairs, a reference to either analog input in a differential pair by a sample slot implies a differential measurement on the pair. The details of single ended and differential measurement are described under the CHNCFG field. Sample slots are disabled using the SDIS register.

23.5.6 Sample Disable Register (SDIS)

This register is an extension to the CLST1and CLST2, providing the ability to enable only the desired samples programmed in the SAMPLE0–SAMPLE7. At reset all samples are enabled. For example, if in a Sequential mode and bit DS5 is set to 1, SAMPLE0 through SAMPLE4 are sampled. However, if Parallel mode is selected and bits DS5 or DS1 are set to 1, only SAMPLE0 and SAMPLE4 are sampled.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	RESERVED								DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	—								R/W							
Address	IPSBAR + 0x19_0005															

Figure 23-19. Sample Disable (SDIS) Register

23.5.6.1 Reserved—Bits 15–8

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

23.5.6.2 Disable Sample (DS*n*)—Bits 7–0

The respective SAMPLE*n* field can be enabled or disabled where $n = 0–7$.

- 0 = Enable SAMPLE*n*
- 1 = Disable SAMPLE*n* and all subsequent samples. Which samples are actually disabled will depend on the conversion mode, sequential/parallel, and the value of SIMULT.

23.5.7 Status Register (STAT)

This register provides the current status of the ADC module. RDY*n* bits are cleared by reading their corresponding Result (RSLT*n*) registers. HLMTI and LLMTI bits are cleared by writing 1 to each asserted bit in the ADC Limit Status (LIMSTAT) register. Likewise, the ZCI bit, bit-10, is cleared by writing 1 to each asserted bit in the ADC Zero Crossing Status (ZXSTAT) register. The EOSI*n* bits are cleared by writing 1 to them. Please see **Figure 23-19** for more information regarding the operation of interrupts.

Except for CIP0 and CIP1 all bits in the STAT Register are *sticky*. Once set to a 1 state, they require some specific action to clear them. They are not cleared automatically on the next scan sequence.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	CIP0	CIP1	0	EOSI1	EOSI0	ZCI	LLMTI	HLMTI	RDY7	RDY6	RDY5	RDY4	RDY3	RDY2	RDY1	RDY0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R		—	R/W		R										
Address	IPSBAR + 0x19_0006															

Figure 23-20. Status (STAT) Register

23.5.7.1 Conversion in Progress 0 (CIP0)—Bit 15

This bit indicates when a scan is in progress.

- 0 = Idle state
- 1 = A scan cycle is in progress. The ADC will ignore all sync pulses or start commands.

This bit supports any sequential scan or parallel scan with SIMULT=1. When executing a parallel scan with SIMULT = 0 this bit services the scan of Converter A while the CIP1 bit services the scan of Converter B.

23.5.7.2 Conversion in Progress 1 (CIP1)—Bit 14

This bit indicates when a scan is in progress.

- 0 = Idle state

- 1 = A scan cycle is in progress. The ADC will ignore all sync pulses or start commands

This refers only to a B Converter scan in non-simultaneous (SIMULT=0) parallel scan modes.

23.5.7.3 Reserved—Bit 13

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

23.5.7.4 End of Scan Interrupt 1 (EOSI1)—Bit 12

This bit indicates whether a scan of analog inputs have been completed since the last read of the STAT register or since a reset. The EOSI1 bit is cleared by writing 1 to it. This bit cannot be set by software.

- 0 = A scan cycle has not been completed, no end of scan IRQ pending
- 1 = A scan cycle has been completed, end of scan IRQ pending

In Looping Scan modes, this interrupt is triggered at the completion of each iteration of the loop. This interrupt is triggered only by the completion of a B Converter scan in non-simultaneous (SIMULT=0) Parallel Scan modes. In this case the EOSI0 interrupt is triggered when Converter A completes its scan.

23.5.7.5 End of Scan Interrupt 0 (EOSI0)—Bit 11

This bit indicates whether a scan of analog inputs has been completed since the last read of the STAT register or since a reset. The EOSI0 bit is cleared by writing 1 to it. This bit cannot be set by software. EOSI0 is the preferred bit to poll for scan completion if interrupts are not enabled.

- 0 = A scan cycle has not been completed, no end of scan IRQ pending
- 1 =A scan cycle has been completed, end of scan IRQ pending

In Looping Scan modes, this interrupt is triggered at the completion of each iteration of a loop.

This interrupt is triggered upon the completion of any sequential scan or parallel scan with SIMULT=1. When executing parallel scans with SIMULT = 0 this interrupt is triggered when Converter A completes it scan while the EOSI1interrupt services Converter B.

23.5.7.6 Zero Crossing Interrupt (ZCI)—Bit 10

This bit is asserted at the completion of an individual conversion experiencing a zero crossing enabled in ADC Zero Crossing Control (ZXCTRL) register. The bit is set as soon as an enabled zero crossing event occurs rather than at the end of the ADC scan.

The ZCI bit is cleared by writing 1 to all active ZCS[7:0] bits in the ZXSTAT register.

- 0 = No ZCI interrupt request
- 1 = Zero crossing encountered; IRQ pending if ZCIE is set

23.5.7.7 Low Limit Interrupt (LLMTI)—Bit 9

If any Low Limit (LOLIMn) register is enabled by having a value other than $0000, low limit checking is enabled. This bit is set at the completion of an individual conversion which may or may not be the end of a scan.

The LLMTI bit is cleared by writing 1 to all active LLS[7:0] bits in the LIMSTAT register.

- 0 = No low limit interrupt request
- 1 = Low limit exceeded, IRQ pending if LLMTIE is set

23.5.7.8 High Limit Interrupt (HLMTI)—Bit 8

If any High Limit (HILIMn) register is enabled by having a value other than 0x7FF8, high limit checking is enabled. This bit is set at the completion of an individual conversion which may or may not be the end of a scan.

The HLMTI bit is cleared by writing 1 to all active HLS[7:0] bits in the LIMSTAT register.

- 0 = No high limit interrupt request
- 1 = High limit exceeded, IRQ pending if HLMTIE is set

23.5.7.9 Ready Sample 7−0 (RDYn)—Bits 7−0

These bits indicate samples seven through zero are ready to be read. The RDYn bits are set as the individual channel conversions are completed and stored in a RSLTn register. These bits are cleared after a read from the corresponding ADC Results (RSLTn) Register. If polling the RDYn bits to determine if a particular sample is executed, care should be taken not to start a new scan until all enabled samples are completed.

NOTE

RDYn bits can be cleared when the debugger reads the corresponding Results register during a debug session.

- 0 = Sample not ready or has been read
- 1 = Sample ready to be read

Figure 23-21 illustrates how five interrupts sources are combined into three entries in the interrupt vector table.
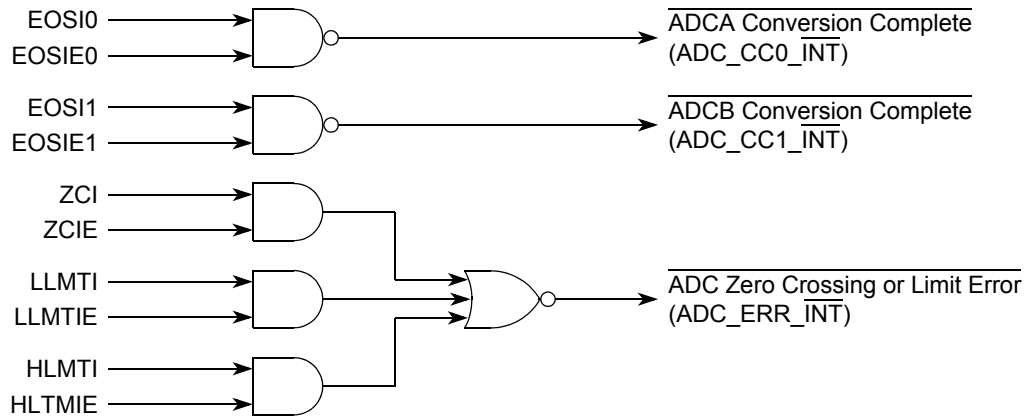
Figure 23-21. ADC Interrupt Sources

23.5.8 Limit Status Register (LIMSTAT)

The ADC Limit Status (LIMSTAT) register latches in the result of the comparison between the result of the sample in the RSLT*n* register and the respective Limit Register, HILIM*n* or LOLIM*n*.

Here is an example. If the result for RSLT0 is greater than the value programmed into the HILIM0, then set the HLS0 bit to 1. An interrupt is generated if the HLMTIE bit is set in CTRL1. These bits are *sticky*. Once set, the bits require a specific modification to clear them. They are not cleared automatically by subsequent conversions. A bit may only be cleared by writing a value of one to that specific bit.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	HLS7	HLS6	HLS5	HLS4	HLS3	HLS2	HLS1	HLS0	LLS7	LLS6	LLS5	LLS4	LLS3	LLS2	LLS1	LLS0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W															
Address	IPSBAR + 0x19_0007															

Figure 23-22. Limit Status (LIMSTAT) Register

23.5.9 Zero Crossing Status Register (ZXSTAT)

The ADC Zero Crossing Status (ZXSTAT) register latches in the result of the comparison between the current result of the sample and the previous result of the same results register. For example, if the result for the channel programmed in SAMPLE0 changes sign from the previous conversion and the respective ZCE bit in ZXCTRL register is set to 11b (any edge change) then set the ZCS0 bit to 1. An interrupt is generated if the ZCIE bit is set in the CTRL1 register. These bits are *sticky*. Once set, they require a write to clear them. They are not cleared automatically by subsequent conversions. A bit may only be cleared by writing a value of 1 to that specific bit.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field									ZCS7	ZCS6	ZCS5	ZCS4	ZCS3	ZCS2	ZCS1	ZCS0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W				—								R/W				
Address							IPSBAR + 0x19_0008									

Figure 23-23. Zero Crossing Status (ZXSTAT) Register

23.5.9.1   Reserved—Bits 15–8

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

23.5.9.2   Zero Crossing Status (ZCS[7:0])—Bits 7–0

The zero crossing condition is determined by examining the ADC value after it has been adjusted by the offset for the result register. Please see Figure 23-7. Each bit of the register is cleared by writing 1 to that register bit.

- 0 = a. A sign change did not occur in a comparing the current RSLTn value and the previous RSLTn value,

  or

  b. Zero crossing control is disabled for sample n in the ADC Zero Crossing Control (ZXCTRL) register

- 1 = In a comparison between the current channelx result and the previous channelx result, a sign change condition occurred as defined in the ADC Zero Crossing Control (ZXCTRL) register

23.5.10   Result 0-7 Registers (RSLT0–7)

The eight Result Registers contain the converted results from a scan. The SAMPLE0 result is loaded into RSLT0 Register, SAMPLE1 result in RSLT1 Register, and so on. In a simultaneous Parallel Scan mode, the first channel pair, designated by SAMPLE0 and SAMPLE4 in register LIST1/2, is stored in RSLT0 and RSLT4, respectively.

When writing to this register, only the RSLT portion of the value written is used. This value is modified as shown in Figure 23-7 and the result of the subtraction is stored. The SEXT bit is only set as a result of this subtraction and is not directly determined by the value written.

ADC Result Register 0 − Address: ADC_BASE + $9
ADC Result Register 1 − Address: ADC_BASE + $A
ADC Result Register 2 − Address: ADC_BASE + $B
ADC Result Register 3 − Address: ADC_BASE + $C
ADC Result Register 4 − Address: ADC_BASE + $D
ADC Result Register 5 − Address: ADC_BASE + $E

ADC Result Register 6 − Address: ADC_BASE + $F

ADC Result Register 7 − Address: ADC_BASE + $10

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	SEXT	RSLT												0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R	R/W												—		
Address	IPSBAR + 0x19_0009 - 0x19_0010															

Figure 23-24. Result (RSLT0−7) Registers

23.5.10.1 Sign Extend (SEXT)—Bit 15

SEXT is the sign-extend bit of the result. When the SEXT bit is set to 1, it implies a negative result. When the SEXT bit is set to 0, it implies a positive result. If only positive results are required, then the respective ADC Offset (OFFSTn) register must be set to a value of 0.

23.5.10.2 Digital Result of the Conversion (RSLT)—Bits 14−3

RSLT can be interpreted as either a signed integer or a signed fixed point (fractional) number. As a fixed point number, the RSLT can be used directly. As a signed integer, one has the option to right shift with sign extend (ASR) three places to fit it into the range [0,4095]. Or one can accept the number as presented in the register, knowing there are missing codes because the lower three LSBs are always zero.

Negative results, SEXT = 1, are always presented in twos complement format. If it is a requirement of an application, the Result registers always be positive, the Offset register (OFFSTn) must always be set to 0.

The interpretation of the numbers programmed into the ADC Limit and Offset (LOLIMn, HILIMn, and OFFSTn) registers should match your interpretation of the result register.

23.5.10.3 Test Data (Test_Data)—Bits 14−3

When the ADC is stopped or in Power-Down mode this field can be written by accessing the register in the memory map. Please see Section 23.4.3, "ADC Data Processing more information.

23.5.10.4 Reserved—Bits 2−0

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

23.5.11 Low and High Limit Registers (LOLIM0-7 and HILIM0-7)

Each ADC sample is compared against the values in the Limit Registers. The comparison is based upon the raw conversion value before the offset correction is applied. Refer to Figure 23-7. ADC Limit Registers (LOLIMn and HILIM$n)$ correspond to Results (RSLTn) registers. The High Limit register is used for the comparison of *Result > High Limit*. The Low Limit register is used for the comparison of

Result < Low Limit. The limit checking can be disabled by programming the respective limit register with 0x7FF8 for the high limit and 0x0000 for the low limit. At reset, limit checking is disabled.

ADC Low Limit Register 0 − Address: ADC_BASE + $11

ADC Low Limit Register 1 − Address: ADC_BASE + $12

ADC Low Limit Register 2 − Address: ADC_BASE + $13

ADC Low Limit Register 3 − Address: ADC_BASE + $14

ADC Low Limit Register 4 − Address: ADC_BASE + $15

ADC Low Limit Register 5 − Address: ADC_BASE + $16

ADC Low Limit Register 6 − Address: ADC_BASE + $17

ADC Low Limit Register 7 − Address: ADC_BASE + $18

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—							LLMT						0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	—							R/W							—	
Address	IPSBAR + 0x19_0011 - 0x19_0018															

Figure 23-25. Low Limit Register (LOLIM0−7)

ADC High Limit Register 0−Address: ADC_BASE + $19

ADC High Limit Register 1−Address: ADC_BASE + $1A

ADC High Limit Register 2−Address: ADC_BASE + $1B

ADC High Limit Register 3−Address: ADC_BASE + $1C

ADC High Limit Register 4−Address: ADC_BASE + $1D

ADC High Limit Register 5−Address: ADC_BASE + $1E

ADC High Limit Register 6−Address: ADC_BASE + $1F

ADC High Limit Register 7−Address: ADC_BASE + $20

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—							HLMT						0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	—							R/W							—	
Address	IPSBAR + 0x19_0019 - 0x19_0020															

Figure 23-26. High Limit Register (HILIM0–7)

23.5.12 Offset Registers (OFFST0–7)

Value of the Offset (OFFST*n*) register is used to correct the ADC result before it is stored in the RSLT*n* registers.

ADC Offset Register 0–Address:	ADC_BASE + $21
ADC Offset Register 1–Address:	ADC_BASE + $22
ADC Offset Register 2–Address:	ADC_BASE + $23
ADC Offset Register 3–Address:	ADC_BASE + $24
ADC Offset Register 4–Address:	ADC_BASE + $25
ADC Offset Register 5–Address:	ADC_BASE + $26
ADC Offset Register 6–Address:	ADC_BASE + $27
ADC Offset Register 7–Address:	ADC_BASE + $28

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—	OFFSET												0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	—	R/W												—		
Address	IPSBAR + 0x19_0021 - 0x19_0028															

Figure 23-27. Offset 0-7 (OFFST0-7) Registers

The offset value is subtracted from the ADC result. In order to obtain unsigned results, the respective Offset register should be programmed with a value of $0000, thus giving a result range of $0000 to $7FF8.

23.5.13 Power Control Register (PWR)

This register controls the power management features of the ADC module. There are manual power-down control bits for the two ADC converters and the shared voltage reference generator. There are also five distinct power modes. The following terms are used to describe power modes and their related controls.

1. Powered down state
   Each converter and the voltage reference generator can individually be put into a powered down state. When powered down, the unit consumes no power. Results of scans referencing a powered down converter are undefined. The voltage reference generator and at least one converter must be powered up to use the ADC module.

2. Manual power-down controls
   Each converter and the voltage reference generator have a manual power control bit capable of forcing that component into the power down state. Also, each converter and the voltage reference generator can be powered up/down automatically as part of ADC operation.

3. Idle state
   The ADC module is idle when neither of the two converters has a scan in process.

4. Active state

The ADC module is active when at least one of the two converters has a scan in process.

5. Current mode

- Normal current mode is used to power the converters at clock rates above 100kHz.

- Standby Current mode uses less power and is engaged only when the ADC clock is at 100kHz. The current mode active does not affect the number of ADC clock cycles required to do a conversion or the accuracy of a conversion. The ADC module may change the current mode when idle as part of the power saving strategy. Both converters will be in the same current mode at all times.

In addition to the power modes, Startup delay is defined as:

- Auto Power-Down and Auto Standby Power modes cause a startup delay when the ADC module goes between the idle and active states to allow time to switch clocks or power configurations. The number of ADC clocks used in the startup delay is defined by the PUDELAY field.

See the discussion of power modes in the Functional Description Section 23.4, "Functional Description" for details of the five power modes and how to configure them. See Section 23.4.7, "ADC Clock" for a more detailed description of the clocking system and the control of current mode.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	ASB	—	—	PSTS2	PSTS1	PSTS0	PUDELAY						APD	PD2	PD1	PD0
Reset	0	0	0	0	0	0	0	0	1	1	0	1	0	1	1	1
R/W	R/W	—		R			R/W									
Address	IPSBAR + 0x19_0029															

Figure 23-28. Power Control (PWR) Register

23.5.13.1 Auto Standby (ASB)—Bit 15

The ASB bit selects Auto Standby mode. ASB is ignored if APD is 1. When the ADC is idle, Auto Standby mode selects the standby clock as the ADC clock source and puts the converters into Standby Current mode. At the start of any scan, the conversion clock is selected as the ADC clock and a delay of PUDELAY ADC clock cycles is imposed for current levels to stabilize. After this delay, the ADC will initiate the scan. When the ADC returns to the idle state, the standby clock is again selected and the converters revert to the standby current state.

- 0 = Auto standby mode disabled
- 1 = Auto standby mode enabled

23.5.13.2 Reserved—Bits 14–13

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

23.5.13.3 Voltage Reference Power Status 2 (PSTS2)—Bit 12

PSTS2 is a *read-only* bit. It simply reflects whether the voltage reference circuit is currently enabled.

- 0 = Voltage reference circuit is currently powered up
- 1 = Voltage reference circuit is currently powered down

23.5.13.4 Converter B Power Status 1 (PSTS1)—Bit 11

PSTS1 is a *read-only* bit. It is asserted immediately following a write of 1 to PD1. It is deasserted PUDELAY ADC clock cycles after writing 0 to PD1 if APD is 0. This bit can be read as a status bit to determine when the ADC is ready for operation. During Auto Power-Down mode, this bit indicates the current powered state of Converter B.

- 0 = ADC Converter B is currently powered up
- 1 = ADC Converter B is currently powered down

23.5.13.5 Converter A Power Status 0 (PSTS0)—Bit 10

PSTS0 is a *read-only* bit. It is asserted immediately following a write of 1 to PD0. It is deasserted PUDELAY ADC clock cycles after writing 0 to PD0 if APD is 0. This bit can be read as a status bit to determine when the ADC is ready for operation. During Auto Power-Down mode, this bit indicates the current powered state of Converter A.

- 0 = ADC Converter A is currently powered up
- 1 = ADC Converter A is currently powered down

23.5.13.6 Power-Up Delay (PUDELAY)—Bits 9–4

This 6-bit field determines the number of ADC clocks provided to power-up an ADC converter (after setting PD0 or PD1 to 0) before allowing a scan to start. It also determines the number of ADC clocks of delay provided in Auto Power-Down (APD) and Auto Standby (ASB) modes between when the ADC goes from the idle to active state and when the scan is allowed to start. The default value is 13 ADC clocks. Accuracy of the initial conversions in a scan will be degraded if PUDELAY is set to too small a value.

NOTE

PUDELAY defaults to a value typically sufficient for any power mode. The latency of a scan can be reduced by reducing PUDELAY to the lowest value for which accuracy is not degraded. Please refer to the *Device Data Sheet* for further details.

23.5.13.7 Auto Power-Down (APD)—Bit 3

Auto Power-Down mode powers down converters when not in use for a scan. APD takes precedence over ASB. When a scan is started in APD mode, a delay of PUDELAY ADC clock cycles is imposed during which the needed converter(s), if idle, are powered up. The ADC will then initiate a scan equivalent to when APD is not active. When the scan is completed, the converter(s) are powered down again.

- 0 = Auto Power-Down mode is not active
- 1 = Auto Power-Down mode is active

NOTE

If ASB or APD is asserted while a scan is in progress, that scan is unaffected and the ADC will wait to enter its low power state until after all conversions are complete and both ADCs are idle.

NOTE

ASB and APD are not useful in looping modes. The continuous nature of scanning means the low power state can never be entered.

23.5.13.8 Power-Down Control for Voltage Reference Circuit 2 (PD2)—Bit 2

This bit controls the power-down of the ADC's voltage reference current.

- 0 = Manually Power-Up voltage reference circuit
- 1 = Power-Down voltage reference circuit is controlled by PD0 and PD1 (default)

The voltage reference circuit is shared by both converters. When PD2=1 the voltage reference will be activated whenever PD1 or PD0 are powered up.   It is not usually necessary to modify this bit, since powering down both Converter A and Converter B will automatically power-down the voltage reference.

23.5.13.9 Manual Power-Down for Converter B (PD1)—Bit 1

This bit forces ADC Converter B to power-down.

- 0 = Power-Up ADC Converter B
- 1 = Power-Down ADC Converter B

Asserting PD1 powers down Converter B immediately. The results of a scan using Converter B will be invalid while PD1 is asserted. When PD1 is cleared, Converter B is either continuously powered up (APD = 0) or automatically powered up when needed (APD=1).

When clearing PD1 in any power mode except Auto Power-Down (APD=1), wait PUDELAY ADC clock cycles before initiating a scan to stabilize power levels within the converter. The PSTS1 bit can be polled to determine when the PUDELAY time has elapsed. Failure to follow this procedure can result in loss of accuracy of the first two samples.

23.5.13.10 Manual Power-Down for Converter A (PD0)—Bit 0

This bit forces ADC Converter A to power-down.

- 0 = Power-Up ADC Converter A
- 1 = Power-Down ADC Converter A

Asserting PD0 powers down Converter A immediately. The results of a scan using Converter A will be invalid while PD0 is asserted. When PD0 is cleared, Converter A is either continuously powered up (APD = 0) or automatically powered up when needed (APD=1).

When clearing PD0 in any power mode except Auto Power-Down (APD=1), wait PUDELAY ADC clock cycles before initiating a scan to stabilize power levels within the converter. The PSTS0 bit can be polled

to determine when the PUDELAY time has elapsed. *Failure to follow this procedure can result in loss of accuracy of the first two samples.*

23.5.14 Voltage Reference Register (VREF)

In earlier series this register supported ADC calibration and had a different name. Improvements in ADC performance have eliminated the need for on-chip calibration support, hence the new name.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	SEL_VREFH	SEL_VREFL	RESERVED													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	—													
Address	IPSBAR + 0x19_002A															

Figure 23-29. Voltage Reference (VREF) Register

23.5.14.1 Select V_{REFH} Source (SEL_VREFH)—Bit 15

This bit selects the source of the V_{REFH} reference for conversions.
- 0 = Internal VR_X
- 1 = ANA2

23.5.14.2 Select V_{REFL} Source (SEL_VREFL)—Bit 14

This bit selects the source of the V_{REFL} reference for conversions.
- 0 = Internal VR_X
- 1 = ANB2

23.5.14.3 Reserved—Bits 13–0

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

Preliminary

Chapter 24
Pulse Width Modulation (PWM) Module

24.1 Introduction

This chapter describes the configuration and operation of the pulse width modulation (PWM) module. It includes a block diagram, programming model, and functional description.

24.1.1 Overview

The PWM module shown in Figure 24-1, generates a synchronous series of pulses having programmable period and duty cycle. With a suitable low-pass filter, the PWM can be used as a digital-to-analog converter.
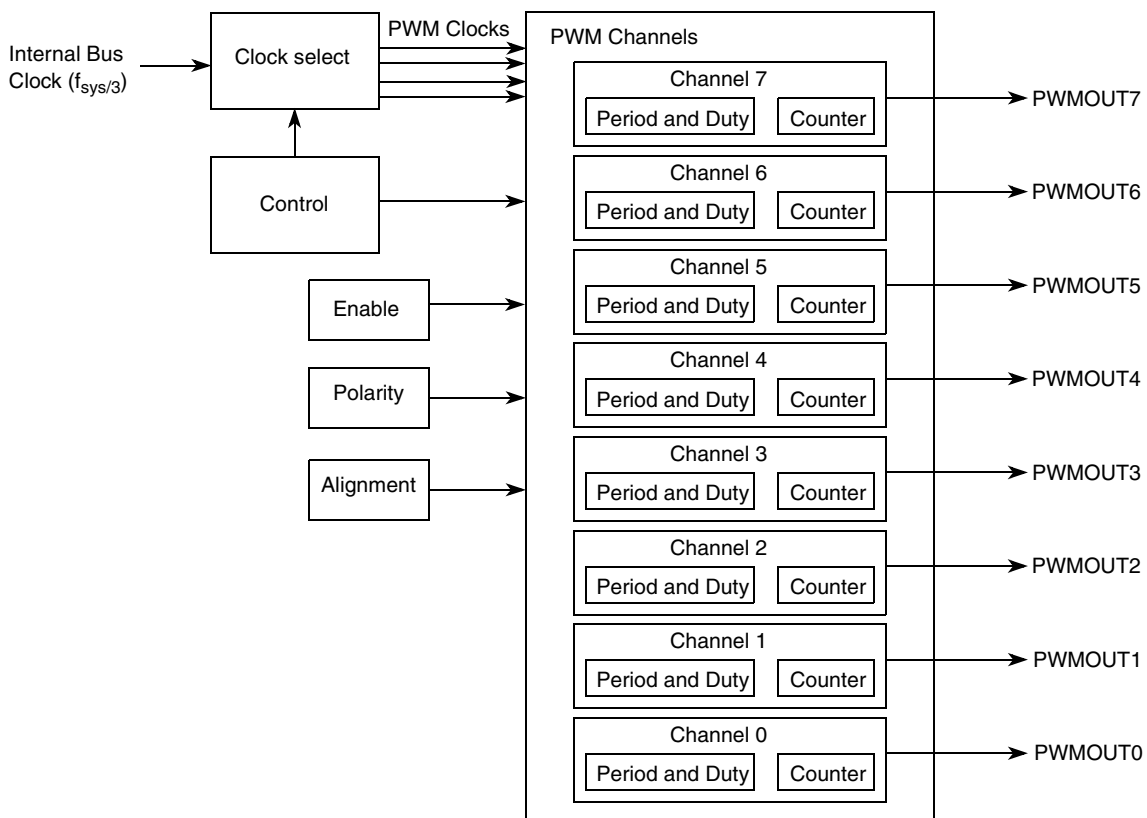
Figure 24-1. PWM Block Diagram

Summary of the main features include:

- Double-buffered period and duty cycle

- Left- or center-aligned outputs
- Eight independent PWM modules
- Byte-wide registers provide programmable duty cycle and period control
- Four programmable clock sources

NOTE

The GPIO module must be configured to enable the peripheral function of the appropriate pins (refer to Chapter 11, "General Purpose I/O Module") prior to configuring the PWM module.

24.2 Memory Map/Register Definition

This section describes the registers and control bits in the PWM module. There are eight independent PWM modules, each with its own control and counter registers. The memory map for the PWM is shown in below.

Table 24-1. PWM Memory Map

Address[1,2]	Register	Access	Reset Value	Section/Page
0x001B_0000	PWM Enable Register (PWME)	R/W	0x00	24.2.1/24-3
0x001B_0001	PWM Polarity Register (PWMPOL)	R/W	0x00	24.2.2/24-4
0x001B_0002	PWM Clock Select Register (PWMCLK)	R/W	0x00	24.2.3/24-4
0x001B_0003	PWM Prescale Clock Select Register (PWMPRCLK)	R/W	0x00	24.2.4/24-5
0x001B_0004	PWM Center Align Enable Register (PWMCAE)	R/W	0x00	24.2.5/24-6
0x001B_0005	PWM Control Register (PWMCTL)	R/W	0x00	24.2.6/24-7
0x001B_0008	PWM Scale A Register (PWMSCLA)	R/W	0x00	24.2.7/24-8
0x001B_0009	PWM Scale B Register (PWMSCLB)	R/W	0x00	24.2.8/24-8
0x001B_000C ... 0x001B_0013	PWM Channel n Counter Register (PWMCNTn)	R/W	0x00	24.2.9/24-9
0x001B_0014 ... 0x001B_001B	PWM Channel n Period Register (PWMPERn)	R/W	0xFF	24.2.10/24-10
0x001B_001C ... 0x001B_0023	PWM Channel n Duty Register (PWMDTYn)	R/W	0xFF	24.2.11/24-11
0x001B_0024	PWM Shutdown Register (PWMSDN)	R/W	0x00	24.2.12/24-11

[1] Addresses not assigned to a register and undefined register bits are reserved for expansion. Write accesses to these reserved address spaces and reserved register bits have no effect.

[2] 32-bit access to any of these registers will result in a bus transfer error (see Section 11.2.6, "SCM Interrupt Register (SCMIR)").

24.2.1 PWM Enable Register (PWME)

Each PWM channel has an enable bit (PWME*n*) to start its waveform output. While in run mode, if all eight PWM output channels are disabled (PWME[7:0] = 0) the prescaler counter shuts off for power savings. See Section 24.3.2.1, "PWM Enable" for more information.

Address: 0x001B_0000 (PWME)   Access: User Read/Write

	7	6	5	4	3	2	1	0
R W	PWME7	PWME6	PWME5	PWME4	PWME3	PWME2	PWME1	PWME0
Reset:	0	0	0	0	0	0	0	0
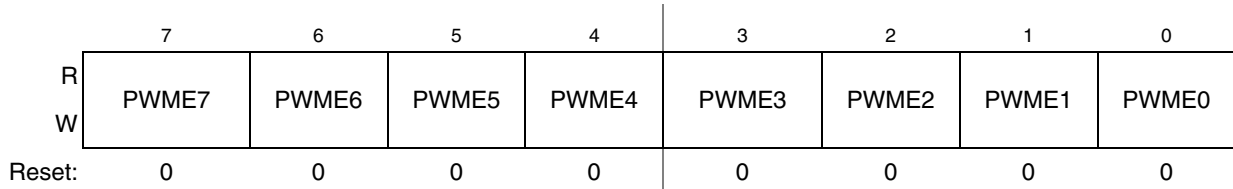
Figure 24-2. PWM Enable Register (PWME)

Table 24-2. PWME Field Descriptions

Field	Description
7 PWME5	PWM channel 7 output enable. If enabled, the PWM signal becomes available at PWMOUT7 when its corresponding clock source begins its next cycle. 0 PWM output disabled 1 PWM output enabled
6 PWME4	PWM channel 6 output enable. If enabled, the PWM signal becomes available at PWMOUT6 when its corresponding clock source begins its next cycle. If PWMCTL[CON67] is set, then this bit has no effect and PWMOUT6 is disabled. 0 PWM output disabled 1 PWM output enabled
5 PWME5	PWM channel 5 output enable. If enabled, the PWM signal becomes available at PWMOUT5 when its corresponding clock source begins its next cycle. 0 PWM output disabled 1 PWM output enabled
4 PWME4	PWM channel 4 output enable. If enabled, the PWM signal becomes available at PWMOUT4 when its corresponding clock source begins its next cycle. If PWMCTL[CON45] is set, then this bit has no effect and PWMOUT4 is disabled. 0 PWM output disabled 1 PWM output enabled
3 PWME3	PWM channel 3 output enable. If enabled, the PWM signal becomes available at PWMOUT3 when its corresponding clock source begins its next cycle. 0 PWM output disabled 1 PWM output enabled
2 PWME2	PWM channel 2 output enable. If enabled, the PWM signal becomes available at PWMOUT2 when its corresponding clock source begins its next cycle. If PWMCTL[CON23] is set, then this bit has no effect and PWMOUT2 is disabled. 0 PWM output disabled 1 PWM output enabled, if PWMCTL[CON23]=0

MCF5213 Reference Manual, Rev. 1.1

Preliminary

Table 24-2. PWME Field Descriptions (continued)

Field	Description
1 PWME1	PWM channel 1 output enable. If enabled, the PWM signal becomes available at PWMOUT1 when its corresponding clock source begins its next cycle. 0 PWM output disabled 1 PWM output enabled
0 PWME0	PWM channel 0 output enable. If enabled, the PWM signal becomes available at PWMOUT0 when its corresponding clock source begins its next cycle. If PWMCTL[CON01] is set, then this bit has no effect and PWMOUT0 is disabled. 0 PWM output disabled 1 PWM output enabled, if PWMCTL[CON01]=0

24.2.2 PWM Polarity Register (PWMPOL)

The starting polarity of each PWM channel waveform is determined by the associated PWMPOL[PPOLn] bit. If the polarity is changed while a PWM signal is being generated, a truncated or stretched pulse can occur during the transition.

Address: 0x001B_0001 (PWMPOL)   Access: User Read/Write

	7	6	5	4	3	2	1	0
R W	PPOL7	PPOL6	PPOL5	PPOL4	PPOL3	PPOL2	PPOL1	PPOL0
Reset:	0	0	0	0	0	0	0	0
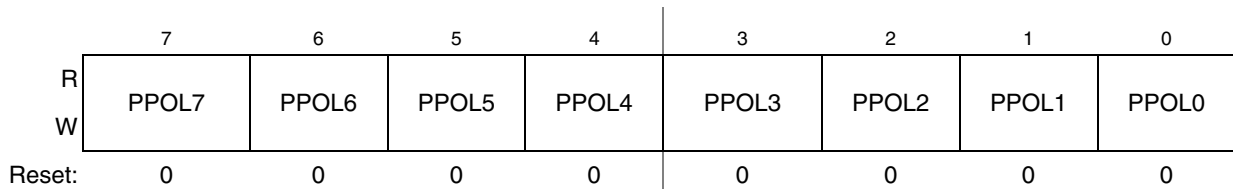
Figure 24-3. PWM Polarity Register (PWMPOL)

Table 24-3. PWMPOL Field Descriptions

Field	Description
7–0 PPOLn	PWM channel n polarity. The even-numbered channels' polarity has no effect when the corresponding PWMCTL[CON$n(n+1)$] bit is set. For example, if PWMCTL[CON01] = 1 then PWMPOL[PPOL0] has no affect. 0 PWM channel n output is low at the beginning of the period, then goes high when the duty count is reached 1 PWM channel n output is high at the beginning of the period, then goes low when the duty count is reached

24.2.3 PWM Clock Select Register (PWMCLK)

Each PWM channel has the capability of selecting one of two clocks. For channels 0, 1, 4, and 5 the clock choices are clock A or SA. For channels 2, 3, 6, and 7 the choices are clock B or SB. The clock selection is done with the below PWMCLK[PCLKn] control bits. If a clock select is changed while a PWM signal is being generated, a truncated or stretched pulse can occur during the transition.

Address: 0x001B_0002 (PWMCLK)   Access: User Read/Write

	7	6	5	4	3	2	1	0
R	PCLK7	PCLK6	PCLK5	PCLK4	PCLK3	PCLK2	PCLK1	PCLK0
W								
Reset:	0	0	0	0	0	0	0	0
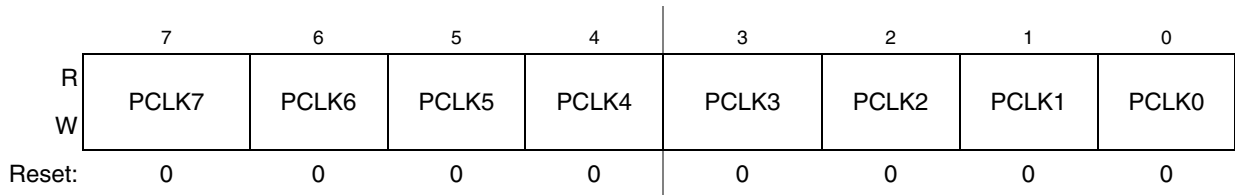
Figure 24-4. PWM Clock Select Register (PWMCLK)

Table 24-4. PWMCLK Field Descriptions

Field	Description
7–0 PCLK*n*	PWM channel *n* clock select. Selects between one of two clock sources for each PWM channel. See Section 24.2.4, "PWM Prescale Clock Select Register (PWMPRCLK)" and Section 24.2.7, "PWM Scale A Register (PWMSCLA)" for more information on how the different clock rates are generated. The even-numbered channels' clock select has no effect when the corresponding PWMCTL[CON*n(n+1)*] bit is set. For example, if PWMCTL[CON01] = 1 then PWMCLK[PCLK0] has no affect.

		PCLK6 & PCLK7 (PWM6 & PCLK7 Clock Source)	PCLK4 & PCLK5 (PWM4 & PWM5 Clock Source)	PCLK2 & PCLK3 (PWM2 & PWM3 Clock Source)	PCLK0 & PCLK1 (PWM0 & PWM1 Clock Source)
	0	B	A	B	A
	1	SB	SA	SB	SA

24.2.4   PWM Prescale Clock Select Register (PWMPRCLK)

The PWMPRCLK register selects the prescale clock source for clocks A and B independently. If the clock prescale is changed while a PWM signal is being generated, a truncated or stretched pulse can occur during the transition.

Address: 0x001B_0003 (PWMPRCLK)   Access: User Read/Write

	7	6	5	4	3	2	1	0
R	0		PCKB		0		PCKA	
W								
Reset:	0	0	0	0	0	0	0	0
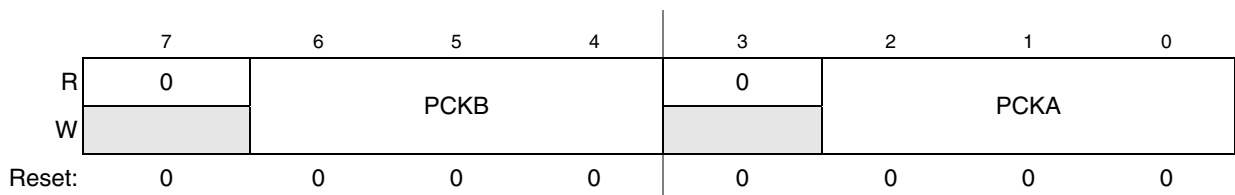
Figure 24-5. PWM Prescale Clock Select Register (PWMPRCLK)

Table 24-5. PWMPRCLK Field Descriptions

Field	Description
7	Reserved, should be cleared.
6–4 PCKB	Clock B prescalar select. These three bits control the rate of Clock B which can be used for PWM channels 2, 3, 6 and 7. <table><tr><th>PCKB</th><th>Clock B Rate</th></tr><tr><td>000</td><td>Internal bus clock $\div 2^0$</td></tr><tr><td>001</td><td>Internal bus clock $\div 2^1$</td></tr><tr><td>...</td><td>...</td></tr><tr><td>111</td><td>Internal bus clock $\div 2^7$</td></tr></table>
3	Reserved, should be cleared.
2–0 PCKA	Clock A prescalar select. These three bits control the rate of Clock A which can be used for PWM channels 0, 1, 4 and 5. <table><tr><th>PCKA</th><th>Clock A Rate</th></tr><tr><td>000</td><td>Internal bus clock $\div 2^0$</td></tr><tr><td>001</td><td>Internal bus clock $\div 2^1$</td></tr><tr><td>...</td><td>...</td></tr><tr><td>111</td><td>Internal bus clock $\div 2^7$</td></tr></table>

24.2.5 PWM Center Align Enable Register (PWMCAE)

The PWMCAE register contains eight control bits for the selection of center-aligned outputs or left-aligned outputs for each PWM channel. Write these bits only when the corresponding channel is disabled. See Section 24.3.2.5, "Left-Aligned Outputs" and Section 24.3.2.6, "Center-Aligned Outputs" for a more detailed description of the PWM output modes.

Address: 0x001B_0004 (PWMCAE)   Access: User Read/Write

	7	6	5	4	3	2	1	0
R W	CAE7	CAE6	CAE5	CAE4	CAE3	CAE2	CAE1	CAE0
Reset:	0	0	0	0	0	0	0	0
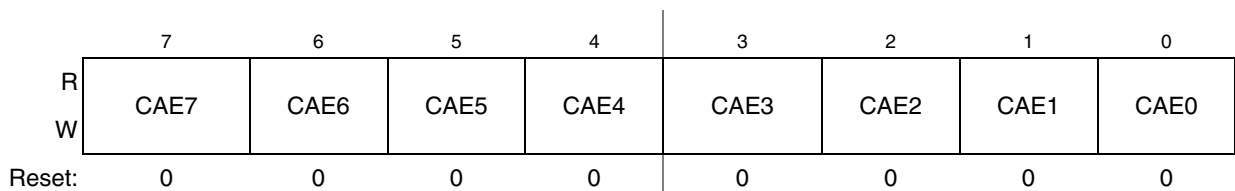
Figure 24-6. PWM Center Align Enable Register (PWMCAE)

Table 24-6. PWMCAE Field Descriptions

Field	Description
7–0 CAE*n*	Center align enable for channel *n*. The even-numbered channels' center align enable has no effect when the corresponding PWMCTL[CON*n(n+1)*] bit is set. For example, if PWMCTL[CON01] = 1 then PWMCAE[CAE0] has no affect. 0 Channel *n* operates in left-aligned output mode 1 Channel *n* operates in center-aligned output mode

24.2.6   PWM Control Register (PWMCTL)

The PWMCTL register provides various control of the PWM module. Change the CON*n(n+1)* bits only when both corresponding channels are disabled. See Section 24.3.2.7, "PWM 16-Bit Functions" for a more detailed description of the concatenation function.

Address: 0x001B_0005 (PWMCTL)   Access: User Read/Write

	7	6	5	4	3	2	1	0
R	CON67	CON45	CON23	CON01	PSWAI	PFRZ	0	0
W								
Reset:	0	0	0	0	0	0	0	0
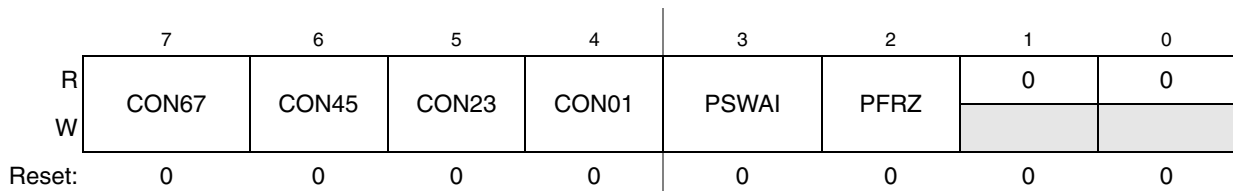
Figure 24-7. PWM Control Register (PWMCTL)

Table 24-7. PWMCTL Field Descriptions

Field	Description
7 CON67	Concatenates PWM channels 6 and 7 to form one 16-bit PWM channel. 0 Channels 6 and 7 are separate 8-bit PWMs 1 Concatenate PWM 6 and 7. Channel 6 becomes the high order byte and channel 6 the low order byte. PWMOUT7 is the output for this 16-bit PWM signal, and PWMOUT6 is disabled. The channel 7 clock select, polarity, center align enable, and enable bits control this concatenated output.
6 CON45	Concatenates PWM channels 4 and 5 to form one 16-bit PWM channel. 0 Channels 4 and 5 are separate 8-bit PWMs 1 Concatenate PWM 4 and 5. Channel 4 becomes the high order byte and channel 5 the low order byte. PWMOUT5 is the output for this 16-bit PWM signal, and PWMOUT4 is disabled. The channel 5 clock select, polarity, center align enable, and enable bits control this concatenated output.
5 CON23	Concatenates PWM channels 2 and 3 to form one 16-bit PWM channel. 0 Channels 2 and 3 are separate 8-bit PWMs 1 Concatenate PWM 2 and 3. Channel 2 becomes the high order byte and channel 3 the low order byte. PWMOUT3 is the output for this 16-bit PWM signal, and PWMOUT2 is disabled. The channel 3 clock select, polarity, center align enable, and enable bits control this concatenated output.
4 CON01	Concatenates PWM channels 0 and 1 to form one 16-bit PWM channel. 0 Channels 0 and 1 are separate 8-bit PWMs 1 Concatenate PWM 0 and 1. Channel 0 becomes the high order byte and channel 1 the low order byte. PWMOUT1 is the output for this 16-bit PWM signal, and PWMOUT0 is disabled. The channel 1 clock select, polarity, center align enable, and enable bits control this concatenated output.
3 PSWAI	PWM stops in doze mode. Disables the input clock to the prescaler while in doze mode. 0 Allow the clock to the prescaler while in doze mode 1 Stop the input clock to the prescaler whenever the core is in doze mode

Table 24-7. PWMCTL Field Descriptions (continued)

Field	Description
2 PFRZ	PWM counters stop in debug mode ($\overline{\text{BKPT}}$ asserted). 0 Allow PWM counters to continue while in debug mode 1 Disable PWM input clock to the prescaler when the core is in debug mode. Useful for emulation as it allows the PWM function to be suspended.
1–0	Reserved, should be cleared.

24.2.7   PWM Scale A Register (PWMSCLA)

PWMSCLA is the programmable scale value used in scaling clock A to generate clock SA. Clock SA is generated according to the following equation:

$$\text{Clock SA} = \frac{\text{Clock A}}{2 \times \text{PWMSCLA}}$$ ***Eqn. 24-1***

Any value written to this register will cause the scale counter to load the new scale value (PWMSCLA).

Address: 0x001B_0008 (PWMSCLA)   Access: User Read/Write

	7	6	5	4	3	2	1	0
R				SCALEA				
W								
Reset:	0	0	0	0	0	0	0	0
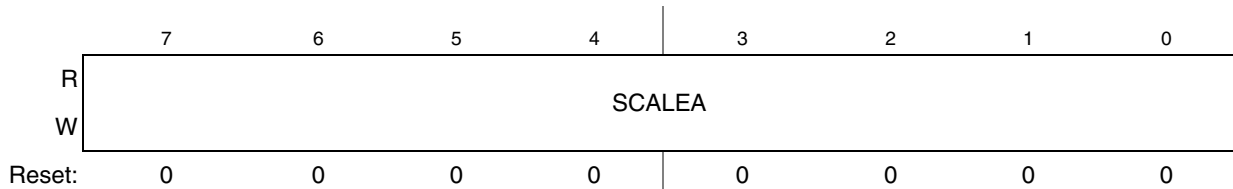
Figure 24-8. PWM Scale A Register (PWMSCLA)

Table 24-8. PWMSCLA Field Descriptions

Field	Description
7–0 SCALEA	Part of divisor used to form Clock SA from Clock A. | SCALEA	Value | |---|---| | 0x00 | 256 | | 0x01 | 1 | | 0x02 | 2 | | ... | ... | | 0xFF | 255 |

24.2.8   PWM Scale B Register (PWMSCLB)

PWMSCLB is the programmable scale value used in scaling clock B to generate clock SB. Clock SB is generated according to the following equation:

$$\text{Clock SB} = \frac{\text{Clock B}}{2 \times \text{PWMSCLB}}$$ ***Eqn. 24-2***

Any value written to this register will cause the scale counter to load the new scale value (PWMSCLB).

Address: 0x001B_0009 (PWMSCLB)   Access: User Read/Write

	7	6	5	4	3	2	1	0
R								
W				SCALEB				
Reset:	0	0	0	0	0	0	0	0
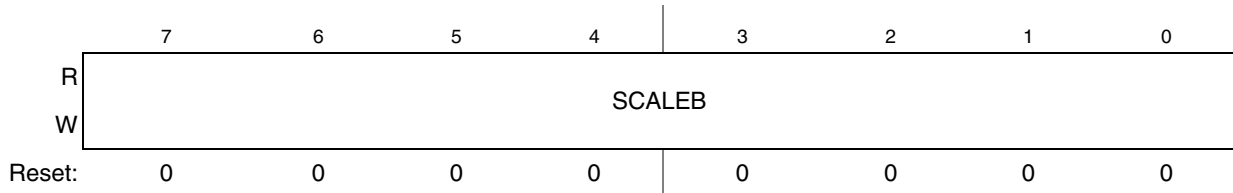
Figure 24-9. PWM Scale B Register (PWMSCLB)

Table 24-9. PWMSCLB Field Descriptions

Field	Description
7–0 SCALEB	Divisor used to form Clock SB from Clock B. <table><tr><th>SCALEB</th><th>Value</th></tr><tr><td>0x00</td><td>256</td></tr><tr><td>0x01</td><td>1</td></tr><tr><td>0x02</td><td>2</td></tr><tr><td>...</td><td>...</td></tr><tr><td>0xFF</td><td>255</td></tr></table>

24.2.9 PWM Channel Counter Registers (PWMCNT*n*)

Each channel has a dedicated 8-bit up/down counter which runs at the rate of the selected clock source, PWMCLK[PCLK*n*]. The user can read the counters at any time without affecting the count or the operation of the PWM channel. Any value written to the counter causes the counter to reset to 0x00, the counter direction to be set to up for center-aligned mode, the immediate load of both duty and period registers with values from the buffers, and the output to change according to the polarity bit.

The counter is also cleared at the end of the effective period (see Section 24.3.2.5, "Left-Aligned Outputs" and Section 24.3.2.6, "Center-Aligned Outputs" for more details). When the channel is disabled (PWME*n*=0), the PWMCNT*n* register does not count. When a channel is enabled (PWME*n*=1), the associated PWM counter starts at the count in the PWMCNT*n* register. For more detailed information on the operation of the counters, refer to Section 24.3.2.4, "PWM Timer Counters."

Address: 0x001B_000C (PWMCNT0)   Access: User Read/Write
　　　　0x001B_000D (PWMCNT1)
　　　　0x001B_000E (PWMCNT2)
　　　　0x001B_000F (PWMCNT3)
　　　　0x001B_0010 (PWMCNT4)
　　　　0x001B_0011 (PWMCNT5)
　　　　0x001B_0012 (PWMCNT6)
　　　　0x001B_0013 (PWMCNT7)

	7	6	5	4	3	2	1	0
R				COUNT				
W								
Reset:	0	0	0	0	0	0	0	0
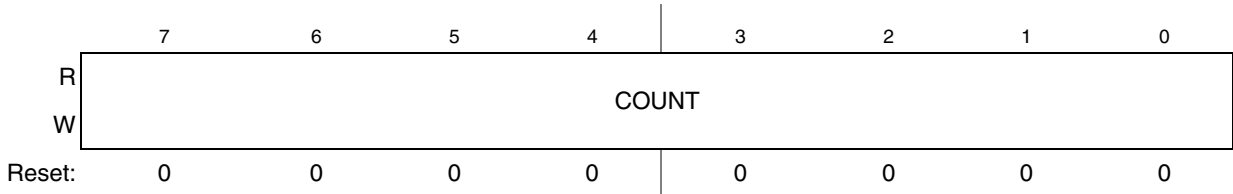
Figure 24-10. PWM Counter Registers (PWMCNT*n*)

Table 24-10. PWMCNT*n* Field Descriptions

Field	Description
7–0 COUNT	Current value of the PWM up counter. Resets to zero when written.

24.2.10　PWM Channel Period Registers (PWMPER*n*)

The PWM period registers determine the period of the associated PWM channel. Refer to Section 24.3.2.3, "PWM Period and Duty" for more information.

Calculating the output period depends on the output mode (center-aligned has twice the period as left-aligned mode) as well as PWMPER*n*. See the below equation:

$$\text{PWM}n \text{ period} = \text{Channel clock period} \times (\text{PWMCAE[CAE}n] + 1) \times \text{PWMPER}n \qquad \textbf{\textit{Eqn. 24-3}}$$

For boundary case programming values (e.g. PWMPER*n* = 0x00), please refer to Section 24.3.2.8, "PWM Boundary Cases."

Address: 0x001B_0014 (PWMPER0)   Access: User Read/Write
　　　　0x001B_0015 (PWMPER1)
　　　　0x001B_0016 (PWMPER2)
　　　　0x001B_0017 (PWMPER3)
　　　　0x001B_0018 (PWMPER4)
　　　　0x001B_0019 (PWMPER5)
　　　　0x001B_001A (PWMPER6)
　　　　0x001B_001B (PWMPER7)

	7	6	5	4	3	2	1	0
R				PERIOD				
W								
Reset:	1	1	1	1	1	1	1	1
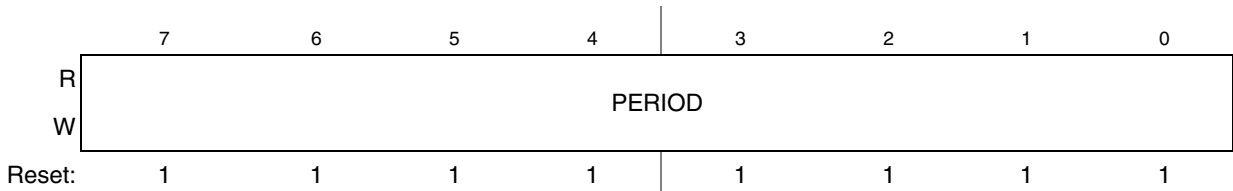
Figure 24-11. PWM Period Registers (PWMPER*n*)

Table 24-11. PWMPER*n* Field Descriptions

Field	Description
7–0 PERIOD	Period counter for the output PWM signal. If PERIOD = 0x00, the PWM*n* output is always high (PPOL*n*=1) or always low (PPOL*n*=0). See Section 24.3.2.8, "PWM Boundary Cases" for other special cases.

24.2.11 PWM Channel Duty Registers (PWMDTY*n*)

The PWM duty registers determine the duty cycle of the associated PWM channel. To calculate the output duty cycle (high time as a percentage of period) for a particular channel:

$$\text{Duty Cycle} = \left| \left(1 - \text{PWMPOL}[\text{PPOL}n] - \frac{\text{PWMDTY}n}{\text{PWMPER}n} \right) \right| \times 100\% \qquad \textit{Eqn. 24-4}$$

For boundary case programming values (e.g. PWMDTY*n* = 0x00 or PWMDTY*n* > PWMPER*n*), please refer to Section Section 24.3.2.8, "PWM Boundary Cases."

Address: 0x001B_001C (PWMDTY0)   Access: User Read/Write
         0x001B_001D (PWMDTY1)
         0x001B_001E (PWMDTY2)
         0x001B_001F (PWMDTY3)
         0x001B_0020 (PWMDTY4)
         0x001B_0021 (PWMDTY5)
         0x001B_0022 (PWMDTY6)
         0x001B_0023 (PWMDTY7)

	7	6	5	4	3	2	1	0
R				DUTY				
W								
Reset:	1	1	1	1	1	1	1	1
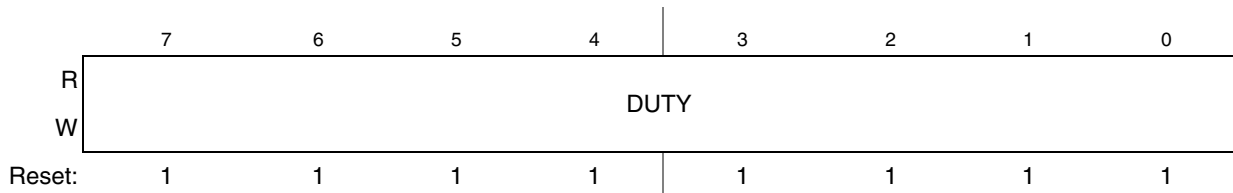
Figure 24-12. PWM Duty Registers (PWMDTY*n*)

Table 24-12. PWMDTY*n* Field Descriptions

Field	Description
7–0 DUTY	Contains the duty value used to determine when a transition will occur on the PWM output signal. When a match occurs with the corresponding PWMCNT*n* register, the PWM output will toggle. If DUTY = 0x00, the PWM*n* output is always low (PPOL*n*=1) or always high (PPOL*n*=0). See Section 24.3.2.8, "PWM Boundary Cases" for other special cases.

24.2.12 PWM Shutdown Register (PWMSDN)

The PWM shutdown register provides emergency shutdown functionality of the PWM module. The PWMSDN[7:1] bits are ignored if PWMSDN[SDNEN] is cleared.

Address: 0x24 (PWMSDN)   Access: Read/Write

	7	6	5	4	3	2	1	0
R	IF	IE	0	LVL	0	PWM7IN	PWM7IL	SDNEN
W	w1c		RESTART					
Reset:	0	0	0	0	0	0	0	0
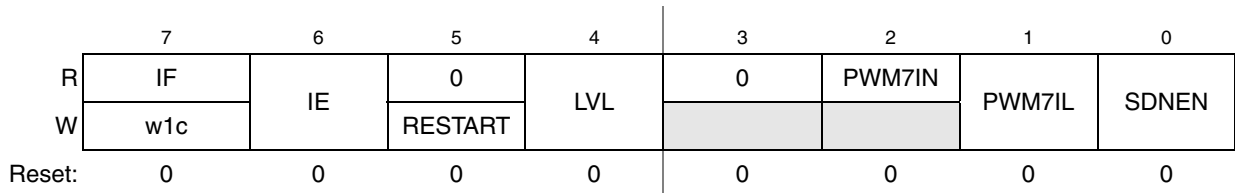
Figure 24-13. PWM Shutdown Register (PWMSDN)

Table 24-13. PWMSDN Field Descriptions

Field	Description
7 IF	PWM interrupt flag. Any change in state of PWM7IN will be flagged by setting this bit. The flag is cleared by writing a '1' to it. Writing '0' has no effect. 0 No change in PWM7IN input 1 Change in PWM7IN input
6 IE	PWM interrupt enable. An interrupt will be triggered to the device's interrupt controller when PWMSDN[IF] is set. 0 Interrupt is disabled 1 Interrupt is enabled
5 RESTART	PWM restart. After setting the RESTART bit, the PWM channels start running after the corresponding counter resets to zero. Also, if emergency shutdown is cleared (after being set), the PWM outputs will restart after the corresponding counter resets to zero. This bit is self-clearing, so is always read as zero.
4 LVL	PWM shutdown output level. Describes the behaivor of the PWM outputs when PWM7IN input is asserted and PWMSDN[SDNEN] is set. 0 PWM outputs are forced to logic 0 1 PWM outputs are forced to logic 1
3	Reserved, should be cleared.
2 PWM7IN	PWM channel 7 input status. Reflects the current status of the PWMOUT7 pin. Read only.
1 PWM7IL	PWM channel 7 input polariy. If PWMSDN[SDNEN] is set, this bit sets the active level of the PWM 7 channel 0 PWM 7 input is active low 1 PWN 7 input is active high
0 SDNEN	PWM emergency shutdown enable. If set, the pin associated with PWM channel 7 is forced to input and the emergency shutdown feature is enabled. 0 Emergency shutdown is disabled 1 Emergency shutdown is enabled

24.3 Functional Description

24.3.1 PWM Clock Select

There are four available clocks named Clock A, B, SA (Scaled A), and SB (Scaled B), all of which are based on the internal bus clock.

Clock A and B can be programmed to run at 1, 1/2, ..., 1/128 times the internal bus clock. Clock SA and SB use clock A and B respectively as an input and divides it further with a reloadable counter. The rates available for clock SA and SB are programmable to run at clock A and B divided by 2, 4, ..., or 512. Each

PWM channel has the capability of selecting one of two clocks, either the prescaled clock (clock A or B) or the scaled clock (clock SA or SB). The block diagram in Figure 24-14 shows the four different clocks and how the scaled clocks are created.
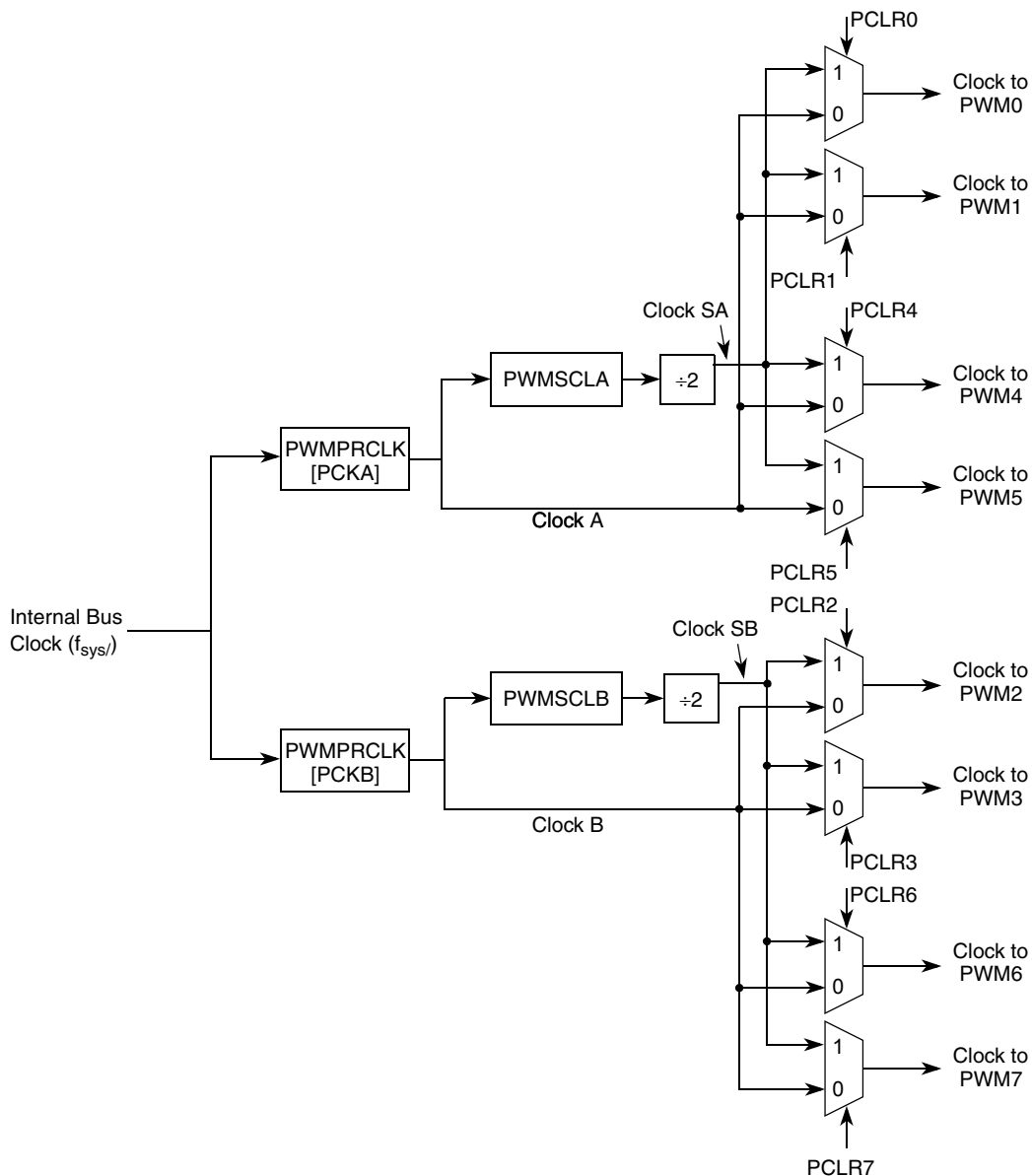
Figure 24-14. PWM Clock Select Block Diagram

24.3.1.1 Prescaled Clock (A or B)

The internal bus clock is the input clock to the PWM prescaler which can be disabled when the device is in debug mode by setting the PWMCTL[PFRZ] bit. This is useful for reducing power consumption and for emulation in order to freeze the PWM. The input clock is also disabled when all PWM channels are disabled (PWME*n*=0).

Clock A and B are scaled values of the input clock. The value is software selectable for both clock A and B and has options of 1, 1/2, ..., or 1/128 times the internal bus clock. The value selected for clock A and B are determined by the PWMPRCLK[PCKAn] and PWMPRCLK[PCKBn] bits.

24.3.1.2   Scaled Clock (SA or SB)

The scaled A (SA) and scaled B (SB) clocks use clock A and B respectively as inputs and divide it further with a user programmable value and then divide this by 2. The rates available for clock SA are programmable to run at clock A divided by 2, 4, ..., or 512. Similar rates are available for clock SB.

Clock SA equals Clock A divided by two times the value in the PWMSCLA register:

$$\text{Clock SA} = \frac{\text{Clock A}}{2 \times \text{PWMSCLA}}$$

Eqn. 24-5

Similarly, Clock SB is generated according to the following equation:

$$\text{Clock SB} = \frac{\text{Clock B}}{2 \times \text{PWMSCLB}}$$

Eqn. 24-6

As an example, consider the case in which the user writes 0xFF into the PWMSCLA register. Clock A for this case is selected to be internal bus clock divided by 4. A pulse will occur at a rate of once every 255×4 bus cycles. Passing this through the divide by two circuit produces a clock signal of the internal bus clock divided by 2040. Similarly, a value of 0x01 in the PWMSCLA register when Clock A is internal bus clock divided by 4 will produce an internal bus clock divided by 8 rate.

Writing to PWMSCLA or PWMSCLB causes the associated 8-bit down counter to be re-loaded. Otherwise, when changing rates the counter would have to count down to 0x01 before counting at the proper rate. Forcing the associated counter to re-load the scale register value every time PWMSCLA or PWMSCLB is written prevents this.

Writing to the scale registers while channels are operating can cause irregularities in the PWM outputs.

24.3.1.3   Clock Select

Each PWM channel has the capability of selecting one of two clocks. For channels 0, 1, 4, and 5 the clock choices are clock A or SA. For channels 2, 3, 6 and 7 the choices are clock B or SB. The clock selection is done with the PWMCLK[PCLKx] control bits.

Changing clock control bits while channels are operating can cause irregularities in the PWM outputs.

24.3.2   PWM Channel Timers

The main part of the PWM module is the actual timers. Each of the timer channels has a counter, a period register and a duty register (each are 8-bit). The waveform output period is controlled by a match between the period register and the value in the counter. The duty is controlled by a match between the duty register and the counter value and causes the state of the output to change during the period. The starting polarity of the output is also selectable on a per channel basis. Figure 24-15 shows a block diagram for a PWM timer.
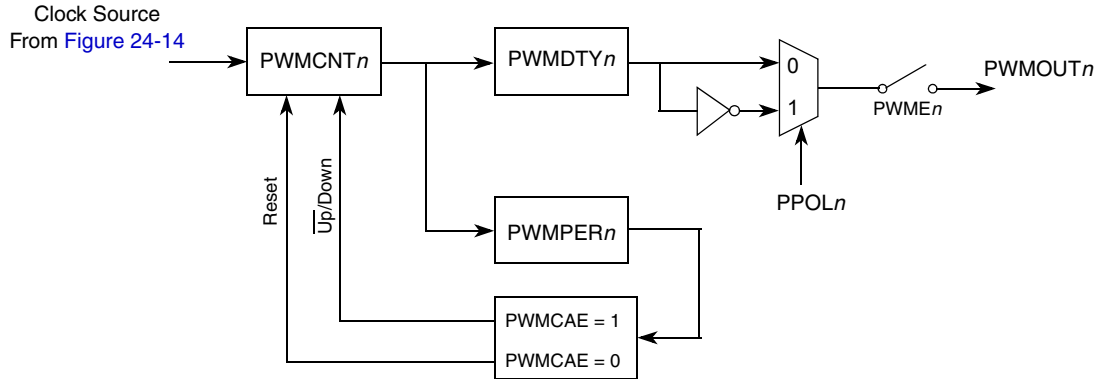
Figure 24-15. PWM Timer Channel Block Diagram

24.3.2.1　PWM Enable

Each PWM channel has an enable bit (PWME*n*) to start its waveform output. When any of the PWME*n* bits are set (PWME*n*=1), the associated PWM output signal is enabled immediately. However, the actual PWM waveform is not available on the associated PWM output until its clock source begins its next cycle due to the synchronization of PWME*n* and the clock source. An exception to this is when channels are concatenated. Refer to Section 24.3.2.7, "PWM 16-Bit Functions" for more detail.

Note that the first PWM cycle after enabling the channel can be irregular. When the channel is disabled (PWME*n*=0), the counter for the channel does not count.

24.3.2.2　PWM Polarity

Each channel has a polarity bit to allow starting a waveform cycle with a high or low signal. This is shown on the block diagram as a mux select. When one of the bits in the PWMPOL register is set, the associated PWM channel output is high at the beginning of the waveform, then goes low when the duty count is reached. Conversely, if the polarity bit is zero, the output starts low and then goes high when the duty count is reached.

24.3.2.3　PWM Period and Duty

Dedicated period and duty registers exist for each channel and are double buffered so that if they change while the channel is enabled, the change will not take effect until one of the following occurs:

- The effective period ends
- The PWMCNT*n* register is written (counter resets to 0x00)
- The channel is disabled, PWME*n* = 0

In this way, the output of the PWM will always be either the old waveform or the new waveform, not some variation in between. If the channel is not enabled, then writes to the period and duty registers will go directly to the latches as well as the buffer.

A change in duty or period can be forced into effect immediately by writing the new value to the duty and/or period registers and then writing to the counter. This forces the counter to reset and the new duty

and/or period values to be latched. In addition, since the counter is readable it is possible to know where the count is with respect to the duty value and software can be used to make adjustments. When forcing a new period or duty into effect immediately, an irregular PWM cycle can occur.

Depending on the polarity bit, the duty registers will contain the count of either the high time or the low time.

24.3.2.4   PWM Timer Counters

Each channel has a dedicated 8-bit up/down counter which runs at the rate of the selected clock source (see Figure 24-14 for the available clock sources and rates). The counter compares to two registers, a duty register and a period register as shown in Figure 24-15. When the PWM counter matches the duty register the output flip-flop changes state causing the PWM waveform to also change state. A match between the PWM counter and the period register behaves differently depending on what output mode is selected as shown in Figure 24-15 and described in Section 24.3.2.5, "Left-Aligned Outputs" and Section 24.3.2.6, "Center-Aligned Outputs."

Each channel counter can be read at anytime without affecting the count or the operation of the PWM channel.

Any value written to the counter causes the counter to reset to 0x00, the counter direction to be set to up, the immediate load of both duty and period registers with values from the buffers, and the output to change according to the polarity bit. When the channel is disabled (PWMEn = 0), the counter stops. When a channel becomes enabled (PWMEn = 1), the associated PWM counter continues from the count in the PWMCNTn register. This allows the waveform to continue where it left off when the channel is re-enabled. When the channel is disabled, writing "0" to the period register will cause the counter to reset on the next selected clock.

NOTE

If the user wants to start a new "clean" PWM waveform without any "history" from the old waveform, the user must write to channel counter (PWMCNTn) prior to enabling the PWM channel (PWMEn = 1).

Generally, writes to the counter are done prior to enabling a channel in order to start from a known state. However, writing a counter can also be done while the PWM channel is enabled (counting). The effect is similar to writing the counter when the channel is disabled except that the new period is started immediately with the output set according to the polarity bit. Writing to the counter while the channel is enabled can cause an irregular PWM cycle to occur.

The counter is cleared at the end of the effective period (see Section 24.3.2.5, "Left-Aligned Outputs" and Section 24.3.2.6, "Center-Aligned Outputs" for more details).

Table 24-14. PWM Timer Counter Conditions

Counter Clears (0x00)	Counter Counts	Counter Stops
When PWMCNTn register written to any value	When PWM channel is enabled (PWMEn = 1). Counts from last value in PWMCNTn.	When PWM channel is disabled (PWMEn = 0)
Effective period ends		

24.3.2.5   Left-Aligned Outputs

The PWM timer provides the choice of two types of outputs, left- or center-aligned outputs. They are selected with the PWMCAE[CAE*n*] bits. If the CAE*n* bit is cleared, the corresponding PWM output will be left-aligned.

In left-aligned output mode, the 8-bit counter is configured as an up counter only. It compares to two registers, a duty register and a period register as shown in the block diagram in Figure 24-15. When the PWM counter matches the duty register the output flip-flop changes state causing the PWM waveform to also change state. A match between the PWM counter and the period register resets the counter and the output flip-flop as shown in Figure 24-15 as well as performing a load from the double buffer period and duty register to the associated registers as described in Figure 24.3.2.3. The counter counts from 0 to the value in the period register minus 1.

NOTE

Changing the PWM output mode from left-aligned output to center-aligned output (or vice versa) while channels are operating can cause irregularities in the PWM output. It is recommended to program the output mode before enabling the PWM channel.
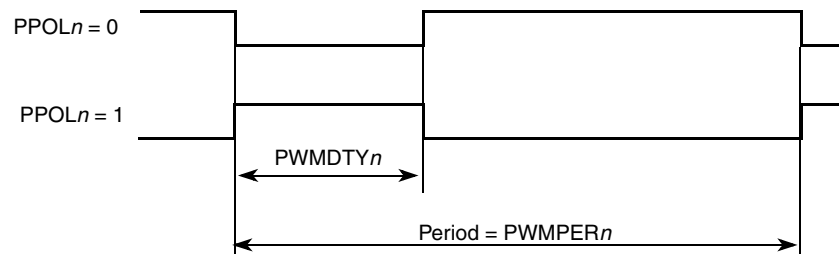
Figure 24-16. PWM Left-Aligned Output Waveform

To calculate the output frequency in left-aligned output mode for a particular channel, take the selected clock source frequency for the channel (A, B, SA, or SB) and divide it by the value in the period register for that channel.

$$\text{PWM}n \text{ frequency} = \frac{\text{Clock (A, B, SA, or SB)}}{\text{PWMPER}n} \qquad \textbf{\textit{Eqn. 24-7}}$$

The PWM*n* duty cycle (high time as a percentage of period) is expressed as:

$$\text{Duty Cycle} = \left(1 - \text{PWMPOL[PPOL}n] - \frac{\text{PWMDTY}n}{\text{PWMPER}n}\right) \times 100\% \qquad \textbf{\textit{Eqn. 24-8}}$$

24.3.2.5.1   Left-Aligned Output Example

As an example of a left-aligned output, consider the following case:

Clock Source = internal bus clock, where internal bus clock = MHz (ns period)

PPOL*n* = 0, PWMPER*n* = 4, PWMDTY*n* = 1

PWM*n* Frequency = MHz ÷ 4 = MHz

PWM*n* Period = ns

$$PWMn \text{ Duty Cycle} = \left(1 - \frac{1}{4}\right) \times 100\% = 75\%$$

Shown below is the output waveform generated:
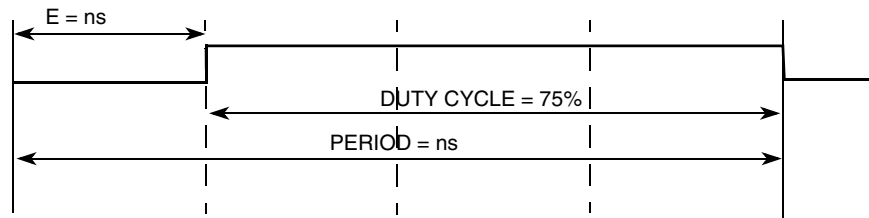
Figure 24-17. PWM Left-Aligned Output Example Waveform

24.3.2.6   Center-Aligned Outputs

For center-aligned output mode selection, set the PWMCAE[CAE*n*] bit and the corresponding PWM output will be center-aligned.

The 8-bit counter operates as an up/down counter in this mode and is set to up whenever the counter is equal to 0x00. The counter compares to two registers, a duty register and a period register as shown in the block diagram in Figure 24-15. When the PWM counter matches the duty register the output flip-flop changes state causing the PWM waveform to also change state. A match between the PWM counter and the period register changes the counter direction from an up-count to a down-count. When the PWM counter decrements and matches the duty register again, the output flip-flop changes state causing the PWM output to also change state. When the PWM counter decrements and reaches zero, the counter direction changes from a down-count back to an up-count and a load from the double buffer period and duty registers to the associated registers is performed as described in Figure 24.3.2.3. The counter counts from 0 up to the value in the period register and then back down to 0. Thus the effective period is PWMPER*n* × 2.

Changing the PWM output mode from left-aligned output to center-aligned output (or vice versa) while channels are operating can cause irregularities in the PWM output. It is recommended to program the output mode before enabling the PWM channel.
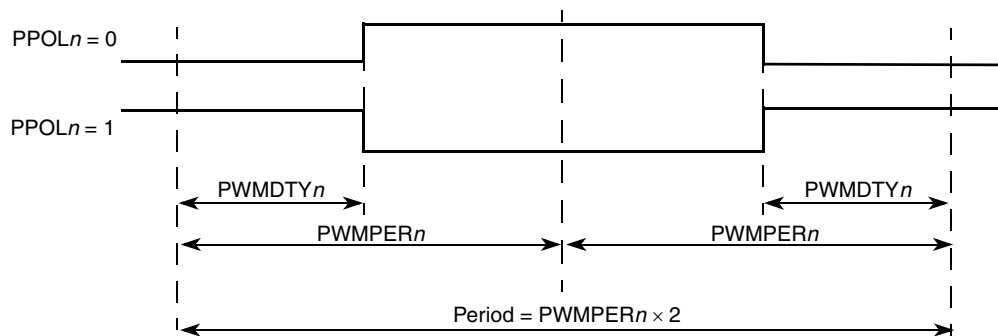
Figure 24-18. PWM Center-Aligned Output Waveform

To calculate the output frequency in center-aligned output mode for a particular channel, take the selected clock source frequency for the channel (A, B, SA, or SB) and divide it by twice the value in the period register for that channel.

$$\text{PWM}n \text{ frequency} = \frac{\text{Clock (A, B, SA, or SB)}}{2 \times \text{PWMPER}n}$$

Eqn. 24-9

The PWM*n* duty cycle (high time as a percentage of period) is expressed as:

$$\text{Duty Cycle} = \left(1 - \text{PWMPOL[PPOL}n] - \frac{\text{PWMDTY}n}{\text{PWMPER}n}\right) \times 100\%$$

Eqn. 24-10

24.3.2.6.1 Center-Aligned Output Example

As an example of a center-aligned output, consider the following case:

Clock Source = internal bus clock, where internal bus clock= MHz (ns period)

PPOL*n* = 0, PWMPER*n* = 4, PWMDTY*n* = 1

PWM*n* Frequency = MHz / (2×4) = MHz

PWM*n* Period = ns

PWMn Duty Cycle $= \left(1 - \frac{1}{4}\right) \times 100\% = 75\%$

Shown below is the output waveform generated.
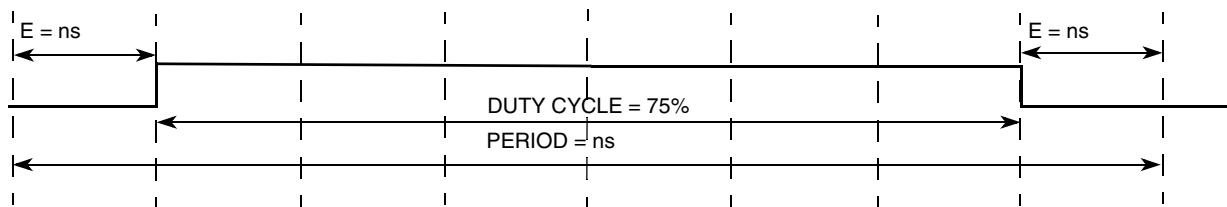
Figure 24-19. PWM Center-Aligned Output Example Waveform

24.3.2.7 PWM 16-Bit Functions

The PWM timer also has the option of generating 8-channels of 8-bits or 4-channels of 16-bits for greater PWM resolution. This 16-bit channel option is achieved through the concatenation of two 8-bit channels.

The PWMCTL register contains four concatenation control bits, each of which is used to concatenate a pair of PWM channels into one 16-bit channel. Channels 0 and 1 are concatenated with the CON01 bit, channels 2 and 3 are concatenated with the CON23 bit, and so on. Change these bits only when both corresponding channels are disabled.

As shown in Figure 24-20, when channels 2 and 3 are concatenated, channel 2 registers become the high order bytes of the double byte channel. When channels 0 and 1 are concatenated, channel 0 registers become the high order bytes of the double byte channel.

When using the 16-bit concatenated mode, the clock source is determined by the low order 8-bit channel clock select control bits (the odd numbered channel). The resulting PWM is output to the pins of the corresponding low order 8-bit channel as also shown in Figure 24-20. The polarity of the resulting PWM output is controlled by the PPOL*n* bit of the corresponding low order 8-bit channel as well.

Once concatenated mode is enabled (PWMCTL[CON*nn*] bits set) then enabling/disabling the corresponding 16-bit PWM channel is controlled by the low order PWME*n* bit. In this case, the high order bytes' PWME*n* bits have no effect and their corresponding PWM output is disabled.

In concatenated mode, writes to the 16-bit counter by using a 16-bit access or writes to either the low or high order byte of the counter will reset the 16-bit counter. Reads of the 16-bit counter must be made by 16-bit access to maintain data coherency.
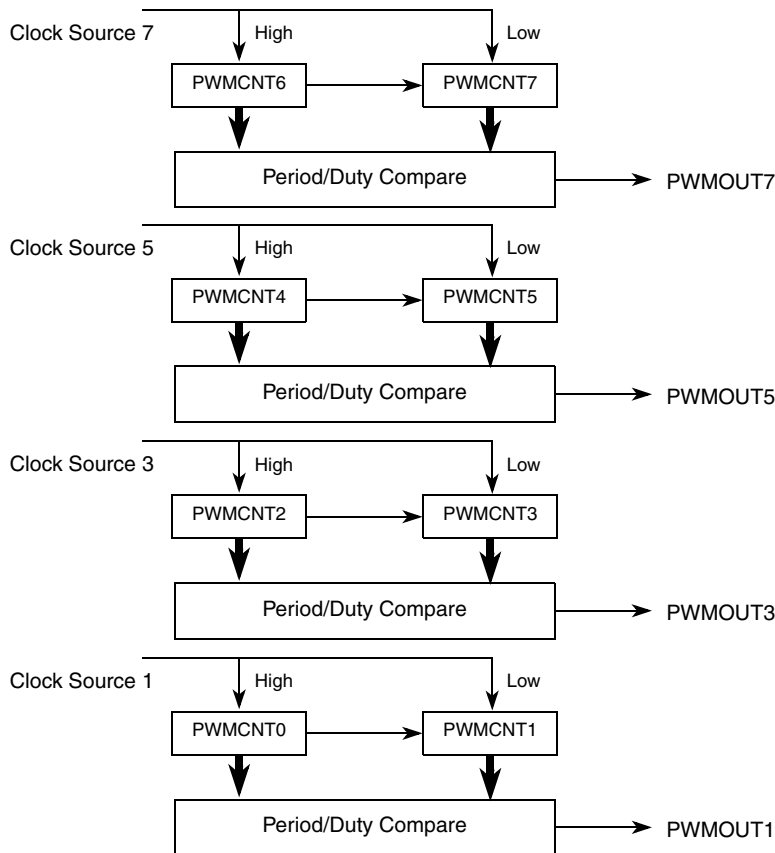
Figure 24-20. PWM 16-Bit Mode

Either left- or center-aligned output mode can be used in concatenated mode and is controlled by the low order CAE*n* bit. The high order CAE*n* bit has no effect. The table shown below is used to summarize which channels are used to set the various control bits when in 16-bit mode.

Table 24-15. 16-bit Concatenation Mode Summary

CON*nn*	PWME*n*	PPOL*n*	PCLK*n*	CAE*n*	PWM*n* Output
CON67	PWM7	PPOL7	PCLK7	CAE7	PWMOUT7
CON45	PWM5	PPOL5	PCLK5	CAE5	PWMOUT5
CON23	PWME3	PPOL3	PCLK3	CAE3	PWMOUT3
CON01	PWME1	PPOL1	PCLK1	CAE1	PWMOUT1

24.3.2.8   PWM Boundary Cases

The following table summarizes the boundary conditions for the PWM regardless of the output mode (left-or center-aligned) and 8-bit (normal) or 16-bit (concatenation):

Table 24-16. PWM Boundary Cases

PWMDTY*n*	PWMPER*n*	PPOL*n*	PWM*n* Output
0x00 (indicates no duty)	>0x00	1	Always Low
0x00 (indicates no duty)	>0x00	0	Always High
XX	0x00[1] (indicates no period)	1	Always High
XX	0x00[1] (indicates no period)	0	Always Low
≥ PWMPER*n*	XX	1	Always High
≥ PWMPER*n*	XX	0	Always Low

[1]   Counter = 0x00 and does not count.

Chapter 25
FlexCAN

25.1 Introduction

The FlexCAN is a communication controller implementing the controller area network (CAN) protocol, an asynchronous communications protocol used in automotive and industrial control systems. It is a high speed (1 Mbps), short distance, priority based protocol that can communicate using a variety of mediums (for example, fiber optic cable or an unshielded twisted pair of wires). The FlexCAN supports both the standard and extended identifier (ID) message formats specified in the CAN protocol specification, revision 2.0, part B.

The CAN protocol was primarily, but not only, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness, and required bandwidth. A general working knowledge of the CAN protocol revision 2.0 is assumed in this document. For details, refer to the CAN protocol revision 2.0 specification.

25.1.1 Block Diagram

A block diagram describing the various submodules of the FlexCAN module is shown in Figure 25-1. Each submodule is described in detail in subsequent sections. The message buffer architecture is shown in Figure 25-2.
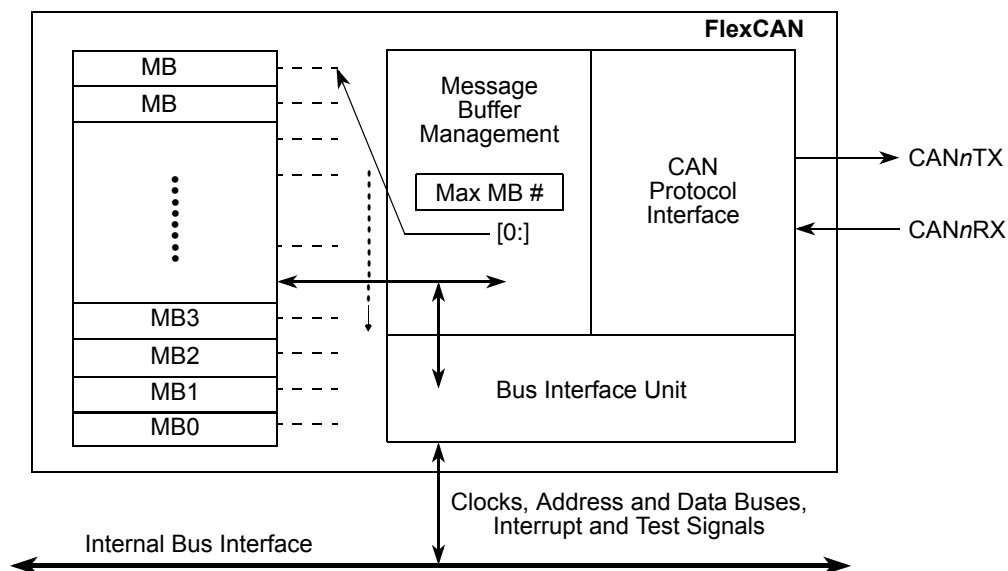
Figure 25-1. FlexCAN Block Diagram and Pinout

MCF5213 Reference Manual, Rev. 1.1
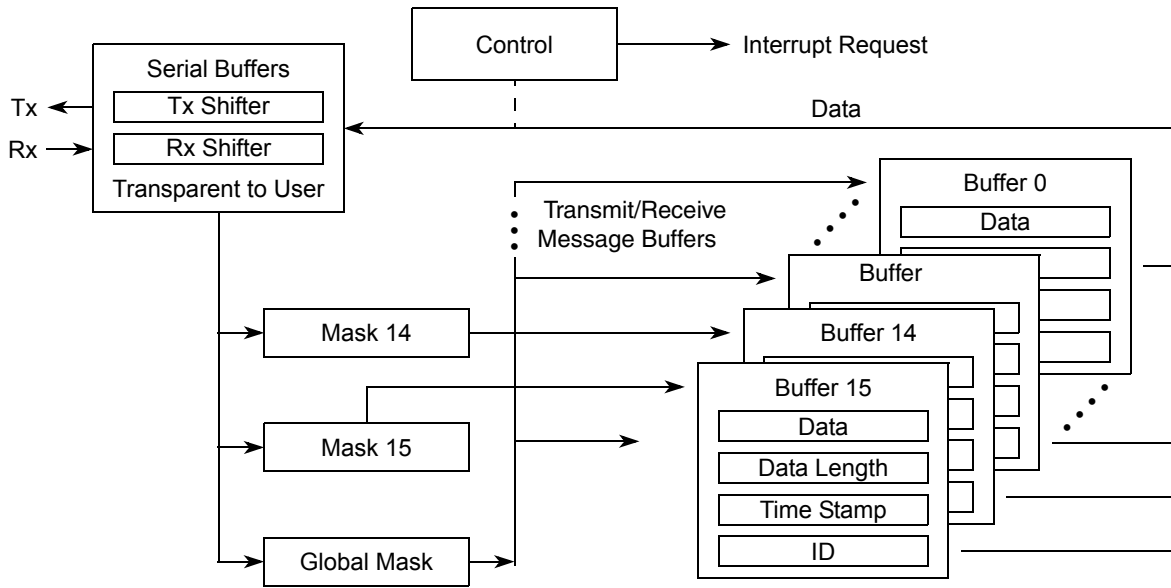
Figure 25-2. FlexCAN Message Buffer Architecture

25.1.1.1 The CAN System

A typical CAN system is shown below in Figure 25-3.
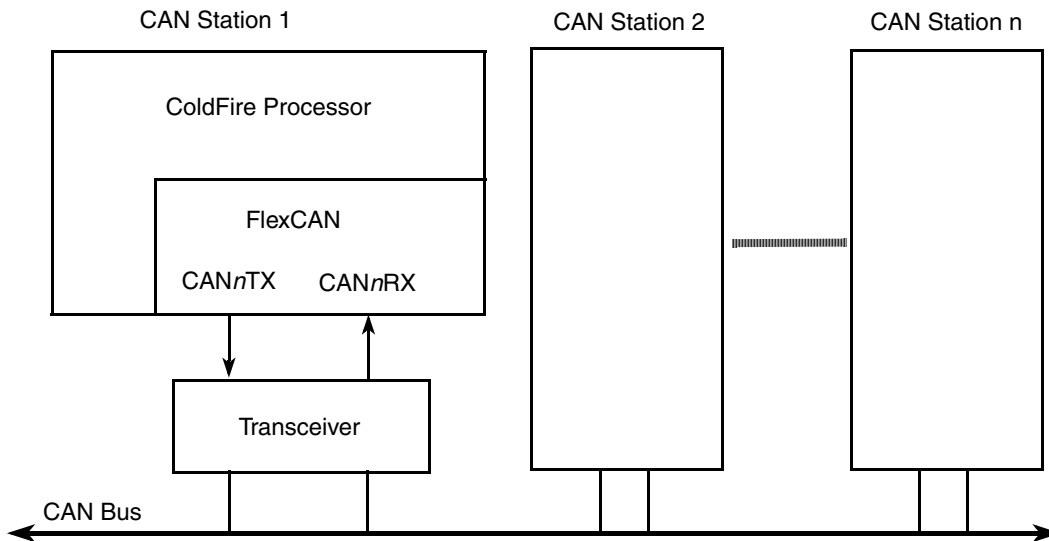
Figure 25-3. Typical CAN System

Each CAN station is connected physically to the CAN bus through a transceiver. The transceiver provides the transmit drive, waveshaping, and receive/compare functions required for communicating on the CAN bus. It can also provide protection against damage to the FlexCAN caused by a defective CAN bus or defective stations.

25.1.2 Features

Following are the main features of the FlexCAN module:

- Full implementation of the CAN protocol specification version 2.0B
  - — Standard data and remote frames (up to 109 bits long)
  - — Extended data and remote frames (up to 127 bits long)
  - — 0–8 bytes data length
  - — Programmable bit rate up to 1 Mbps
  - — Content-related addressing
- Up to flexible message buffers of zero to eight bytes data length, each configurable as Rx or Tx, all supporting standard and extended messages
- Listen-only mode capability
- Three programmable mask registers: global (for MBs 0–13), special for MB14, and special for MB15
- Programmable transmission priority scheme: lowest ID or lowest buffer number
- Time Stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Programmable I/O modes
- Maskable interrupts
- Independent of the transmission medium (an external transceiver is assumed)
- Open network architecture
- Multimaster bus
- High immunity to EMI
- Short latency time due to an arbitration scheme for high-priority messages

25.1.3 Modes of Operation

25.1.3.1 Normal Mode

In normal mode, the module operates receiving and/or transmitting message frames, errors are handled normally, and all the CAN protocol functions are enabled. User and supervisor modes differ in the access to some restricted control registers.

25.1.3.2 Freeze Mode

Freeze mode is entered by:

- Setting CANMCR*n*[FRZ], and
- Setting CANMCR*n*[HALT], or by asserting the $\overline{\text{BKPT}}$ signal.

Once entry into freeze mode is requested, the FlexCAN waits until an intermission or idle condition exists on the CAN bus, or until the FlexCAN enters the error passive or bus off state. Once one of these

conditions exists, the FlexCAN waits for the completion of all internal activity such as arbitration, matching, move-in, and move-out. When this happens, the following events occur:

- The FlexCAN stops transmitting/receiving frames.
- The prescaler is disabled, thus halting all CAN bus communication.
- The FlexCAN ignores its Rx pins and drives its Tx pins as recessive.
- The FlexCAN loses synchronization with the CAN bus and the NOTRDY and FRZACK bits in CANMCR*n* are set.
- The CPU is allowed to read and write the error counter registers (in other modes they are read-only).

After engaging one of the mechanisms to place the FlexCAN in freeze mode, the user must wait for the FRZACK bit to be set before accessing any other registers in the FlexCAN, otherwise unpredictable operation may occur. In freeze mode, all memory mapped registers are accessible.

To exit freeze mode, the $\overline{\text{BKPT}}$ line must be negated or the HALT bit in CANMCR*n* must be cleared. Once freeze mode is exited, the FlexCAN will resynchronize with the CAN bus by waiting for 11 consecutive recessive bits before beginning to participate in CAN bus communication.

25.1.3.3   Module Disabled Mode

This mode disables the FlexCAN module; it is entered by setting CANMCR*n*[MDIS]. If the module is disabled during freeze mode, it shuts down the system clocks, sets the LPMACK bit, and clears the FRZACK bit.

If the module is disabled during transmission or reception, FlexCAN does the following:

- Waits to be in either idle or bus-off state, or else waits for the third bit of intermission and then checks it to be recessive
- Waits for all internal activities such as arbitration, matching, move-in, and move-out to finish
- Ignores its Rx input pin and drives its Tx pin as recessive
- Shuts down the system clocks

The bus interface unit continues to operate, enabling the CPU to access memory mapped registers, except the free-running timer, the error counter register and the message buffers, which cannot be accessed when the module is disabled. Exiting from this mode is done by negating the MDIS bit, which will resume the clocks and negate the LPMACK bit.

25.1.3.4   Loop-Back Mode

The module enters this mode when the LPB bit in the control register is set. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

25.1.3.5 Listen-Only Mode

In listen-only mode, transmission is disabled, all error counters are frozen and the module operates in a CAN error passive mode. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message. Because the module does not influence the CAN bus in this mode, the device is capable of functioning like a monitor or for automatic bit-rate detection.

25.2 External Signal Description

Each FlexCAN module has two I/O signals connected to the external MPU pins: CAN0TX, CAN0RX, CAN1TX, and CAN1RX. CAN*n*TX transmits serial data to the CAN bus transceiver, while CAN*n*RX receives serial data from the CAN bus transceiver.

25.3 Memory Map/Register Definition

The FlexCAN module address space is split into 128 bytes starting at the base address, and then an extra bytes starting at the base address +128. The upper are fully used for the message buffer structures, as described in Section 25.3.9, "Message Buffer Structure." Out of the lower 128 bytes, only part is occupied by various registers.

Table 25-1. FlexCAN Memory Map

Offset FlexCAN	Register	Affected by Hard Reset	Affected by Soft Reset	Access	Reset Value	Section/Page
Supervisor-only Access Registers						
0x1C_0000	FlexCAN Module Configuration Register (CANMCR)	Y	Y	R/W	0xD890_000F	25.3.1/25-6
Supervisor/User Access Registers						
0x1C_0004	FlexCAN Control Register (CANCTRL)	Y	N	R/W	0x0000_0000	25.3.2/25-8
0x1C_0008	Free Running Timer (TIMER)	Y	Y	R/W	0x0000_0000	25.3.3/25-10
0xC_0010	Rx Global Mask (RXGMASK)	Y	N	R/W	0x1FFF_FFFF	25.3.4/25-11
0x1C_0014	Rx Buffer 14 Mask (RX14MASk)	Y	N	R/W	0x1FFF_FFFF	25.3.4/25-11
0x1C_0018	Rx Buffer 15 Mask (RX15MASK)	Y	N	R/W	0x1FFF_FFFF	25.3.4/25-11
0x1C_001C	Error Counter Register (ERRCNT)	Y	Y	R/W	0x0000_0000	25.3.6/25-14
0x1C_0020	Error and Status Register (ERRSTAT)	Y	Y	R/W	0x0000_0000	25.3.6/25-14

Table 25-1. FlexCAN Memory Map (continued)

Offset FlexCAN	Register	Affected by Hard Reset	Affected by Soft Reset	Access	Reset Value	Section/Page
0x1C_0028	Interrupt Mask Register (IMASK)	Y	Y	R/W	0x0000_0000	25.3.7/25-15
0x1C_0030	Interrupt Flag Register (IFLAG)	Y	Y	R/W	0x0000_0000	25.3.8/25-16
0x1C_0080	Message Buffers 0–15 (MB0–15)	N	N	R/W		25.3.9/25-16

NOTE

The FlexCAN has no hard-wired protection against invalid bit/field programming within its registers. Specifically, no protection is provided if the programming does not meet CAN protocol requirements.

Programming the FlexCAN control registers is typically done during system initialization, prior to the FlexCAN becoming synchronized with the CAN bus. The configuration registers can be changed after synchronization by halting the FlexCAN module. This is done when the user sets the CANMCRn[HALT] bit. The FlexCAN responds by setting the CANMCRn[NOTRDY] bit.

25.3.1 FlexCAN Configuration Register (CANMCRn)

CANMCRn defines global system configurations, such as the module operation mode and maximum message buffer configuration. Most of the fields in this register can be accessed at any time, except the MAXMB field, which should only be changed while the module is in freeze mode.

IPSBAR 0x1C_0000 (CANMCR0)
Offset: Access: Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MDIS	FRZ	0	HALT	NOT RDY	0	SOFT RST	FRZ ACK	SUPV	0	0	LPM ACK	0	0	0	0
W																
Reset	1	1	0	1	1	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
Reset	0	0	0	0	0	0	0	0	0	0	0		1	1	1	1

Figure 25-4. FlexCAN Configuration Register (CANMCRn)

Table 25-2. CANMCR*n* Field Descriptions

Field	Description
31 MDIS	Module disable. This bit controls whether FlexCAN is enabled or not. When disabled, FlexCAN shuts down the FlexCAN clocks that drive the CAN interface and Message Buffer sub-module. This is the only bit in CANMCR*n* not affected by soft reset. See Section 25.1.3.3, "Module Disabled Mode," for more information. 0 Enable the FlexCAN module, clocks enabled 1 Disable the FlexCAN module, clocks disabled
30 FRZ	Freeze mode enable. When set, the FlexCAN can enter freeze mode when the $\overline{\text{BKPT}}$ line is asserted or the HALT bit is set. Clearing this bit causes the FlexCAN to exit freeze mode. Refer to Section 25.1.3.2, "Freeze Mode," for more information. 0 FlexCAN ignores the $\overline{\text{BKPT}}$ signal and the CANMCR*n*[HALT] bit. 1 FlexCAN module enabled to enter debug mode.
29	Reserved, should be cleared.
28 HALT	Halt FlexCAN. Setting this bit puts the FlexCAN module into freeze mode. It has the same effect as assertion of the $\overline{\text{BKPT}}$ signal. This bit is set after reset and should be cleared after initializing the message buffers and control registers. FlexCAN message buffer receive and transmit functions are inactive until this bit is cleared. While in freeze mode, the CPU has write access to the error counter register (ERRCNT*n*), that is otherwise read-only. 0 The FlexCAN operates normally 1 FlexCAN enters freeze mode if FRZ = 1
27 NOTRDY	FlexCAN not ready. This bit indicates that the FlexCAN is either in disable or freeze mode. This bit is read-only and it is cleared once the FlexCAN exits these modes. 0 FlexCAN is either in normal mode, listen-only mode, or loop-back mode. h1 FlexCAN is in disable or freeze mode.
26	Reserved, should be cleared.
25 SOFTRST	Soft reset. When set, the FlexCAN resets its internal state machines (sequencer, error counters, error flags, and timer) and the host interface registers (CANMCR*n* [except the MDIS bit], TIMER, ERRCNT, ERRSTAT, IMASK, and IFLAG). The configuration registers that control the interface with the CAN bus are not changed (CANCTRL*n*, RXGMASK*n*, RX14MASK*n*, RX15MASK*n*). Message buffers are also not changed. This allows SOFTRST to be used as a debug feature while the system is running. Since soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFTRST bit remains set while reset is pending and is automatically cleared when reset completes. The user should poll this bit to know when the soft reset has completed. 0 Soft reset cycle completed 1 Soft reset cycle initiated
24 FRZACK	Freeze acknowledge. Indicates that the FlexCAN module has entered freeze mode. The user should poll this bit after freeze mode has been requested, to know when the module has actually entered freeze mode. When freeze mode is exited, this bit is cleared once the FlexCAN prescaler is enabled. This is a read-only bit. 0 The FlexCAN has exited freeze mode and the prescaler is enabled. 1 The FlexCAN has entered freeze mode, and the prescaler is disabled.
23 SUPV	Supervisor/user data space. Places the FlexCAN registers in either supervisor or user data space. 0 Registers with access controlled by the SUPV bit are accessible in either user or supervisor privilege mode. 1 Registers with access controlled by the SUPV bit are restricted to supervisor mode.
22–21	Reserved, should be cleared.

Table 25-2. CANMCR*n* Field Descriptions (continued)

Field	Description
20 LPMACK	Low power mode acknowledge. Indicates that FlexCAN is disabled. Disabled mode cannot be entered until all current transmission or reception processes have finished, so the CPU can poll the LPMACK bit to know when the FlexCAN has actually entered low power mode. See Section 25.1.3.3, "Module Disabled Mode," for more information. This bit is read-only. 0 FlexCAN not disabled. 1 FlexCAN is in disabled mode.
19–	Reserved, should be cleared.
–0 MAXMB	Maximum number of message buffers. Defines the maximum number of message buffers that will take part in the matching and arbitration process. The reset value (0xF) is equivalent to message buffer (MB) configuration. This field should be changed only while the module is in freeze mode. **Note:** Maximum MBs in Use = MAXMB + 1

25.3.2   FlexCAN Control Register (CANCTRL*n*)

CANCTRL*n* is defined for specific FlexCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, loop back mode, listen-only mode, bus off recovery behavior, and interrupt enabling. It also determines the division factor for the clock prescaler. Most of the fields in this register should only be changed while the module is disabled or in freeze mode. Exceptions are the BOFFMSK, ERRMSK, and BOFFREC bits which can be accessed at any time.

IPSBAR 0x1C_0004 (CANCTRL0)   Access: User read/write
  Offset:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W				PRESDIV					RJW			PSEG1			PSEG2	
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	BOFF MSK	ERR MSK	CLK_ SRC	LPB	0	0	0	0	SMP	BOFF REC	TSYN	LBUF	LOM	PROPSEG		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-5. FlexCAN Control Register (CANCTRL*n*)

Table 25-3. CANCTRL*n* Field Descriptions

Field	Description
31–24 PRESDIV	Prescaler division factor. Defines the ratio between the clock source frequency (set by CLK_SRC bit) and the serial clock (S clock) frequency. The S clock period defines the time quantum of the CAN protocol. For the reset value, the S clock frequency is equal to the clock source frequency. The maximum value of this register is 0xFF, that gives a minimum S clock frequency equal to the clock source frequency divided by 256. For more information refer to Section 25.4.8, "Bit Timing." $$\text{S clock frequency} = \frac{f_{sys/2} \text{ or EXTAL}}{\text{PRESDIV} + 1} \qquad \textit{Eqn. 25-1}$$ *Eqn. 25-2*
23–22 RJW	Resynchronization jump width. Defines the maximum number of time quanta (one time quantum is equal to the S clock period) that a bit time can be changed by one resynchronization. The valid programmable values are 0–3. Resync jump width = (RJW + 1) time quanta
21–19 PSEG1	Phase buffer segment 1. Defines the length of phase buffer segment 1 in the bit time. The valid programmable values are 0–7. Phase buffer segment 1 = (PSEG1 + 1) time quanta
18–16 PSEG2	Phase buffer segment 2. Defines the length of phase buffer segment 2 in the bit time. The valid programmable values are 1–7. Phase buffer segment 2 = (PSEG2 + 1) time quanta
15 BOFFMSK	Bus off interrupt mask. 0 Bus off interrupt disabled 1 Bus off interrupt enabled
14 ERRMSK	Error interrupt mask. 0 Error interrupt disabled 1 Error interrupt enabled
13 CLK_SRC	Clock source. Selects the clock source for the CAN interface to be fed to the prescalar. This bit should only be changed while the module is disabled. 0 Clock source is EXTAL 1 Clock source is the internal bus clock, $f_{sys/2}$
12 LPB	Loop back. Configures FlexCAN to operate in loop-back mode. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. 0 Loop back disabled 1 Loop back enabled
11–8	Reserved, should be cleared.
7 SMP	Sampling mode. Determines whether the FlexCAN module will sample each received bit one time or three times to determine its value. 0 One sample, taken at the end of phase buffer segment 1, is used to determine the value of the received bit. 1 Three samples are used to determine the value of the received bit. The samples are taken at the normal sample point and at the two preceding periods of the S-clock; a majority rule is used.

Table 25-3. CANCTRL*n* Field Descriptions (continued)

Field	Description
6 BOFFREC	Bus off recovery mode. Defines how FlexCAN recovers from bus off state. If this bit is cleared, automatic recovering from bus off state occurs according to the *CAN Specification 2.0B*. If the bit is set, automatic recovering from bus off is disabled and the module remains in bus off state until the bit is cleared by the user. If the bit is cleared before 128 sequences of 11 recessive bits are detected on the CAN bus, then bus off recovery happens as if the BOFFREC bit had never been set. If the bit is cleared after 128 sequences of 11 recessive bits occurred, then FlexCAN will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After clearing, the BOFFREC bit can be set again during bus off, but it will only be effective the next time the module enters bus off. If BOFFREC was cleared when the module entered bus off, setting it during bus off will not be effective for the current bus off recovery. 0 Automatic recovering from bus off state enabled, according to CAN Spec 2.0B 1 Automatic recovering from bus off state disabled
5 TSYN	Timer synchronize mode. Enables the mechanism that resets the free-running timer each time a message is received in Message Buffer 0. This feature provides the means to synchronize multiple FlexCAN stations with a special "SYNC" message (global network time). 0 Timer synchronization disabled. 1 Timer synchronization enabled. Note: There can be a bit clock skew of four to five counts between different FlexCAN modules that are using this feature on the same network.
4 LBUF	Lowest buffer transmitted first. Defines the ordering mechanism for message buffer transmission. 0 Message buffer with lowest ID is transmitted first 1 Lowest numbered buffer is transmitted first
3 LOM	Listen-only mode. Configures FlexCAN to operate in listen-only mode. In this mode transmission is disabled, all error counters are frozen, and the module operates in a CAN error passive mode. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message. 0 FlexCAN module is in normal active operation; listen-only mode is deactivated 1 FlexCAN module is in listen-only mode operation
2–0 PROPSEG	Propagation segment. Defines the length of the propagation segment in the bit time. The valid programmable values are 0–7. Propagation segment time = (PROPSEG + 1) time-quanta **Note:** A time-quantum = 1 S clock period.

25.3.3   FlexCAN Free Running Timer Register (TIMER*n*)

This register represents a 16-bit free running counter that can be read and written to by the CPU. The timer starts from 0x0000 after reset, counts linearly to 0xFFFF, and wraps around.

The timer is clocked by the FlexCAN bit-clock (which defines the baud rate on the CAN bus). During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. During freeze mode, the timer is not incremented.

The timer value is captured at the beginning of the identifier (ID) field of any frame on the CAN bus. This captured value is written into the TIMESTAMP entry in a message buffer after a successful reception or transmission of a message.

Writing to the timer is an indirect operation. The data is first written to an auxiliary register and then an internal request/acknowledge procedure across clock domains is executed. All this is transparent to the

user, except for the fact that the data will take some time to be actually written to the register. If desired, software can poll the register to discover when the data was actually written.

IPSBAR 0x1C_0008 (TIMER0)
Offset:

Access: User read/write
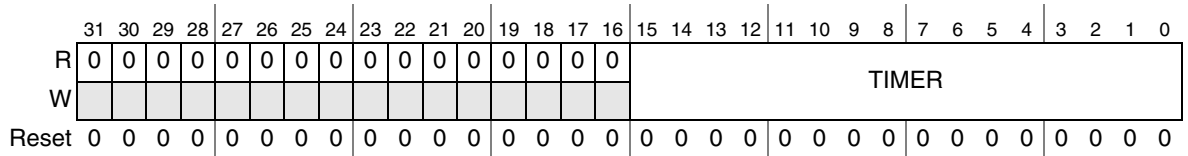
Figure 25-6. FlexCAN Timer Register (TIMER*n*)

Table 25-4. TIMER*n* Field Descriptions

Field	Description
31–16	Reserved, should be cleared.
15–0 TIMER	Free running timer. Captured at the beginning of the identifier (ID) field of any frame on the CAN bus. This captured value is written into the TIMESTAMP entry in a message buffer after a successful reception or transmission of a message.

25.3.4   Rx Mask Registers (RXGMASK*n*, RX14MASK*n*, RX15MASK*n*)

These registers are used as acceptance masks for received frame IDs. Three masks are defined: A global mask (RXGMASK*n*) used for Rx buffers 0–13 and two separate masks for buffers 14 (RX14MASK*n*) and 15 (RX15MASK*n*). The meaning of each mask bit is the following:

MI*n* bit = 0: The corresponding incoming ID bit is "don't care".

MI*n* bit = 1: The corresponding ID bit is checked against the incoming ID bit, to see if a match exists.

Note that these masks are used both for Standard and Extended ID formats. The value of the mask registers should not be changed while in normal operation (only while in freeze mode), as locked frames that matched a message buffer (MB) through a mask may be transferred into the MB (upon release) but may no longer match.

Table 25-5. Mask Examples for Normal/Extended Messages

	Base ID ID28.................ID18	IDE	Extended ID ID17..ID0	Match
MB2-ID	1 1 1 1 1 1 1 1 0 0 0	0		
MB3-ID	1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	
MB4-ID	0 0 0 0 0 0 1 1 1 1 1	0		
MB5-ID	0 0 0 0 0 0 1 1 1 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	
MB14-ID	1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	
Rx_Global_Mask	1 1 1 1 1 1 1 1 1 1 0		1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1	
Rx_Msg in[1]	1 1 1 1 1 1 1 1 0 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	MB3[1]
Rx_Msg in[2]	1 1 1 1 1 1 1 1 0 0 1	0		MB2[2]

Table 25-5. Mask Examples for Normal/Extended Messages (continued)

	Base ID ID28.................ID18	IDE	Extended ID ID17.......................................ID0	Match
Rx_Msg in[3]	1 1 1 1 1 1 1 1 0 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0	3
Rx_Msg in[4]	0 1 1 1 1 1 1 1 0 0 0	0		4
Rx_Msg in[5]	0 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	MB14[5]
RX14MASK	0 1 1 1 1 1 1 1 1 1 1		1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0	
Rx_Msg in[6]	1 0 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	6
Rx_Msg in[7]	0 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	MB14[7]

[1] Match for Extended Format (MB3).
[2] Match for Normal Format. (MB2).
[3] Mismatch for MB3 because of ID0.
[4] Mismatch for MB2 because of ID28.
[5] Mismatch for MB3 because of ID28, Match for MB14 (Uses RX14MASK*n*).
[6] Mismatch for MB14 because of ID27 (Uses RX14MASK*n*).
[7] Match for MB14 (Uses RX14MASK*n*).

Offset: 0x1C_0010 (RXGMASK0)   Access: User read/write
        0x1C_0014 (RX14MASK0)
        0x1C_0018 (RX15MASK0)

	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					MI Standard ID													MI Extended ID											
W																																
Reset	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
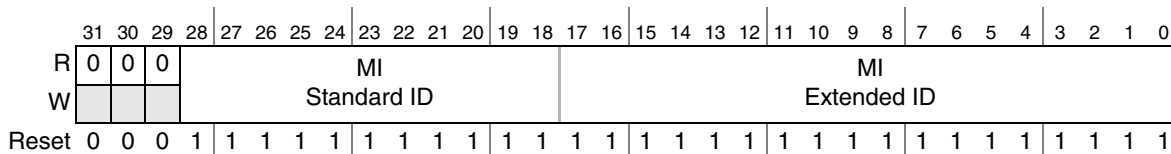
Figure 25-7. FlexCAN Rx Mask Registers (RXGMASK*n*, RX14MASK*n*, RX15MASK*n*)

Table 25-6. RX*xx*MASK*n* Field Descriptions

Field	Description
31–29	Reserved, should be cleared.
28–18 MI28–18	Standard ID mask bits. These bits are the same mask bits for the Standard and Extended Formats.
17–0 MI17–0	Extended ID mask bits. These bits are used to mask comparison only in Extended Format.

25.3.5 FlexCAN Error Counter Register (ERRCNT*n*)

This register has two 8-bit fields reflecting the value of two FlexCAN error counters: transmit error counter (TXECTR) and receive error counter (RXECTR). The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FlexCAN module. Both counters are read-only except in freeze mode, where they can be written by the CPU.

Writing to the ERRCNT*n* register while in freeze mode is an indirect operation. The data is first written to an auxiliary register and then an internal request/acknowledge procedure across clock domains is executed. All this is transparent to the user, except for the fact that the data will take some time to be actually written to the register. If desired, software can poll the register to discover when the data was actually written.

FlexCAN responds to any bus state as described in the protocol, e.g. transmit error-active or error-passive flag, delay its transmission start time (error-passive), and avoid any influence on the bus when in bus off state. The following are the basic rules for FlexCAN bus state transitions.

- If the value of TXECTR or RXECTR increases to be greater than or equal to 128, the FLTCONF field in the error and status register (ERRSTAT*n*) is updated to reflect error-passive state.

- If the FlexCAN state is error-passive, and either TXECTR or RXECTR decrements to a value less than or equal to 127 while the other already satisfies this condition, the ERRSTAT*n*[FLTCONF] field is updated to reflect error-active state.

- If the value of TXECTR increases to be greater than 255, the ERRSTAT*n*[FLTCONF] field is updated to reflect bus off state, and an interrupt may be issued. The value of TXECTR is then reset to zero.

- If FlexCAN is in bus off state, then TXECTR is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, TXECTR is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the TXECTR. When TXECTR reaches the value of 128, the ERRSTAT*n*[FLTCONF] field is updated to be error-active and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the TXECTR value.

- If during system start-up, only one node is operating, then its TXECTR increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ERRSTAT*n*[ACKERR] bit). After the transition to error-passive state, the TXECTR does not increment anymore by acknowledge errors. Therefore the device never goes to the bus off state.

- If the RXECTR increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume to error-active state.

IPSBAR 0x1C_001C (ERRCNT0)   Access: User read/write
Offset:

	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	0	0	0	0	0	0	0	0	RXECTR	TXECTR
W																		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0

Figure 25-8. FlexCAN Error Counter Register (ERRCNT*n*)

Table 25-7. ERRCNT*n* Field Descriptions

Field	Description
31–16	Reserved, should be cleared.
15–8 RXECTR	Receive error counter. Indicates current number of receive errors.
7–0 TXECTR	Transmit error counter. Indicates current number of transmit errors.

25.3.6   FlexCAN Error and Status Register (ERRSTAT*n*)

ERRSTAT*n* reflects various error conditions, some general status of the device, and is the source of three interrupts to the CPU. The reported error conditions (bits 15:10) are those occurred since the last time the CPU read this register. The read action clears bits 15-10. Bits 9–3 are status bits.

Most bits in this register are read only, except for BOFFINT, WAKINT, and ERRINT, which are interrupt flags that can be cleared by writing 1 to them. Writing 0 has no effect. Refer to Section 25.5.1, "Interrupts."

IPSBAR 0x1C_0020 (CANCTRL0)   Access: User read/write
  Offset:

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BIT1 ERR	BIT0 ERR	ACK ERR	CRC ERR	FRM ERR	STF ERR	TX WRN	RX WRN	IDLE	TXRX	FLT CONF		0	BOFF INT	ERR INT	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-9. FlexCAN Error and Status Register (ERRSTAT*n*)

Table 25-8. ERRSTAT*n* Field Descriptions

Field	Description
31–16	Reserved, should be cleared.
15 BIT1ERR	Bit1 error. Indicates inconsistency between the transmitted and received bit in a message. 0 No transmit bit error 1 At least one bit sent as recessive was received as dominant **Note:** The transmit bit error field is not modified during the arbitration field or the ACK slot bit time of a message, or by a transmitter that detects dominant bits while sending a passive error frame.
14 BIT0ERR	Bit0 error. Indicates inconsistency between the transmitted and received bit in a message. 0 No transmit bit error 1 At least one bit sent as dominant was received as recessive
13 ACKERR	Acknowledge error. Indicates whether an acknowledgment has been correctly received for a transmitted message. 0 No ACK error was detected since the last read of this register. 1 An ACK error was detected since the last read of this register.

Table 25-8. ERRSTAT*n* Field Descriptions (continued)

Field	Description
12 CRCERR	Cyclic redundancy check error. Indicates whether or not a CRC error has been detected by the receiver. 0 No CRC error was detected since the last read of this register. 1 A CRC error was detected since the last read of this register.
11 FRMERR	Message form error. 0 No form error was detected since the last read of this register. 1 A form error was detected since the last read of this register.
10 STFERR	Bit stuff error. 0 No bit stuffing error was detected since the last read of this register. 1 A bit stuffing error was detected since the last read of this register.
9 TXWRN	Transmit error status flag. Reflects the status of the FlexCAN transmit error counter. 0 Transmit error counter < 96 1 TXErrCounter ≥ 96
8 RXWRN	Receiver error status flag. Reflects the status of the FlexCAN receive error counter. 0 Receive error counter < 96 1 RxErrCounter ≥ 96
7 IDLE	Idle status. Indicates when there is activity on the CAN bus. 0 The CAN bus is not idle. 1 The CAN bus is idle.
6 TXRX	Transmit/receive status. Indicates when the FlexCAN module is transmitting or receiving a message. TXRX has no meaning when IDLE = 1. 0 The FlexCAN is receiving a message if IDLE = 0. 1 The FlexCAN is transmitting a message if IDLE = 0.
5–4 FLTCONF	Fault confinement state. Indicates the confinement state of the FlexCAN module, as shown below. If the CANCTRL*n*[LOM] bit is set, FLTCONF will indicate error-passive. Since the CANCTRL*n* register is not affected by soft reset, the FLTCONF field will not be affected by soft reset if the LOM bit is set. 00 Error active 01 Error passive 1x Bus off
3	Reserved, should be cleared.
2 BOFFINT	Bus off interrupt. Used to request an interrupt when the FlexCAN enters the bus off state. The user must write a 1 to clear this bit. Writing 0 has no effect. 0 No bus off interrupt requested. 1 This bit is set when the FlexCAN state changes to bus off. If the CANCTRL*n*[BOFFMSK] bit is set an interrupt request is generated. This interrupt is not requested after reset.
1 ERRINT	Error interrupt. Indicates that at least one of the ERRSTAT*n*[15:10] bits is set. The user must write a 1 to clear this bit. Writing 0 has no effect. 0 No error interrupt request. 1 At least one of the error bits is set. If the CANCTRL*n*[ERRMSK] bit is set, an interrupt request is generated.
0	Reserved, should be cleared.

25.3.7   Interrupt Mask Register (IMASK*n*)

IMASK*n* contains one interrupt mask bit per buffer. It enables the CPU to determine which buffer will generate an interrupt after a successful transmission/reception (that is, when the corresponding IFLAG*n* bit is set).

IPSBAR 0x1C_002A (IMASK0)   Access: User read/write
Offset:

	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
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 25-10. FlexCAN Interrupt Mask Register (IMASK*n*)

Table 25-9. IMASK*n* Field Descriptions

Field	Description
–0 BUF*n*M	Buffer interrupt mask. Enables the respective FlexCAN message buffer (MB0 to MB) interrupt. These bits allow the CPU to designate which buffers will generate interrupts after successful transmission/reception. 0 The interrupt for the corresponding buffer is disabled. 1 The interrupt for the corresponding buffer is enabled. **Note:** Setting or clearing an IMASK*n* bit can assert or negate an interrupt request, if the corresponding IFLAG*n* bit it is set.

25.3.8   Interrupt Flag Register (IFLAG*n*)

IFLAG*n* contains one interrupt flag bit per buffer. Each successful transmission/reception sets the corresponding IFLAG*n* bit and, if the corresponding IMASK*n* bit is set, will generate an interrupt.

The interrupt flag is cleared by writing a 1, while writing 0 has no effect.

IPSBAR 0x1C_0030 (IFLAG0)   Access: User read/write
Offset:

	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
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 25-11. FlexCAN Interrupt Flags Register (IFLAG*n*)

Table 25-10. IFLAG*n* Field Descriptions

Field	Description
–0 BUF*n*I	Buffer interrupt flag. Indicates a successful transmission/reception for the corresponding message buffer. If the corresponding IMASK*n* bit is set, an interrupt request will be generated. The user must write a 1 to clear an interrupt flag; writing 0 has no effect. 0 No such occurrence. 1 The corresponding buffer has successfully completed transmission or reception.

25.3.9   Message Buffer Structure

The message buffer memory map starts at an offset of 0x80 from the FlexCAN's base address (CAN0: 0x1C_0000. The -byte message buffer space is fully used by the message buffer structures.

Each message buffer consists of a control and status field that configures the message buffer, an identifier field for frame identification, and up to eight bytes of data.
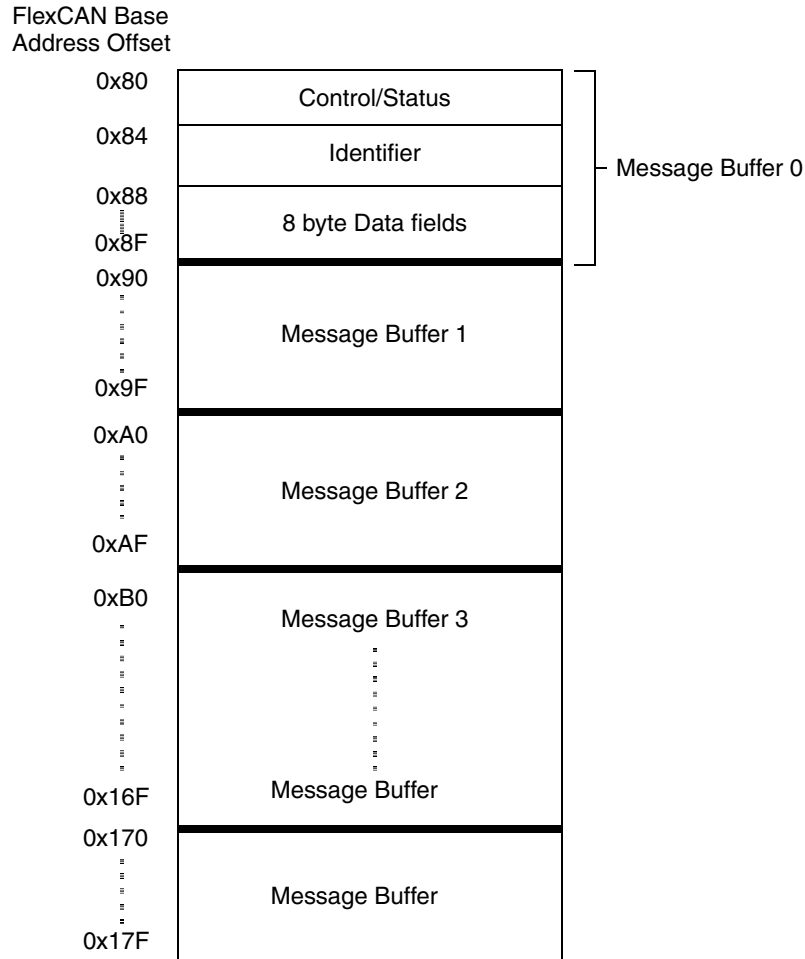
Figure 25-12. FlexCAN Message Buffer Memory Map

The message buffer structure used by the FlexCAN module is shown in Figure 25-13. Both standard and extended frames used in the *CAN Specification Version 2.0, Part B* are represented. A standard frame is represented by the 11-bit standard identifier, and an extended frame is represented by the combined 29-bits of the standard identifier (11 bits) and the extended identifier (18 bits).
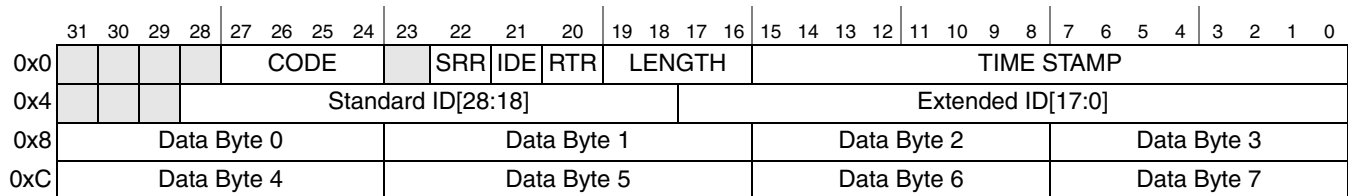
Figure 25-13. Message Buffer Structure for Both Extended and Standard Frames

Table 25-11. Message Buffer Field Descriptions

Field	Description
31–28	Reserved, should be cleared.
27–24 CODE	Message buffer code. Can be accessed (read or write) by the CPU and by the FlexCAN module itself, as part of the message buffer matching and arbitration process. The encoding is shown in Table 25-12 and Table 25-13. See Section 25.4, "Functional Overview," for additional information.
23	Reserved, should be cleared.
22 SRR	Substitute remote request. Fixed recessive bit, used only in extended format. It must be set by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FlexCAN receives this bit as dominant, then it is interpreted as arbitration loss. 0 Dominant is not a valid value for transmission in Extended Format frames 1 Recessive value is compulsory for transmission in Extended Format frames
21 IDE	ID extended bit. Identifies whether the frame format is standard or extended. 0 Standard frame format 1 Extended frame format
20 RTR	Remote transmission request. Used for requesting transmissions of a data frame. If FlexCAN transmits this bit as 1 (recessive) and receives it as 0 (dominant), it is interpreted as arbitration loss. If this bit is transmitted as 0 (dominant), then if it is received as 1 (recessive), the FlexCAN module treats it as bit error. If the value received matches the value transmitted, it is considered as a successful bit transmission. 0 Indicates the current MB has a data frame to be transmitted 1 Indicates the current MB has a remote frame to be transmitted
19–16 LENGTH	Length of data in bytes. Indicates the length (in bytes) of the Rx or Tx data; data is located in offset 0x8 through 0xF of the MB space (see Figure 25-13). In reception, this field is written by the FlexCAN module, copied from the DLC (data length code) field of the received frame. DLC is defined by the *CAN Specification* and refers to the data length of the actual frame before it is copied into the message buffer. In transmission, this field is written by the CPU and is used as the DLC field value of the frame to be transmitted. When RTR is set, the frame to be transmitted is a remote frame and will be transmitted without the DATA field, regardless of the LENGTH field.
15–0 TIME STAMP	Free-running counter time stamp. Stores the value of the free-running timer which is captured when the beginning of the identifier (ID) field appears on the CAN bus.
31–29	Reserved, should be cleared.
28–0 ID	Standard frame identifier: In standard frame format, only the 11 most significant bits (28 to 18) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored.
	Extended frame identifier: In extended frame format, all bits (both the 11 bits of the standard frame identifier and the 18 bits of the extended frame identifier) are used for frame identification in both receive and transmit cases.
31–24, 23–16, 15–8, 7–0 DATA	Data field. Up to eight bytes can be used for a data frame. For Rx frames, the data is stored as it is received from the CAN bus. For Tx frames, the CPU provides the data to be transmitted within the frame.

Table 25-12. Message Buffer Code for Rx Buffers

Rx Code BEFORE Rx New Frame	Description	Rx Code AFTER Rx New Frame	Comment
0000	INACTIVE: MB is not active.	—	MB does not participate in the matching process.
0100	EMPTY: MB is active and empty.	0010	MB participates in the matching process. When a frame is received successfully, the code is automatically updated to FULL.
0010	FULL: MB is full.	0010	The act of reading the control & status (C/S) word followed by unlocking the MB does not make the code return to EMPTY. It remains FULL. If a new frame is written to the MB after the C/S word was read and the MB was unlocked, the code still remains FULL.
		0110	If the MB is FULL and a new frame should be written into this MB before the CPU had time to read it, the MB is overwritten, and the code is automatically updated to OVERRUN.
0110	OVERRUN: A frame was overwritten into a full buffer.	0010	If the code indicates OVERRUN but the CPU reads the C/S word and then unlocks the MB, when a new frame is written to the MB, the code returns to FULL.
		0110	If the code already indicates OVERRUN, and yet another new frame must be written, the MB will be overwritten again, and the code will remain OVERRUN.
0XY1[1]	BUSY: Flexcan is updating the contents of the MB with a new receive frame. The CPU should not try to access the MB.	0010	An EMPTY buffer was written with a new frame (XY was 01).
		0110	A FULL/OVERRUN buffer was overwritten (XY was 11).

[1] Note that for transmit message buffers (see Table 25-13), the BUSY bit should be ignored upon read.

Table 25-13. Message Buffer Code for Tx Buffers

MB*n*[RTR]	Initial Tx Code	Code After Successful Transmission	Description
X	1000	—	INACTIVE: Message buffer not ready for transmit and will participate in the arbitration process.
0	1100	1000	Data frame to be transmitted once, unconditionally. After transmission, the MB automatically returns to the INACTIVE state.
1	1100	0100	Remote frame to be transmitted unconditionally once, and message buffer becomes an Rx message buffer with the same ID for data frames.

Table 25-13. Message Buffer Code for Tx Buffers (continued)

MB*n*[RTR]	Initial Tx Code	Code After Successful Transmission	Description
0	1010	1010	Transmit a data frame whenever a remote request frame with the same ID is received. This message buffer participates simultaneously in both the matching and arbitration processes. The matching process compares the ID of the incoming remote request frame with the ID of the MB. If a match occurs, this message buffer is allowed to participate in the current arbitration process and the CODE field is automatically updated to 1110 to allow the MB to participate in future arbitration runs. When the frame is eventually transmitted successfully, the code automatically returns to 1010 to restart the process again.
0	1110	1010	This is an intermediate code that is automatically written to the message buffer as a result of match to a remote request frame. The data frame will be transmitted unconditionally once, and then the code will automatically return to 1010. The CPU can also write this code with the same effect.

25.4   Functional Overview

The FlexCAN module is flexible in that each one of its 16 message buffers (MBs) can be assigned either as a transmit buffer or a receive buffer. Each MB, which is up to 8 bytes long, is also assigned an interrupt flag bit that indicates successful completion of either transmission or reception.

An arbitration algorithm decides the prioritization of MBs to be transmitted based on either the message ID or the MB ordering. A matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed on its ID field. A masking scheme makes it possible to match the ID programmed on the MB with a range of IDs on received CAN frames. Data coherency mechanisms are implemented to guarantee data integrity during MB manipulation by the CPU.

Before proceeding with the functional description, an important concept must be explained. A message buffer is said to be "active" at a given time if it can participate in the matching and arbitration algorithms that are happening at that time. An Rx MB with a 0000 code is inactive (refer to Table 25-12). Similarly, a Tx MB with a 1000 code is inactive (refer to Table 25-13). An MB not programmed with either 0000 or 1000 will be temporarily deactivated (will not participate in the current arbitration/matching run) when the CPU writes to the C/S field of that MB.

25.4.1   Transmit Process

The CPU prepares or changes an MB for transmission by executing the following steps:

1. Writing the control/status word to hold Tx MB inactive (CODE = 1000).
2. Writing the ID word.
3. Writing the data bytes.
4. Writing the control/status word (active CODE, LENGTH).

NOTE

The first and last steps are mandatory!

The first write to the control/status word is important in case there was pending reception or transmission. The write operation immediately deactivates the MB, removing it from any currently ongoing arbitration or ID matching processes, giving time for the CPU to program the rest of the MB (see Section 25.4.5.2, "Message Buffer Deactivation"). Once the MB is activated in the fourth step, it will participate in the arbitration process and eventually be transmitted according to its priority. At the end of the successful transmission, the value of the free running timer (TIMERn) is written into the message buffer's Time Stamp field, the Code field in the Control and Status word is updated, a status flag is set in the IFLAGn register and an interrupt is generated if allowed by the corresponding IMASKn register bit. The new Code field after transmission depends on the code that was used to activate the MB in step four (see Table 25-13).

25.4.2   Arbitration Process

The arbitration process is an algorithm executed by the message buffer management (MBM) that scans the entire MB memory looking for the highest priority message to be transmitted. All MBs programmed as transmit buffers will be scanned to find the lowest ID or the lowest MB number, depending on the CANCTRLn[LBUF] bit.

NOTE

If CANCTRLn[LBUF] is cleared, the arbitration considers not only the ID, but also the RTR and IDE bits placed inside the ID at the same positions they are transmitted in the CAN frame.

The arbitration process is triggered in the following events:

- During the CRC field of the CAN frame
- During the error delimiter field of the CAN frame
- During intermission, if the winner MB defined in a previous arbitration was deactivated, or if there was no MB to transmit, but the CPU wrote to the C/S word of any MB after the previous arbitration finished
- When MBM is in idle or bus off state and the CPU writes to the C/S word of any MB
- Upon leaving freeze mode

Once the highest priority MB is selected, it is transferred to a temporary storage space called serial message buffer (SMB), which has the same structure as a normal MB but is not user accessible. This operation is called "move-out." At the first opportunity window on the CAN bus, the message on the SMB is transmitted according to the CAN protocol rules. FlexCAN transmits up to eight data bytes, even if the DLC (data length code) value is bigger. Refer to Section 25.4.5.1, "Serial Message Buffers (SMBs)," for more information on serial message buffers.

25.4.3   Receive Process

The CPU prepares or changes an MB for frame reception by executing the following steps:

1. Writing the control/status word to hold Rx MB inactive (CODE = 0000).

2. Writing the ID word.

3. Writing the control/status word to mark the Rx MB as active and empty (CODE = 1000).

NOTE

The first and last steps are mandatory!

The first write to the control/status word is important in case there was a pending reception or transmission. The write operation immediately deactivates the MB, removing it from any currently ongoing arbitration or matching process, giving time for the CPU to program the rest of the MB. Once the MB is activated in the third step, it will be able to receive CAN frames that match the programmed ID. At the end of a successful reception, the value of the free running timer (TIMERn) is written into the time stamp field, the received ID, data (8 bytes at most) and length fields are stored, the CODE field in the control and status word is updated (see Table 25-12), and a status flag is set in the IFLAGn register and an interrupt is generated if allowed by the corresponding IMASKn bit.

The CPU should read a receive frame from its MB in the following way:

1. Read the control/status word (mandatory—activates internal lock for this buffer).

2. Read the ID (optional—needed only if a mask was used).

3. Read the data field words.

4. Read the free-running timer (Releases internal lock —optional).

Upon reading the control and status word, if the BUSY bit is set in the CODE field, then the CPU should defer the access to the MB until this bit is negated. Reading the free running timer is not mandatory. If not executed the MB remains locked, unless the CPU reads the C/S word of another MB. Note that only a single MB is locked at a time. The only mandatory CPU read operation is the one on the control and status word to assure data coherency.

The CPU should synchronize to frame reception by an IFLAGn bit for the specific MB (see Section 25.3.8, "Interrupt Flag Register (IFLAGn)"), and not by the control/status word CODE field for that MB. Polling the CODE field does not work because once a frame was received and the CPU services the MB (by reading the C/S word followed by unlocking the MB), the CODE field will not return to EMPTY. It will remain FULL, as explained in Table 25-12. If the CPU tries to workaround this behavior by writing to the C/S word to force an EMPTY code after reading the MB, the MB is actually deactivated from any currently ongoing matching process. As a result, a newly received frame matching the ID of that MB may be lost. In summary, never do polling by directly reading the C/S word of the MBs. Instead, read the IFLAGn register.

Note that the received identifier field is always stored in the matching MB, thus the contents of the ID field in an MB may change if the match was due to masking.

25.4.3.1 Self-Received Frames

Self-received frames are frames that are sent by the FlexCAN and received by itself. The FlexCAN sends a frame externally through the physical layer onto the CAN bus, and if the ID of the frame matches the ID of the FlexCAN MB, then the frame will be received by the FlexCAN. Such a frame is a self-received frame. Note that FlexCAN does not receive frames transmitted by itself if another device on the CAN bus has an ID that matches the FlexCAN Rx MB ID.

25.4.4 Matching Process

The matching process is an algorithm that scans the entire MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. Only MBs programmed to receive will participate in the matching process for received frames.

While the ID, DLC and data fields are retrieved from the CAN bus, they are stored temporarily in the Serial Message Buffer (Section 25.4.5.1, "Serial Message Buffers (SMBs)"). The matching process takes place during the CRC field. If a matching ID is found in one of the MBs, the contents of the SMB will be transferred to the matched MB during the sixth bit of the end-of-frame field of the CAN protocol. This operation is called "move-in". If any protocol error (CRC, ACK, etc.) is detected, than the move-in operation does not happen.

An MB with a matching ID is "free to receive" a new frame if the MB is not locked (see Section 25.4.5.3, "Locking and Releasing Message Buffers") and the CODE field is either EMPTY or else it is FULL or OVERRUN but the CPU has already serviced the MB (read the C/S word and then unlocked the MB).

Matching to a range of IDs is possible by using ID acceptance masks (RXGMASK*n*, RX14MASK*n*, and RX15MASK*n*). During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is "don't care".

25.4.5 Message Buffer Handling

In order to maintain data coherency and FlexCAN proper operation, the CPU must obey the rules described in Section 25.4.1, "Transmit Process" and Section 25.4.3, "Receive Process." Any form of CPU accessing a MB structure within FlexCAN other than those specified may cause FlexCAN to behave in an unpredictable way.

25.4.5.1 Serial Message Buffers (SMBs)

To allow double buffering of messages, the FlexCAN has two shadow buffers called serial message buffers. These two buffers are used by the FlexCAN for buffering both received messages and messages to be transmitted. Only one SMB is active at a time, and its function depends upon the operation of the FlexCAN at that time. At no time does the user have access to or visibility of these two buffers.

25.4.5.2 Message Buffer Deactivation

If the CPU wants to change the function of an active MB, the recommended procedure is to put the module into freeze mode and then change the CODE field of that MB. This is a safe procedure because the FlexCAN waits for pending CAN bus and MB moving activities to finish before entering freeze mode.

Nevertheless, a mechanism is provided to maintain data coherence when the CPU writes to the Control and Status word of active MBs out of freeze mode.

Any CPU write access to the C/S word of an MB causes that MB to be excluded from the transmit or receive processes during the current matching or arbitration round. This mechanism is called MB deactivation. It is temporary, affecting only for the current match/arbitration round.

The purpose of deactivation is data coherency. The match/arbitration process scans the MBs to decide which MB to transmit or receive. If the CPU updates the MB in the middle of a match or arbitration process, the data of that MB may no longer be coherent, therefore that MB is deactivated.

Even with the coherence mechanism described above, writing to the C/S word of active MBs when not in freeze mode may produce undesirable results. Examples are:

- Matching and arbitration are one-pass processes. If MBs are deactivated after they are scanned, no re-evaluation is done to determine a new match/winner. If an Rx MB with a matching ID is deactivated during the matching process after it was scanned, then this MB is marked as invalid to receive the frame, and FlexCAN will keep looking for another matching MB within the ones it has not scanned yet. If it can not find one, then the message will be lost. Suppose, for example, that two MBs have a matching ID to a received frame, and the user deactivated the first matching MB after FlexCAN has scanned the second. The received frame will be lost even if the second matching MB was "free to receive".

- If a Tx MB containing the lowest ID is deactivated after the FlexCAN has scanned it, then the FlexCAN will look for another winner within the MBs that it has not yet scanned. Therefore, it may transmit an MB that may not have the lowest ID at the time because a lower ID might be present that it had already scanned before the deactivation.

- There is a point in time until which the deactivation of a Tx MB causes it not to be transmitted (end of move-out). After this point, it is transmitted but no interrupt is issued and the CODE field is not updated.

25.4.5.3   Locking and Releasing Message Buffers

Besides message buffer deactivation, the lock/release/busy mechanism is designed to guarantee data coherency during the receive process. The following examples demonstrate how the lock/release/busy mechanism will affect FlexCAN operation.

1. Reading a control/status word of a message buffer triggers a lock for that message buffer. A new received message frame that matches the message buffer cannot be written into this message buffer while it is locked.

2. To release a locked message buffer, the CPU either locks another message buffer (by reading its control/status word) or globally releases any locked message buffer (by reading the free-running timer).

3. If a receive frame with a matching ID is received during the time the message buffer is locked, the receive frame will not be immediately transferred into that message buffer, but will remain in the SMB. There is no indication when this occurs.

4. When a locked message buffer is released, if a frame with a matching identifier exists within the SMB, then this frame will be transferred to the matching message buffer.

5. If two or more receive frames with matching IDs are received while a message buffer with a matching ID is locked, the last received frame with that ID is kept within the serial message buffer, while all preceding ones are lost. There is no indication of lost messages when this occurs.

6. If the user reads the control/status word of a receive message buffer while a frame is being transferred from a serial message buffer, the BUSY code will be indicated. The user should wait until this code is cleared before continuing to read from the message buffer to ensure data coherency. In this situation, the read of the control/status word will not lock the message buffer.

Polling the control/status word of a receive message buffer can lock it, preventing a message from being transferred into that buffer. If the control/status word of a receive message buffer is read, it should then be followed by a read of the control/status word of another buffer, or by reading the free-running timer, to ensure that the locked buffer is unlocked.

NOTE

Deactivation takes precedence over locking. If the CPU deactivates a locked Rx MB, then its lock status is negated, and the MB is marked as invalid for the current matching round. Any pending message on the SMB will not be transferred to the MB anymore.

25.4.6 CAN Protocol Related Frames

25.4.6.1 Remote Frames

The remote frame is a message frame which is transmitted to request a data frame. The FlexCAN can be configured to transmit a data frame automatically in response to a remote frame, or to transmit a remote frame and then wait for the responding data frame to be received.

When transmitting a remote frame, the user initializes a message buffer as a transmit message buffer with the RTR bit set. Once this remote frame is transmitted successfully, the transmit message buffer automatically becomes a receive message buffer, with the same ID as the remote frame that was transmitted.

When a remote frame is received by the FlexCAN, the remote frame ID is compared to the IDs of all transmit message buffers programmed with a CODE of 1010. If there is an exact matching ID, the data frame in that message buffer is transmitted. If the RTR bit in the matching transmit message buffer is set, the FlexCAN will transmit a remote frame as a response.

A received remote frame is not stored in a receive message buffer. It is only used to trigger the automatic transmission of a frame in response. The mask registers are not used in remote frame ID matching. All ID bits (except RTR) of the incoming received frame must match for the remote frame to trigger a response transmission. The matching message buffer immediately enters the internal arbitration process, but is considered as a normal Tx MB, with no higher priority. The data length of this frame is independent of the data length code (DLC) field in the remote frame that initiated its transmission.

25.4.6.2 Overload Frames

Overload frame transmissions are not initiated by the FlexCAN unless certain conditions are detected on the CAN bus. These conditions include:

- Detection of a dominant bit in the first or second bit of intermission.
- Detection of a dominant bit in the seventh (last) bit of the end-of-frame (EOF) field in receive frames.
- Detection of a dominant bit in the eighth (last) bit of the error frame delimiter or overload frame delimiter.

25.4.7 Time Stamp

The value of TIMERn is sampled at the beginning of the identifier field on the CAN bus. For a message being received, the time stamp will be stored in the TIMESTAMP entry of the receive message buffer at the time the message is written into that buffer. For a message being transmitted, the TIMESTAMP entry will be written into the transmit message buffer once the transmission has completed successfully.

The free-running timer can optionally be reset upon the reception of a frame into message buffer 0. This feature allows network time synchronization to be performed. See the CANCTRLn[TSYN] bit.

25.4.8 Bit Timing

The FlexCAN module CANCTRLn register configures the bit timing parameters required by the CAN protocol. The CLK_SRC, PRESDIV, RJW, PSEG1, PSEG2, and the PROPSEG fields allow the user to configure the bit timing parameters.

The CANCTRLn[CLK_SRC] bit defines whether the module uses the internal bus clock or the output of the crystal oscillator via the EXTAL pin. The crystal oscillator clock should be selected whenever a tight tolerance (up to 0.1%) is required for the CAN bus timing. The crystal oscillator clock has better jitter performance than PLL generated clocks. The value of this bit should not be changed unless the module is in disable mode (CANMCRn[MDIS] bit is set)

The PRESDIV field controls a prescaler that generates the serial clock (S-clock), whose period defines the "time quantum" used to compose the CAN waveform. A time quantum is the atomic unit of time handled by the CAN engine.
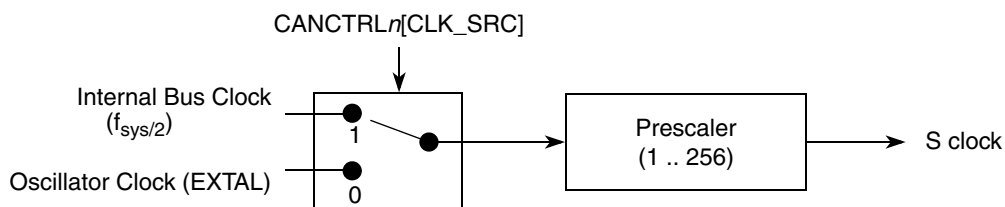
Figure 25-14. CAN Engine Clocking Scheme

$f_{Tq} = \dfrac{f_{sys/2} \text{ or EXTAL}}{(\text{PRESDIV} + 1)}$ A bit time is subdivided into three segments[1] (reference Figure 25-15 and Table 25-14):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section
- Time Segment 1: This segment includes the Propagation Segment and the Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CANCTRL*n* register so that their sum (plus 2) is in the range of 4 to 16 time quanta
- Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CANCTRL*n* register (plus 1) to be 2 to 8 time quanta long

$$\text{Bit Rate} = \frac{f_{Tq}}{(\text{number of Time Quanta})} \qquad \textbf{\textit{Eqn. 25-3}}$$
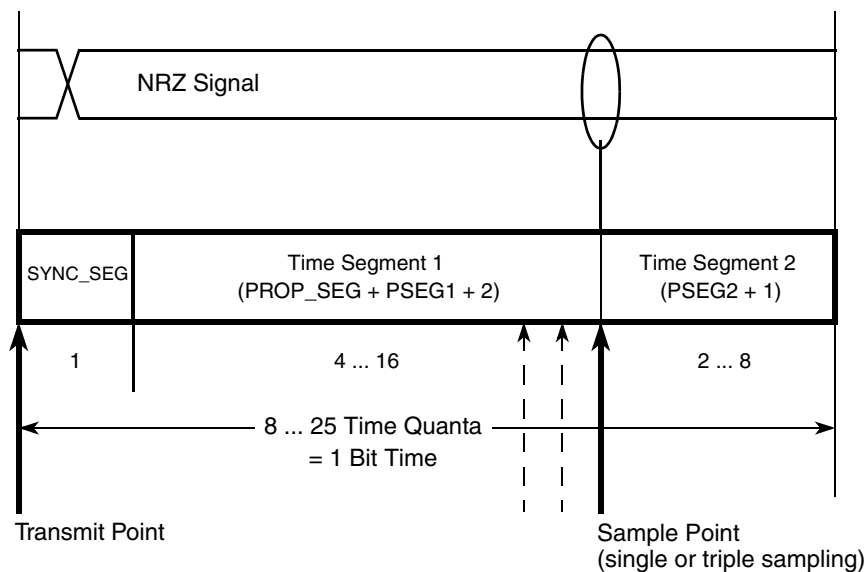
Figure 25-15. Segments within the Bit Time

Table 25-14. Time Segment Syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

Table 25-15 gives an overview of the CAN compliant segment settings and the related parameter values.

---

1. For further explanation of the underlying concepts please refer to ISO/DIS 11519–1, Section 10.3. Reference also the Bosch CAN 2.0A/B protocol specification dated September 1991 for bit timing.

NOTE

It is the user's responsibility to ensure the bit time settings are in compliance with the CAN standard. For bit time calculations, use an IPT (Information Processing Time) of 2, which is the value implemented in the FlexCAN module

Table 25-15. CAN Standard Compliant Bit Time Segment Settings

Time Segment 1	Time Segment 2	Re-synchronization Jump Width
5 .. 10	2	1 .. 2
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

25.5 FlexCAN Initialization Sequence

Initialization of the FlexCAN includes the initial configuration of the message buffers and configuration of the CAN communication parameters following a reset, as well as any reconfiguration which may be required during operation. The FlexCAN module may be reset in three ways:

- Device level hard reset which resets all memory mapped registers asynchronously
- Device level soft reset, which resets some of the memory mapped registers synchronously (refer to Table 25-1 to see which registers are affected by soft reset)
- CANMCRn[SOFT_RST] bit, which has the same effect as the device level soft reset

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The CANMCRn[SOFT_RST] bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset can not be applied while clocks are shut down in any of the low power modes. The low power mode should be exited and the clocks resumed before applying soft reset.

The clock source, CANCTRLn[CLK_SRC], should be selected while the module is in disable mode. After the clock source is selected and the module is enabled (CANMCRn[MDIS] bit cleared), the FlexCAN automatically enters freeze mode. In freeze mode, the FlexCAN is un-synchronized to the CAN bus, the CANMCRn register's HALT and FRZ bits are set, the internal state machines are disabled, and the CANMCRn register's FRZ_ACK and NOT_RDY bits are set. The CANnTX pin is in recessive state and the FlexCAN does not initiate any transmission or reception of CAN frames. Note that the message buffers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization, the FlexCAN must be in freeze mode (see Section 25.1.3.2, "Freeze Mode"). The following is a generic initialization sequence applicable to the FlexCAN module:

1. Initialize all operation modes in the CANCTRL*n* register.
   a) Initialize the bit timing parameters PROPSEG, PSEGS1, PSEG2, and RJW
   b) Select the S-clock rate by programming the PRESDIV field.
   c) Select the internal arbitration mode via the LBUF bit.

2. Initialize message buffers
   a) The control/status word of all message buffers must be written either as an active or inactive message buffer.
   b) All other entries in each message buffer should be initialized as required.

3. Initialize RXGMASK*n*, RX14MASK*n*, and RX15MASK*n* registers for acceptance mask as needed

4. Initialize FlexCAN interrupt handler
   a) Initialize the interrupt controller registers for any needed interrupts. See ," for more information.
   b) Set the required mask bits in the IMASK*n* register (for all message buffer interrupts) and the CANCTRL*n* (for bus off and error interrupts).

5. Clear the CANMCR*n*[HALT] bit. At this point, the FlexCAN will attempt to synchronize with the CAN bus.

25.5.1 Interrupts

There are three interrupt sources for the FlexCAN module. A combined interrupt for all 16 MBs is generated by combining all the interrupt sources from MBs. This interrupt gets generated when any of the 16 MB interrupt sources generates a interrupt. In this case, the CPU must read the IFLAG*n* register to determine which MB caused the interrupt. The other two interrupt sources (bus off and error) act in the same way, and are located in the ERRSTAT*n* register. The bus off and error interrupt mask bits are located in the CANCTRL*n* register.

Chapter 26
Debug Module

26.1 Introduction

This chapter describes the Revision B+ enhanced hardware debug module.

26.1.1 Overview

The debug module is shown in Figure 26-1.
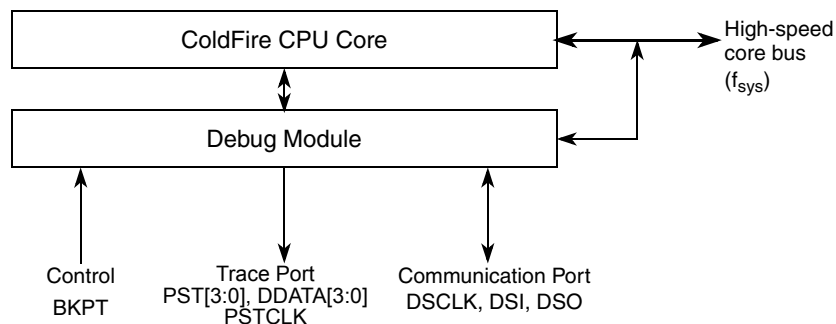
Figure 26-1. Processor/Debug Module Interface

> Depending upon the package, some devices contain only the ALLPST
> signal and do not have the PST[3:0] and DDATA[3:0] signals. ALLPST is
> a logical 'AND' of the PST[3:0] signals, and when asserted reflects that the
> core is halted.

Debug support is divided into three areas:

- Real-time trace support—The ability to determine the dynamic execution path through an application is fundamental for debugging. The ColdFire solution implements an 8-bit parallel output bus that reports processor execution status and data to an external emulator system. See Section 26.3, "Real-Time Trace Support."

- Background debug mode (BDM)—Provides low-level debugging in the ColdFire processor complex. In BDM, the processor complex is halted and a variety of commands can be sent to the processor to access memory, registers, and peripherals. The external emulator uses a three-pin, serial, full-duplex communication port. See Section 26.5, "Background Debug Mode (BDM)," and Section 26.4, "Memory Map/Register Definition."

- Real-time debug support—BDM requires the processor to be halted, which many real-time embedded applications cannot do. External development systems can access memory because the hardware supports concurrent operation of the processor and BDM-initiated commands. Debug interrupts let 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. See Section 26.6, "Real-Time Debug Support.

26.1.1.1 The New Debug Module Hardware (Rev. B+)

The revision B+ debug module features a small enhancement over revision B: the addition of three PC breakpoint registers (PCBR1–3). These new registers are mapped to DRc[3:0] addresses 0x18, 0x1A, and 0x1B. Additional PC breakpoints enable ROM/Flash software debugging. However, there are no masking registers associated with these new registers.

26.1.1.2 Enhancements over Revision A

The new debug hardware also contains the same enhancements that revision B has over revision A, while maintaining backwards compatibility with revision A. These enhancements are discussed below.

The revision B/B+ implementation has added registers that eliminate restrictions between BDM commands and the use of the hardware breakpoint logic. In some cases, the additional hardware is not program-visible; in other cases, there have been extensions to the debug module programming model.

The register containing the BDM memory address is not a program-visible resource. Rather, it is a register loaded automatically during the execution of a BDM command. In the Rev. B design, the execution of a BDM command does not affect the hardware breakpoint logic unless those registers are specifically accessed.

The other register added to the debug module programming model is the BDM address attribute register (BAAR). The BAAR is mapped to a DRc[3:0] address of 0x05. This 8-bit register is equivalent in format of the low-order byte of the AATR register (See Section 26.4.3, "BDM Address Attribute (BAAR)"). This register specifies the memory space attributes associated with all BDM memory-referencing commands.

Additionally, a bit was added to the CSR register (CSR[BKD]) that configures the debug module to assert or not assert an interrupt to the processor when the $\overline{\text{BKPT}}$ signal is asserted. The level 1 and level 2 triggers are also configurable to trigger on either an AND or an OR condition. The revision A debug module only triggers on an AND condition.

26.2 External Signal Description

Table 26-1 describes debug module signals. All ColdFire debug signals are unidirectional and related to a rising edge of the processor's clock signal. The standard 26-pin debug connector is shown in Section 26.8, "Freescale-Recommended BDM Pinout."

Table 26-1. Debug Module Signals

Signal	Description
Development Serial Clock (DSCLK)	Internally synchronized input. (The logic level on DSCLK is validated if it has the same value on two consecutive rising PSTCLK edges.) Clocks the serial communication port to the debug module during packet transfers. Maximum frequency is 1/5 the processor status clock (PSTCLK) speed. At the synchronized rising edge of DSCLK, the data input on DSI is sampled and DSO changes state.
Development Serial Input (DSI)	Internally synchronized input that provides data input for the serial communication port to the debug module.
Development Serial Output (DSO)	Provides serial output communication for debug module responses. DSO is registered internally.
Breakpoint (BKPT)	Input used to request a manual breakpoint. Assertion of BKPT puts the processor into a halted state after the current instruction completes. Halt status is reflected on processor status signals (PST[3:0]) as the value 0xF. Also, in Rev B and B+ if the CSR[BKD] bit is set, the assertion of the BKPT signal will generate a debug interrupt exception to the processor.
Debug Data (DDATA[3:0])	These output signals display the register breakpoint status as a default, or optionally, captured address and operand values. The capturing of data values is controlled by the setting of the CSR. Additionally, execution of the WDDATA instruction by the processor captures operands which are displayed on DDATA. These signals are updated each processor cycle. These signals are not implemented on packages containing fewer than 100 pins.
Processor Status (PST[3:0])	These output signals report the processor status. Table 26-2 shows the encoding of these signals. These outputs indicate the current status of the processor pipeline and, as a result, are not related to the current bus transfer. The PST value is updated each processor cycle. These signals are not implemented on packages containing fewer than 100 pins.
All Processor Status Outputs (ALLPST)	ALLPST is a logical 'AND' of the four PST signals is provided on all packages. PST[3:0] and DDATA[3:0] are not available on the low cost (less than 100 pin) packages. When asserted, reflects that the core is halted.

Figure 26-2 shows PSTCLK timing with respect to PST and DDATA.
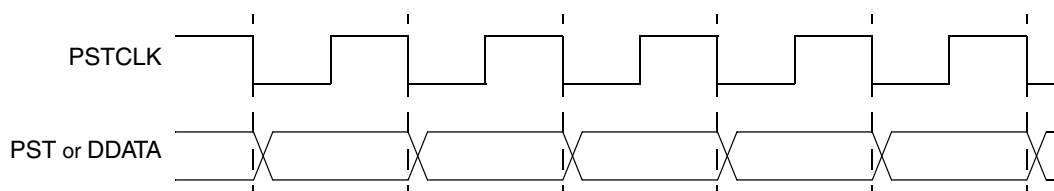
Figure 26-2. PSTCLK Timing

26.3   Real-Time Trace Support

Real-time trace, which defines the dynamic execution path, is a fundamental debug function. The ColdFire solution is to include a parallel output port providing encoded processor status and data to an external development system. This port is partitioned into two 4-bit nibbles: one nibble allows the processor to transmit processor status, (PST), and the other allows operand data to be displayed (debug data, DDATA). The processor status may not be related to the current bus transfer.

External development systems can use PST outputs with an external image of the program to completely track the dynamic execution path. This tracking is complicated by any change in flow, where branch target

address calculation is based on the contents of a program-visible register (variant addressing). DDATA outputs can be configured to display the target address of such instructions in sequential nibble increments across multiple processor clock cycles, as described in Section 26.3.1, "Begin Execution of Taken Branch (PST = 0x5)." Two 32-bit storage elements form a FIFO buffer connecting the processor's high-speed local bus to the external development system through PST[3:0] and DDATA[3:0]. The buffer captures branch target addresses and certain data values for eventual display on the DDATA port, one nibble at a time starting with the least significant bit (lsb).

Execution speeds affected only when both storage elements contain valid data to be dumped to the DDATA port. The core stalls until one FIFO entry is available.

Table 26-2 shows the encoding of these signals.

Table 26-2. Processor Status Encoding

PST[3:0]		Definition
Hex	**Binary**	
0x0	0000	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 outputs with this encoding.
0x1	0001	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.
0x2	0010	Reserved
0x3	0011	Entry into user-mode. Signaled after execution of the instruction that caused the ColdFire processor to enter user mode.
0x4	0100	Begin execution of PULSE and WDDATA instructions. PULSE defines logic analyzer triggers 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 0x4 is signaled on the PST port, followed by the appropriate marker, and then the data transfer on the DDATA port. Transfer length depends on the WDDATA operand size.
0x5	0101	Begin execution of taken branch. 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. See Section 26.3.1, "Begin Execution of Taken Branch (PST = 0x5)." Also indicates that the SYNC_PC command has been issued.
0x6	0110	Reserved
0x7	0111	Begin execution of return from exception (RTE) instruction.
0x8– 0xB	1000– 1011	Indicates the number of bytes to be displayed on the DDATA port on subsequent processor clock cycles. The value is driven onto the PST port one PSTCLK cycle before the data is displayed on DDATA. 0x8 Begin 1-byte transfer on DDATA. 0x9 Begin 2-byte transfer on DDATA. 0xA Begin 3-byte transfer on DDATA. 0xB Begin 4-byte transfer on DDATA.
0xC	1100	Exception processing. Exceptions that enter emulation mode (debug interrupt or optionally trace) generate a different encoding, as described below. Because the 0xC encoding defines a multiple-cycle mode, PST outputs are driven with 0xC until exception processing completes.

Table 26-2. Processor Status Encoding (continued)

PST[3:0]		Definition
Hex	**Binary**	
0xD	1101	Entry into emulator mode. Displayed during emulation mode (debug interrupt or optionally trace). Because this encoding defines a multiple-cycle mode, PST outputs are driven with 0xD until exception processing completes.
0xE	1110	Processor is stopped. Appears in multiple-cycle format when the processor executes a STOP instruction. The ColdFire processor remains stopped until an interrupt occurs, thus PST outputs display 0xE until the stopped mode is exited.
0xF	1111	Processor is halted. Because this encoding defines a multiple-cycle mode, the PST outputs display 0xF until the processor is restarted or reset. (see Section 26.5.1, "CPU Halt")

26.3.1 Begin Execution of Taken Branch (PST = 0x5)

PST is 0x5 when a taken branch is executed. 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, which is indicated by the PST marker value immediately preceding the DDATA nibble that begins the data output.

Bytes are displayed in least-to-most-significant order. The processor captures only those target addresses associated with taken branches which use a variant addressing mode; that is, 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 uses the debug pins to output the following sequence of information on successive processor clock cycles:

1. Use PST (0x5) to identify that a taken branch was executed.
2. Using the PST pins, optionally signal the target address to be displayed sequentially on the DDATA pins. Encodings 0x9–0xB identify the number of bytes displayed.
3. The new target address is optionally available on subsequent cycles using the DDATA port. The number of bytes of the target address displayed on this port is configurable (2, 3, or 4 bytes).

Another example of a variant branch instruction would be a JMP (A0) instruction. Figure 26-3 shows the PST and DDATA outputs that indicate a JMP (A0) execution (assuming the CSR was programmed to display the lower 2 bytes of an address).
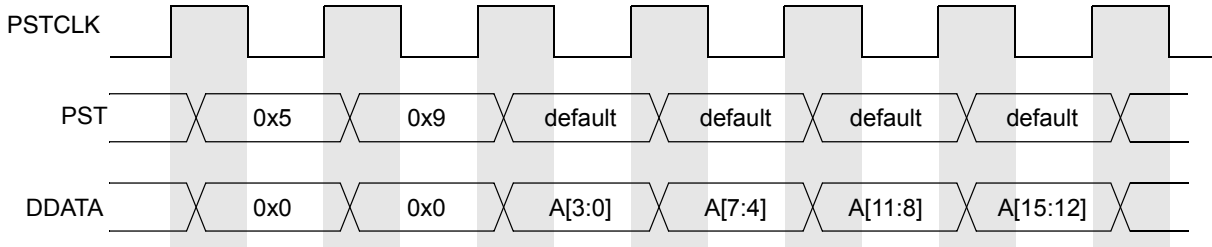
Figure 26-3. Example JMP Instruction Output on PST/DDATA

PST 0x5 indicates a taken branch and the marker value 0x9 indicates a 2-byte address. Thus, the subsequent 4 nibbles of DDATA display the lower 2 bytes of address register A0 in least-to-most-significant nibble order. The PST output after the JMP instruction completes depends on the target instruction. The PST can continue with the next instruction before the address has completely displayed on DDATA because of the DDATA FIFO. If the FIFO is full and the next instruction has captured values to display on DDATA, the pipeline stalls (PST = 0x0) until space is available in the FIFO.

26.4   Memory Map/Register Definition

In addition to the existing BDM commands that provide access to the processor's registers and the memory subsystem, the debug module contains 19 registers to support the required functionality. These registers are also accessible from the processor's supervisor programming model by executing the WDEBUG instruction (write only). Thus, the breakpoint hardware in the debug module can be written by the external development system using the debug serial interface or by the operating system running on the processor core. Software is responsible for guaranteeing that accesses to these resources are serialized and logically consistent. 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 if the ColdFire processor is using the WDEBUG instruction to access debug module registers or the resulting behavior is undefined, while DSCLK is quiescent.

These registers, shown in Table 26-3, are treated as 32-bit quantities, regardless of the number of implemented bits. These registers are also accessed through the BDM port by the commands, WDMREG and RDMREG, described in Section 26.5.3.3, "Command Set Descriptions." These commands contain a 5-bit field, DRc, that specifies the register, as shown in Table 26-3.

Table 26-3. Debug Module Memory Map

DRc[4–0]	Register	Access	Reset Value	Section/Page
0x00	Configuration/Status Register (CSR)	See Note	0x0090_0000	26.4.2/26-7
0x06	Address Attribute Trigger Register (AATR)	See Note	0x0005	26.4.4/26-10
0x07	Trigger Definition Register (TDR)	See Note	0x0000_0000	26.4.5/26-11
0x08	PC Breakpoint Register (PBR)	See Note	Undefined	26.4.6/26-13
0x09	PC Breakpoint Mask Register (PBMR)	See Note	Undefined	26.4.6/26-13
0x0C	Address High Breakpoint Register (ABHR)	See Note	Undefined	26.4.7/26-14
0x0D	Address Low Breakpoint Register (ABLR)	See Note	Undefined	26.4.7/26-14

Table 26-3. Debug Module Memory Map (continued)

DRc[4–0]	Register	Access	Reset Value	Section/Page
0x0E	Data Breakpoint Register (DBR)	See Note	Undefined	26.4.8/26-15
0x0F	Data Breakpoint Mask Register (DBMR)	See Note	Undefined	26.4.8/26-15

Note: Each debug register is accessed as a 32-bit register; reserved fields are not used (don't care).

NOTE

Debug control registers can be written by the external development system or the CPU through the WDEBUG instruction. CSR is write-only from the programming model. It can be read or written through the BDM port using the RDMREG and WDMREG commands.

26.4.1 Shared Debug Resources

The debug module implementation provides a common hardware structure for both BDM and breakpoint functionality. Certain hardware structures are used for both BDM and breakpoint purposes as shown in Table 26-4.

Table 26-4. Shared BDM/Breakpoint Hardware

Register	BDM Function	Breakpoint Function
AATR	Bus attributes for all memory commands	Attributes for address breakpoint
ABHR	Address for all memory commands	Address for address breakpoint
DBR	Data for all BDM write commands	Data for data breakpoint

Thus, loading a register to perform a specific function that shares hardware resources is destructive to the shared function. For example, if an operand address breakpoint is loaded into the debug module, a BDM command to access memory overwrites an address breakpoint in ABHR. If a data breakpoint is configured, a BDM write command overwrites the data breakpoint in DBR.

26.4.2 Configuration/Status Register (CSR)

The CSR defines the debug configuration for the processor and memory subsystem and contains status information from the breakpoint logic. CSR is write-only from the programming model. It can be read from and written to through the BDM port. CSR is accessible in supervisor mode as debug control register 0x00 using the WDEBUG instruction and through the BDM port using the RDMREG and WDMREG commands.

DRc[4:0]: 0x00 (CSR)   Access: Supervisor 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
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-4. Configuration/Status Register (CSR)

Table 26-5. CSR Field Descriptions

Field	Description
31–28 BSTAT	Breakpoint status. Provides read-only status information concerning hardware breakpoints. BSTAT is cleared by a TDR write or by a CSR read when either a level-2 breakpoint is triggered or a level-1 breakpoint is triggered and the level-2 breakpoint is disabled. 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. If FOF is set, a catastrophic halt occurred and forced entry into BDM. FOF is cleared whenever CSR is read.
26 TRG	Hardware breakpoint trigger. If TRG is set, a hardware breakpoint halted the processor core and forced entry into BDM. Reset, the debug GO command, or reading CSR will clear TRG.
25 HALT	Processor halt. If HALT is set, the processor executed a HALT and forced entry into BDM. Reset, the debug GO command, or reading CSR will clear HALT.
24 BKPT	Breakpoint assert. If BKPT is set, $\overline{\text{BKPT}}$ was asserted, forcing the processor into BDM. Reset, the debug GO command, or reading CSR will clear BKPT.
23–20 HRL	Hardware revision level. Indicates the level of debug module functionality. An emulator could use this information to identify the level of functionality supported. 1001 Revision B+ (This is the only valid value for this processor)
19–18	Reserved, should be cleared.
17 PCD	PST/DDATA Disable. Disables the PST/DDATA output signal. PSTCLK is unaffected, it remains under the control of the DISCLK bit in the SYNCR register. 0 Normal operation 1 Disables the generation of the PSTDDATA output signals, and forces these signals to remain quiescent
16 IPW	Inhibit processor writes. Setting IPW inhibits processor-initiated writes to the debug module's programming model registers. IPW can be modified only by commands from the external development system.
15 MAP	Force processor references in emulator mode. 0 All emulator-mode references are mapped into supervisor code and data spaces. 1 The processor maps all references while in emulator mode to a special address space, TT = 10, TM = 101 or 110.
14 TRC	Force emulation mode on trace exception. If TRC = 1, the processor enters emulator mode when a trace exception occurs. If TRC=0, the processor enters supervisor mode.
13 EMU	Force emulation mode. If EMU = 1, the processor begins executing in emulator mode. See Section 26.6.1.1, "Emulator Mode."
12–11 DDC	Debug data control. Controls operand data capture for DDATA, which displays the number of bytes defined by the operand reference size 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 cycles). See Table 26-2. 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. 0 HALT is a supervisor-only instruction. 1 HALT is a supervisor/user instruction.

Table 26-5. CSR Field Descriptions (continued)

Field	Description
9–8 BTB	Branch target bytes. Defines the number of bytes of branch target address DDATA displays. 00 0 bytes 01 Lower 2 bytes of the target address 10 Lower 3 bytes of the target address 11 Entire 4-byte target address See Section 26.3.1, "Begin Execution of Taken Branch (PST = 0x5)."
7	Reserved, should be cleared.
6 NPL	Non-pipelined mode. Determines whether the core operates in pipelined or mode or not. 0 Pipelined mode 1 Nonpipelined mode. The processor effectively executes one instruction at a time with no overlap. This adds at least 5 cycles to the execution time of each instruction. Given an average execution latency of 1.6 cycles/instruction, throughput in non-pipeline mode would be 6.6 cycles/instruction, approximately 25% or less 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, triggers are always reported before the next instruction begins execution and trigger reporting can be considered precise.
5 IPI	Ignore pending interrupts. 1 Core ignores any pending interrupt requests signalled while in single-instruction-step mode. 0 Core services any pending interrupt requests that were signalled while in single-step mode.
4 SSM	Single-step mode. Setting SSM puts the processor in single-step mode. 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.
3–0	Reserved, should be cleared.

26.4.3 BDM Address Attribute (BAAR)

The BAAR register defines the address space for memory-referencing BDM commands. BAAR[R, SZ] are loaded directly from the BDM command, while the low-order 5 bits can be programmed from the external development system. To maintain compatibility with the Rev. A implementation, this register is loaded any time the AATR is written. The BAAR is initialized to a value of 0x05, setting supervisor data as the default address space.

DRc[4:0]: 0x05 (BAAR)   Access: Supervisor write-only

	7	6	5	4	3	2	1	0
R								
W	R	SZ		TT		TM		
Reset:	0	0	0	0	0	1	0	1
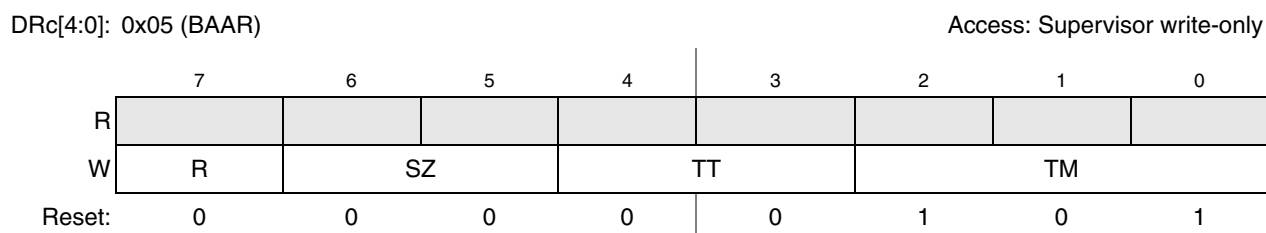
Figure 26-5. BDM Address Attribute Register (BAAR)

Table 26-6. BAAR Field Description

Field	Description
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 26.4.4, "Address Attribute Trigger Register (AATR)."
2–0 TM	Transfer modifier. See the TM definition in the AATR description, Section 26.4.4, "Address Attribute Trigger Register (AATR)."

26.4.4   Address Attribute Trigger Register (AATR)

The 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 local high-speed 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 WDMREG command.

DRc[4:0]: 0x06 (AATR)   Access: Supervisor write-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	RM	SZM		TTM		TMM			R	SZ		TT		TM		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
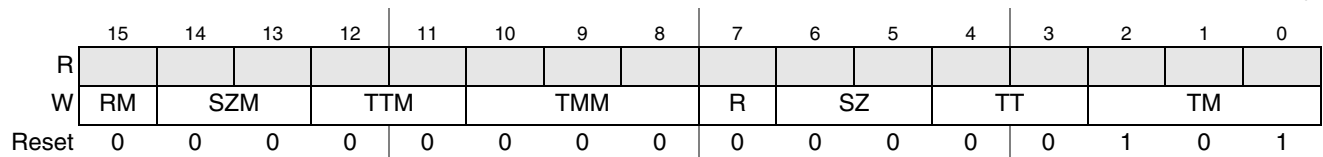
Figure 26-6. Address Attribute Trigger Register (AATR)

Table 26-7. AATR Field Descriptions

Field	Description
15 RM	Read/write mask. Setting RM masks R in address comparisons.
14–13 SZM	Size mask. Setting an SZM bit masks the corresponding SZ bit in address comparisons.
12–11 TTM	Transfer type mask. Setting a TTM bit masks the corresponding TT bit in address comparisons.
10–8 TMM	Transfer modifier mask. Setting a TMM bit 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.

Table 26-7. AATR Field Descriptions (continued)

Field	Description
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. 00 Normal processor access 01 Reserved 10 Emulator mode access 11 Acknowledge/CPU space access These bits also define the TT encoding for BDM memory commands. In this case, the 01 encoding generates an external master or DMA access (for backward compatibility). These bits are used to decode the TM bits.
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). {TABLE_BELOW}

TM	TT=00 (normal mode)	TT=10 (emulator mode)	TT=11 (acknowledge/CPU space transfers)
000	Reserved	Reserved	CPU space access
001	User data access	Reserved	Interrupt ack level 1
010	User code access	Reserved	Interrupt ack level 2
011	Reserved	Reserved	Interrupt ack level 3
100	Reserved	Reserved	Interrupt ack level 4
101	Supervisor data access	Emulator mode access	Interrupt ack level 5
110	Supervisor code access	Emulator code Access	Interrupt ack level 6
111	Reserved	Reserved	Interrupt ack level 7

26.4.5   Trigger Definition Register (TDR)

The TDR configures the operation of the hardware breakpoint logic that corresponds with the ABHR/ABLR/AATR, PBR/PBMR, and DBR/DBMR registers within the debug module. The TDR controls the actions taken under the defined conditions. Breakpoint logic may be configured as a one- or two-level trigger. TDR[31–16] define the second-level trigger and bits 15–0 define the first-level trigger.

NOTE

The debug module has no hardware interlocks, so to prevent spurious breakpoint triggers while the breakpoint registers are being loaded, disable TDR (by clearing TDR[29,13]) 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 WDMREG command.

DRc[4:0]: 0x07 (TDR)   Access: Supervisor write-only

Second Level Trigger

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	TRC		EBL	EDLW	EDWL	EDWU	EDLL	EDLM	EDUM	EDUU	DI	EAI	EAR	EAL	EPC	PCI
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	EBL	EDLW	EDWL	EDWU	EDLL	EDLM	EDUM	EDUU	DI	EAI	EAR	EAL	EPC	PCI
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-7. Trigger Definition Register (TDR)

Table 26-8. 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 always displayed on DDATA. 00 Display on DDATA only 01 Processor halt 10 Debug interrupt 11 Reserved
15 L2T	Level-2 trigger. Determines the logic operation for the trigger between the PC_condition and the (Address_range & 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) **Note:** Debug Rev A only had the 'AND' condition available for the triggers.
14 L1T	Level-1 trigger. Determines the logic operation for the trigger between the PC_condition and the (Address_range & 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) **Note:** Debug Rev A only had the 'AND' condition available for the triggers.
29 & 13 EBL	Enable breakpoint. Global enable for the breakpoint trigger. Setting TDR[EBL] enables a breakpoint trigger. Clearing it disables all breakpoints at that level.

Table 26-8. TDR Field Descriptions (continued)

Field	Description
28–22 & 12–6 EDx	Setting an EDx bit enables the corresponding data breakpoint condition based on the size and placement on the processor's local data bus. Clearing all EDx bits disables data breakpoints. <table><tr><th>Bits</th><th>Field</th><th>Description</th></tr><tr><td>28 & 12</td><td>EDLW</td><td>Data longword. Entire processor's local data bus.</td></tr><tr><td>27 & 11</td><td>EDWL</td><td>Lower data word.</td></tr><tr><td>26 & 10</td><td>EDWU</td><td>Upper data word.</td></tr><tr><td>25 & 9</td><td>EDLL</td><td>Lower lower data byte. Low-order byte of the low-order word.</td></tr><tr><td>24 & 8</td><td>EDLM</td><td>Lower middle data byte. High-order byte of the low-order word.</td></tr><tr><td>23 & 7</td><td>EDUM</td><td>Upper middle data byte. Low-order byte of the high-order word.</td></tr><tr><td>22 & 6</td><td>EDUU</td><td>Upper upper data byte. High-order byte of the high-order word.</td></tr></table>
21 & 5 DI	Data breakpoint invert. Provides a way to invert 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.
20–18 & 4–2 EAx	Enable address bits. Setting an EA bit enables the corresponding address breakpoint. Clearing all three bits disables the breakpoint. <table><tr><th>Bits</th><th>Field</th><th>Description</th></tr><tr><td>20 & 4</td><td>EAI</td><td>Enable address breakpoint inverted. Breakpoint is based outside the range between ABLR and ABHR.</td></tr><tr><td>19 & 3</td><td>EAR</td><td>Enable address breakpoint range. The breakpoint is based on the inclusive range defined by ABLR and ABHR.</td></tr><tr><td>18 & 2</td><td>EAL</td><td>Enable address breakpoint low. The breakpoint is based on the address in the ABLR.</td></tr></table>
17 & 1 EPC	Enable PC breakpoint. 0 Disable PC breakpoint 1 Enable PC breakpoint
16 & 0 PCI	Breakpoint invert. If set, this bit allows execution outside a given region as defined by PBR and PBMR to enable a trigger. If cleared, the PC breakpoint is defined within the region defined by PBR and PBMR.

26.4.6 Program Counter Breakpoint/Mask Registers (PBR, PBMR)

The PBR register defines an instruction address for use as part of the trigger. This register's contents are compared with the processor's program counter register when TDR is configured appropriately. PBR bits are masked by setting corresponding PBMR bits. Results are compared with the processor's program counter register, as defined in TDR. Figure 26-8 shows the PC breakpoint register.

The PC breakpoint register is accessible in supervisor mode using the WDEBUG instruction and through the BDM port using the RDMREG and WDMREG commands using values shown in Section 26.5.3.3, "Command Set Descriptions."

MCF5213 Reference Manual, Rev. 1.1

Preliminary

DRc[4:0]: 0x08 (PBR)   Access: Supervisor 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																	
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
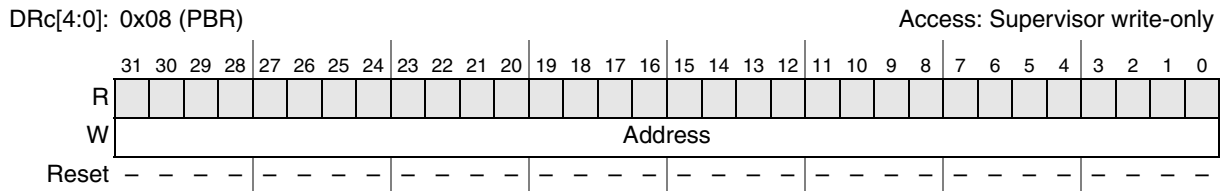
Figure 26-8. Program Counter Breakpoint Register (PBR)

Table 26-9. PBR Field Descriptions

Field	Description
31–1 Address	PC breakpoint address. The address to be compared with the PC as a breakpoint trigger.

Figure 26-8 shows PBMR. PBMR is accessible in supervisor mode as debug control register 0x09 using the WDEBUG instruction and via the BDM port using the WDMREG command.

DRc[4:0]: 0x09 (PBMR)   Access: Supervisor 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	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
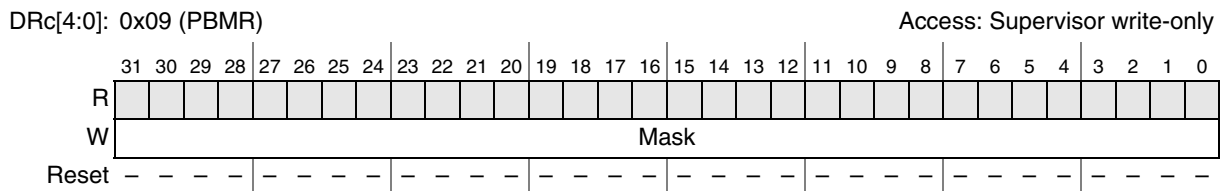
Figure 26-9. Program Counter Breakpoint Mask Register (PBMR)

Table 26-10. PBMR Field Descriptions

Field	Description
31–0 Mask	PC breakpoint mask. A zero in a bit position causes the corresponding PBR bit to be compared to the appropriate PC bit. Setting a PBMR bit causes the corresponding PBR bit to be ignored.

26.4.7 Address Breakpoint Registers (ABLR, ABHR)

The ABLR and ABHR, shown in Figure 26-10, 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:

1. Identical to the value in ABLR
2. Inside the range bound by ABLR and ABHR inclusive
3. Outside that same range

ABHR is accessible in supervisor mode as debug control register 0x0C using the WDEBUG instruction and via the BDM port using the RDMREG and WDMREG commands. ABLR is accessible in supervisor mode as debug control register 0x0D using the WDEBUG instruction and via the BDM port using the WDMREG command.

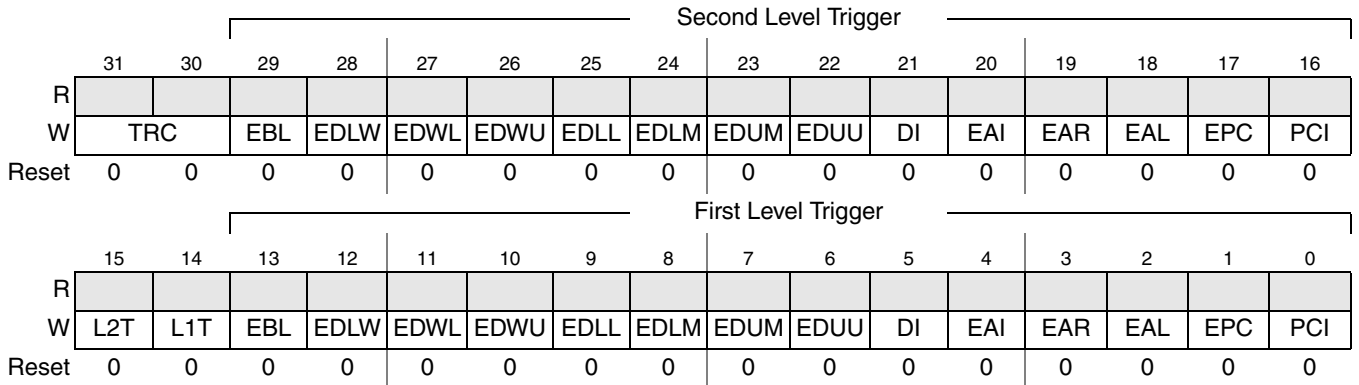DRc[4:0]: 0x0C (ABHR)   Access: Supervisor write-only
0x0D (ABLR)

	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				
Reset	– – – –	– – – –	– – – –	– – – –	– – – –	– – – –	– – – –	– – – –
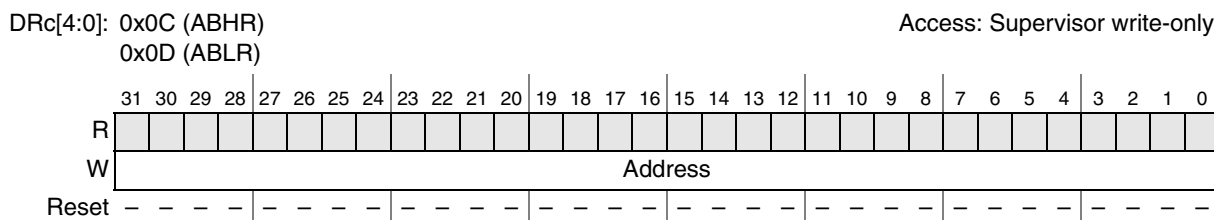
Figure 26-10. Address Breakpoint Registers (ABLR, ABHR)

Table 26-11. 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 26-12. ABHR Field Description

Field	Description
31–0 Address	High address. Holds the 32-bit address marking the upper bound of the address breakpoint range.

26.4.8 Data Breakpoint/Mask Registers (DBR, DBMR)

The DBR specifies data patterns used as part of the trigger into debug mode. DBR bits are masked by setting the corresponding DBMR bits, as defined in TDR. DBR is accessible in supervisor mode as debug control register 0x0E, using the WDEBUG instruction and through the BDM port using the RDMREG and WDMREG commands. DBMR is accessible in supervisor mode as debug control register 0x0F, using the WDEBUG instruction and via the BDM port using the WDMREG command.

DRc[4:0]: 0x0E (DBR)   Access: Supervisor write-only
0x0F (DBMR)

	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	– – – –	– – – –	– – – –	– – – –	– – – –	– – – –	– – – –	– – – –
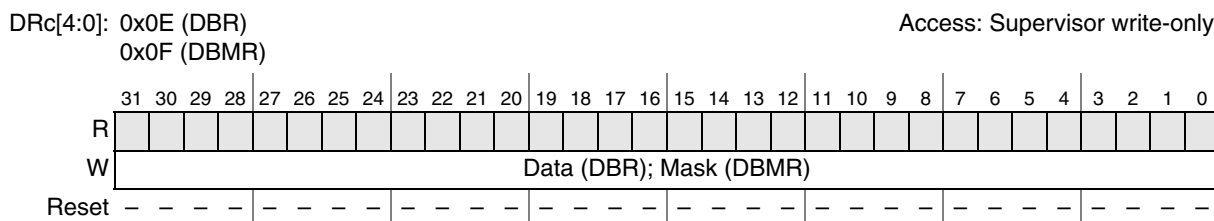
Figure 26-11. Data Breakpoint & Mask Registers (DBR & DBMR)

Table 26-13. 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 26-14. DBMR Field Descriptions

Field	Description
31–0 Mask	Data breakpoint mask. The 32-bit mask for the data breakpoint trigger. Clearing a DBR bit allows the corresponding DBR bit to be compared to the appropriate bit of the processor's local data bus. Setting a DBMR bit causes that bit to be ignored.

The DBR supports both aligned and misaligned references. Table 26-15 shows relationships between processor address, access size, and location within the 32-bit data bus.

Table 26-15. Access Size and Operand Data Location

A[1:0]	Access Size	Operand Location
00	Byte	D[31:24]
01	Byte	D[23:16]
10	Byte	D[15:8]
11	Byte	D[7:0]
0x	Word	D[31:16]
1x	Word	D[15:0]
xx	Longword	D[31:0]

26.5 Background Debug Mode (BDM)

The ColdFire Family implements a low-level system debugger in the microprocessor in a dedicated hardware module. Communication with the development system is handled through a dedicated, high-speed serial command interface. Although some BDM operations, such as CPU register accesses, require the CPU to be halted, other BDM commands, such as memory accesses, can be executed while the processor is running.

26.5.1 CPU Halt

Although most 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:

1. A catastrophic fault-on-fault condition automatically halts the processor.

2. A hardware breakpoint can be configured to generate a pending halt condition similar to the assertion of $\overline{\text{BKPT}}$. This type of halt is always first made pending in the processor. Next, 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. See Section 26.6.1, "Theory of Operation."

3. The execution of a HALT instruction immediately suspends execution. Attempting to execute HALT in user mode while CSR[UHE] = 0 generates a privilege violation exception. If CSR[UHE] = 1, HALT can be executed in user mode. After HALT executes, the processor can be restarted by serial shifting a GO command into the debug module. Execution continues at the instruction after HALT.

4. The assertion of the $\overline{\text{BKPT}}$ input is treated as a pseudo-interrupt; that is, the halt condition is postponed until the processor core samples for halts/interrupts. The processor samples for these conditions once during the execution of each instruction. If there is a pending halt condition at the sample time, the processor suspends execution and enters the halted state.

The assertion of $\overline{\text{BKPT}}$ should be considered in the following two special cases:

- After the system reset signal is negated, the processor waits for 16 processor clock cycles before beginning reset exception processing. If the $\overline{\text{BKPT}}$ input is asserted within eight cycles after $\overline{\text{RSTI}}$ is negated, the processor enters the halt state, signaling halt status (0xF) on the PST outputs. While the processor is in this state, all resources accessible through the debug module can be referenced. This is the only chance to force the processor into emulation mode through CSR[EMU].

  After system initialization, the processor's response to the GO command depends on the set of BDM commands performed while it is halted for a breakpoint. 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.

- The ColdFire architecture also handles a special case of $\overline{\text{BKPT}}$ being asserted while the processor is stopped by execution of the STOP instruction. For this case, 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, that is, the instruction following the STOP opcode.

The CSR[27–24] bits indicate the halt source, showing the highest priority source for multiple halt conditions.

26.5.2 BDM Serial Interface

When the CPU is halted and PST reflects the halt status, the development system can send unrestricted commands to the debug module. The debug module implements a synchronous serial protocol using two inputs (DSCLK and DSI) and one output (DSO), where DSO is specified as a delay relative to the rising edge of the processor clock. See Table 26-1. The development system serves as the serial communication channel master and must generate DSCLK.

The serial channel operates at a frequency from DC to 1/5 of the PSTCLK frequency. The channel uses full-duplex mode, where data is sent and received simultaneously by both master and slave devices. The transmission consists of 17-bit packets composed of a status/control bit and a 16-bit data word. As shown in Figure 26-12, all state transitions are enabled on a rising edge of PSTCLK when DSCLK is high; that is, DSI is sampled and DSO is driven.

Preliminary
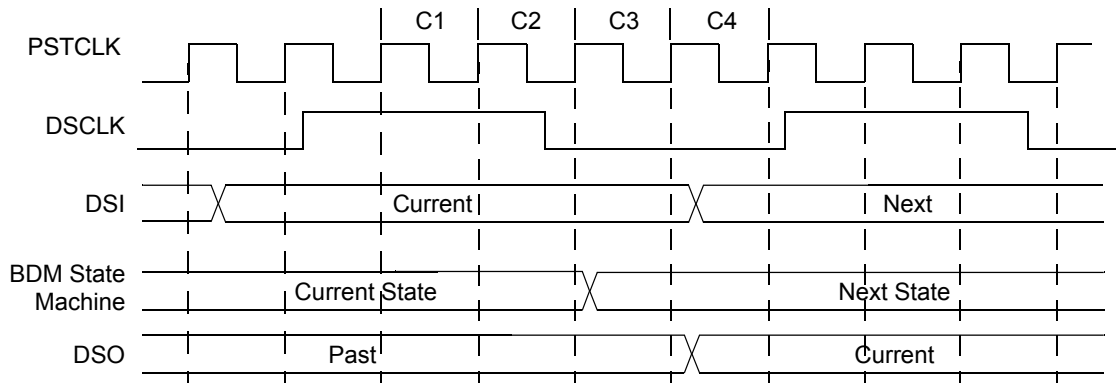
Figure 26-12. BDM Serial Interface Timing

DSCLK and DSI are synchronized inputs. DSCLK acts as a pseudo clock enable and is sampled on the rising edge of the processor clock as well as the DSI. DSO is delayed from the DSCLK-enabled CLK rising edge (registered after a BDM state machine state change). All events in the debug module's serial state machine are based on the processor clock rising edge. DSCLK must also be sampled low (on a positive edge of CLK) between each bit exchange. The msb is transferred first. Because DSO changes state based on an internally-recognized rising edge of DSCLK, DSO cannot be used to indicate the start of a serial transfer. The development system must count clock cycles in a given transfer. C1–C4 are described as follows:

- C1—First synchronization cycle for DSI (DSCLK is high).
- C2—Second synchronization cycle for DSI (DSCLK is high).
- C3—BDM state machine changes state depending upon DSI and whether the entire input data transfer has been transmitted.
- C4—DSO changes to next value.

NOTE

A not-ready response can be ignored except during a memory-referencing cycle. Otherwise, the debug module can accept a new serial transfer after 32 processor clock periods.

26.5.2.1   Receive Packet Format

The basic receive packet, Figure 26-13, consists of 16 data bits and 1 status bit.

15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	Data Field															
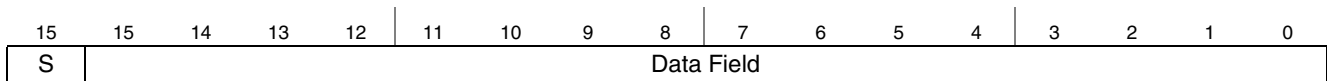
Figure 26-13. Receive BDM Packet

Table 26-16. Receive BDM Packet Field Description

Field	Description
16 S	Status. Indicates the status of CPU-generated messages listed below. The not-ready response can be ignored unless a memory-referencing cycle is in progress. Otherwise, the debug module can accept a new serial transfer after 32 processor clock periods. <table><tr><th>S</th><th>Data</th><th>Message</th></tr><tr><td>0</td><td>xxxx</td><td>Valid data transfer</td></tr><tr><td>0</td><td>FFFF</td><td>Status OK</td></tr><tr><td>1</td><td>0000</td><td>Not ready with response; come again</td></tr><tr><td>1</td><td>0001</td><td>Error–Terminated bus cycle; data invalid</td></tr><tr><td>1</td><td>FFFF</td><td>Illegal Command</td></tr></table>
15–0 Data	Data. Contains the message to be sent from the debug module to the development system. The response message is always a single word, with the data field encoded as shown above.

26.5.2.2 Transmit Packet Format

The basic transmit packet, Figure 26-14, consists of 16 data bits and 1 reserve bit.

15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	Data															
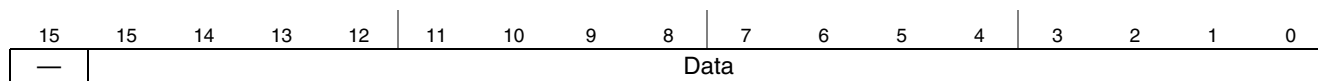
Figure 26-14. Transmit BDM Packet

Table 26-17. Transmit BDM Packet Field Description

Field	Description
16	Reserved, should be cleared.
15–0 Data	Data bits 15–0. Contains the data to be sent from the development system to the debug module.

26.5.3 BDM Command Set

Table 26-18 summarizes the BDM command set. Subsequent paragraphs contain detailed descriptions of each command. Issuing a BDM command when the processor is accessing debug module registers using the WDEBUG instruction causes undefined behavior.

Table 26-18. BDM Command Summary

Command	Mnemonic	Description	CPU State[1]	Section	Command (Hex)
Read A/D register	RAREG/ RDREG	Read the selected address or data register and return the results through the serial interface.	Halted	26.5.3.3.1	0x218 {A/D, Reg[2:0]}
Write A/D register	WAREG/ WDREG	Write the data operand to the specified address or data register.	Halted	26.5.3.3.2	0x208 {A/D, Reg[2:0]}

Table 26-18. BDM Command Summary (continued)

Command	Mnemonic	Description	CPU State[1]	Section	Command (Hex)
Read memory location	READ	Read the data at the memory location specified by the longword address.	Steal	26.5.3.3.3	0x1900—byte 0x1940—word 0x1980—lword[2]
Write memory location	WRITE	Write the operand data to the memory location specified by the longword address.	Steal	26.5.3.3.4	0x1800—byte 0x1840—word 0x1880—lword[2]
Dump memory block	DUMP	Used with READ to dump large blocks of memory. An initial READ is executed to set up the starting address of the block and to retrieve the first result. A DUMP command retrieves subsequent operands.	Steal	26.5.3.3.5	0x1D00—byte 0x1D40—word 0x1D80—lword[2]
Fill memory block	FILL	Used with WRITE to fill large blocks of memory. An initial WRITE is executed to set up the starting address of the block and to supply the first operand. A FILL command writes subsequent operands.	Steal	26.5.3.3.6	0x1C00—byte 0x1C40—word 0x1C80—lword[2]
Resume execution	GO	The pipeline is flushed and refilled before resuming instruction execution at the current PC.	Halted	26.5.3.3.7	0x0C00
No operation	NOP	Perform no operation; may be used as a null command.	Parallel	26.5.3.3.8	0x0000
Synchronize PC to PST/DDATA	SYNC_PC	Capture the current PC and display it on the PST/DDATA outputs	Parallel	26.5.3.3.9	0x0001
Read control register	RCREG	Read the system control register.	Halted	26.5.3.3.10	0x2980
Write control register	WCREG	Write the operand data to the system control register.	Halted	26.5.3.3.11	0x2880
Read debug module register	RDMREG	Read the debug module register.	Parallel	26.5.3.3.12	0x2D {0x4[3] DRc[4:0]}
Write debug module register	WDMREG	Write the operand data to the debug module register.	Parallel	26.5.3.3.13	0x2C {0x4[3] DRc[4:0]}

[1] General command effect and/or requirements on CPU operation:
   - Halted. The CPU must be halted to perform this command.
   - Steal. Command generates bus cycles that can be interleaved with bus accesses.
   - Parallel. Command is executed in parallel with CPU activity.
[2] lword = longword
[3] 0x4 is a three-bit field.

Unassigned command opcodes are reserved by Freescale. All unused command formats within any revision level perform a NOP and return the illegal command response.

26.5.3.1 ColdFire BDM Command Format

All ColdFire Family BDM commands include a 16-bit operation word followed by an optional set of one or more extension words, as shown in Figure 26-15.

15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation							0	R/W	Op Size		0	0	A/D	Register		
Extension Word(s)																
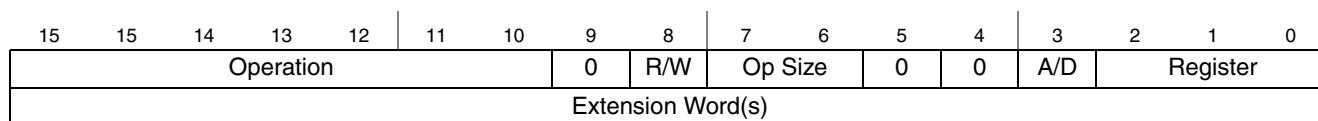
Figure 26-15. BDM Command Format

Table 26-19. BDM Field Descriptions

Field	Description
15–10 Operation	Specifies the command. These values are listed in Table 26-18.
9	Reserved, should be cleared.
8 R/W	Direction of operand transfer. 0 Data is written to the CPU or to memory from the development system. 1 The transfer is from the CPU to the development system.
7–6 Op Size	Operand data size for sized operations. Addresses are expressed as 32-bit absolute values. Note that a command performing a byte-sized memory read leaves the upper 8 bits of the response data undefined. Referenced data is returned in the lower 8 bits of the response. <table><tr><th></th><th>Operand Size</th><th>Bit Values</th></tr><tr><td>00</td><td>Byte</td><td>8 bits</td></tr><tr><td>01</td><td>Word</td><td>16 bits</td></tr><tr><td>10</td><td>Longword</td><td>32 bits</td></tr><tr><td>11</td><td>Reserved</td><td>—</td></tr></table>
5–4	Reserved, should be cleared.
3 A/D	Address/data. Determines whether the register field specifies a data or address register. 0 Indicates a data register. 1 Indicates an address register.
2–0 Register	Contains the register number in commands that operate on processor registers.

26.5.3.1.1 Extension Words as Required

Some commands require extension words for addresses and/or immediate data. Addresses require two extension words because only absolute long addressing is permitted. Longword accesses are forcibly longword-aligned and word accesses are forcibly word-aligned. Immediate data can be 1 or 2 words long. Byte and word data each requires a single extension word and longword data requires two extension words.

Operands and addresses are transferred most-significant word first. In the following descriptions of the BDM command set, the optional set of extension words is defined as address, data, or operand data.

26.5.3.2   Command Sequence Diagrams

The command sequence diagram in Figure 26-16 shows serial bus traffic for commands. Each bubble represents a 17-bit bus transfer. The top half of each bubble indicates the data the development system sends to the debug module; the bottom half indicates the debug module's response to the previous development system commands. Command and result transactions overlap to minimize latency.
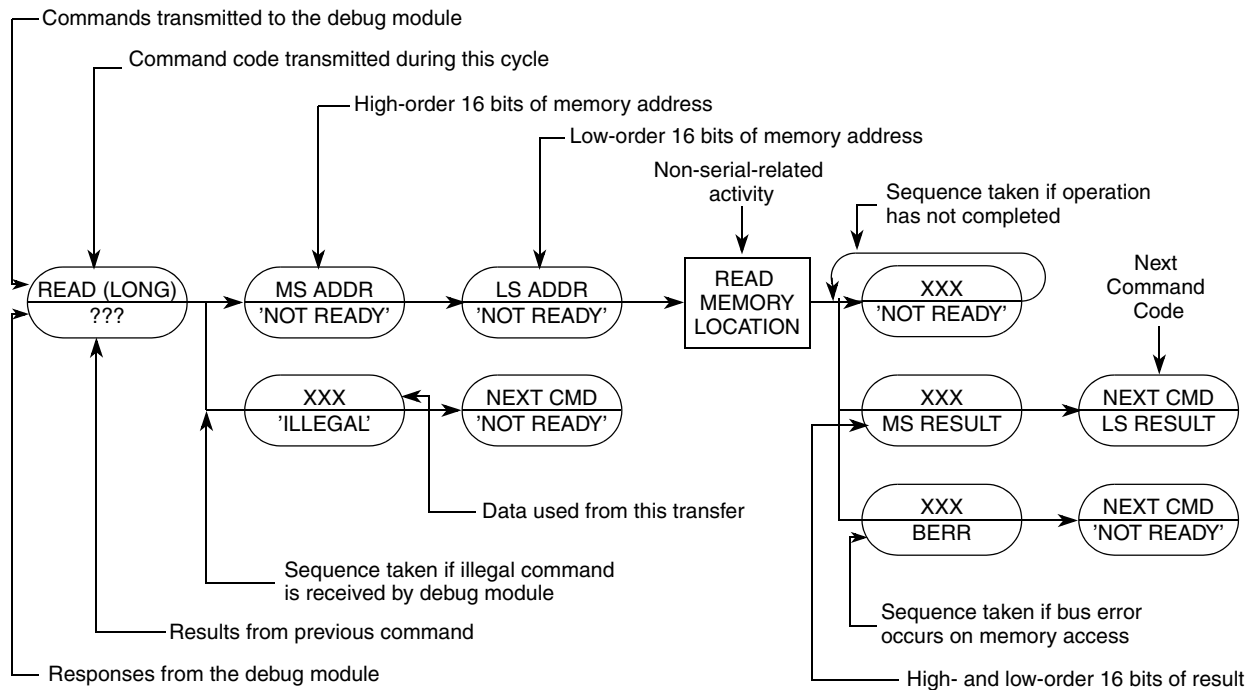
Figure 26-16. Command Sequence Diagram

The sequence is as follows:

- In cycle 1, the development system command is issued (READ in this example). The debug module responds with either the low-order results of the previous command or a command complete status of the previous command, if no results are required.

- In cycle 2, the development system supplies the high-order 16 address bits. The debug module returns a not-ready response unless the received command is decoded as unimplemented, which is indicated by the illegal command encoding. If this occurs, the development system should retransmit the command.

NOTE

A not-ready response can be ignored except during a memory-referencing cycle. Otherwise, the debug module can accept a new serial transfer after 32 processor clock periods.

- In cycle 3, the development system supplies the low-order 16 address bits. The debug module always returns a not-ready response.

- At the completion of cycle 3, the debug module initiates a memory read operation. Any serial transfers that begin during a memory access return a not-ready response.

- Results are returned in the two serial transfer cycles after the memory access completes. For any command performing a byte-sized memory read operation, the upper 8 bits of the response data are undefined and the referenced data is returned in the lower 8 bits. The next command's opcode is sent to the debug module during the final transfer. If a memory or register access is terminated with a bus error, the error status (S = 1, DATA = 0x0001) is returned instead of result data.

26.5.3.3   Command Set Descriptions

The following sections describe the commands summarized in Table 26-18.

NOTE

The BDM status bit (S) is 0 for normally completed commands; S = 1 for illegal commands, not-ready responses, and transfers with bus-errors. Section 26.5.2, "BDM Serial Interface," describes the receive packet format.

Freescale reserves unassigned command opcodes for future expansion. Unused command formats in any revision level perform a NOP and return an illegal command response.

26.5.3.3.1   Read A/D Register (RAREG/RDREG)

Read the selected address or data register and return the 32-bit result. A bus error response is returned if the CPU core is not halted.

Command/Result Formats:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Command		0x2				0x1				0x8			A/D		Register	
Result							D[31:16]									
							D[15:0]									

Figure 26-17. RAREG/RDREG Command Format
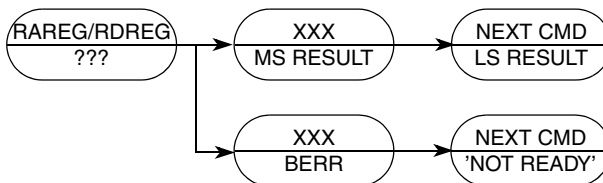
Command Sequence:

Figure 26-18. RAREG/RDREG Command Sequence

Operand Data: None

Result Data: The contents of the selected register are returned as a longword value, most-significant word first.

26.5.3.3.2 Write A/D Register (WAREG/WDREG)

The operand longword data is written to the specified address or data register. A write alters all 32 register bits. A bus error response is returned if the CPU core is not halted.

Command Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2				0x0				0x8				A/D	Register		
D[31:16]															
D[15:0]															
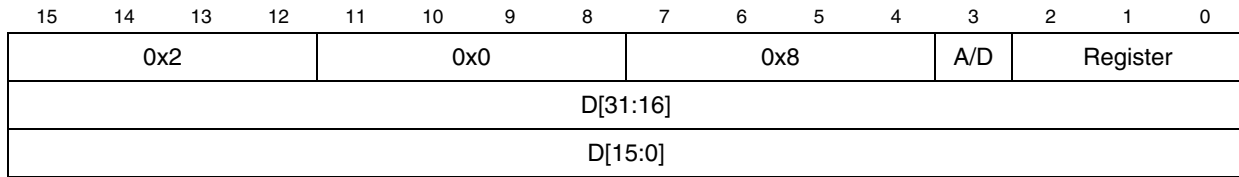
Figure 26-19. WAREG/WDREG Command Format
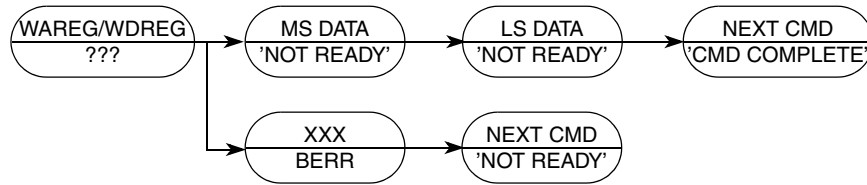
Command Sequence:

Figure 26-20. WAREG/WDREG Command Sequence

Operand Data:	Longword data is written into the specified address or data register. The data is supplied most-significant word first.
Result Data:	Command complete status is indicated by returning 0xFFFF (with S cleared) when the register write is complete.

26.5.3.3.3 Read Memory Location (READ)

Read data at the longword address. Address space is defined by BAAR[TT,TM]. Hardware forces low-order address bits to zeros for word and longword accesses to ensure that word addresses are word-aligned and longword addresses are longword-aligned.

Command/Result Formats:

		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte	Command		0x1				0x9				0x0				0x0		
			A[31:16]														
			A[15:0]														
	Result	X	X	X	X	X	X	X	X		D[7:0]						
Word	Command		0x1				0x9				0x4				0x0		
			A[31:16]														
			A[15:0]														
	Result		D[15:0]														
Longword	Command		0x1				0x9				0x8				0x0		
			A[31:16]														
			A[15:0]														
	Result		D[31:16]														
			D[15:0]														

Figure 26-21. READ Command/Result Formats
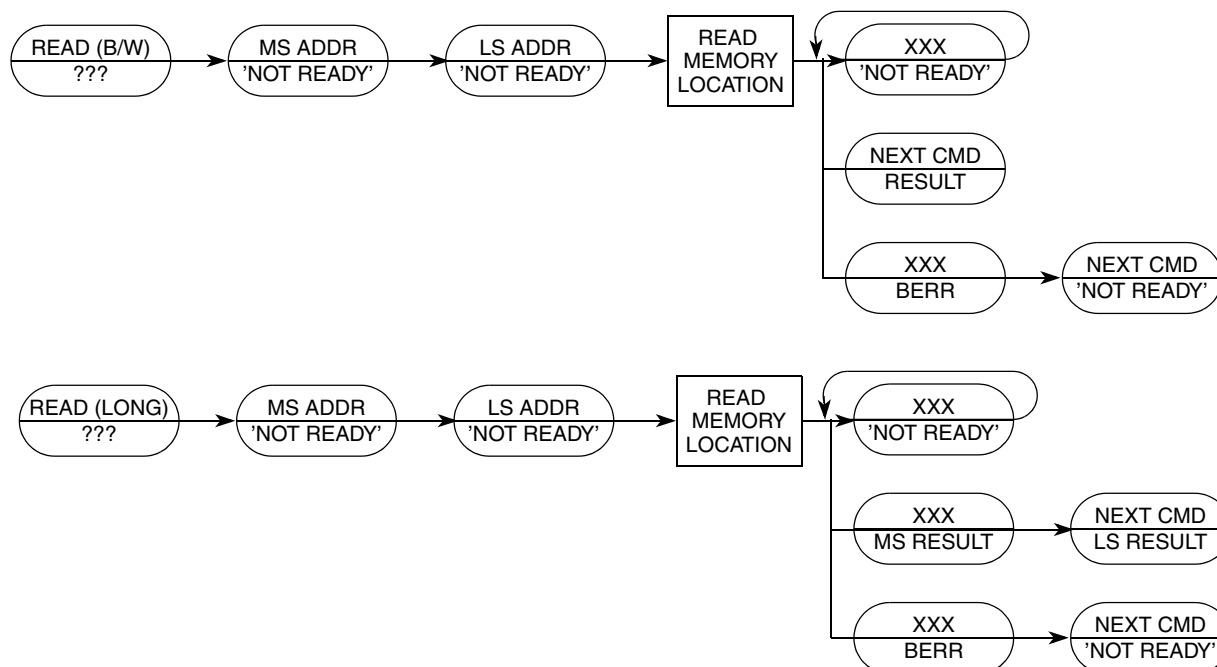
Command Sequence:

Figure 26-22. READ Command Sequence

Operand Data: The only operand is the longword address of the requested location.

Result Data: Word results return 16 bits of data; longword results return 32. Bytes are returned in the LSB of a word result; the upper byte is undefined. 0x0001 (S = 1) is returned if a bus error occurs.

26.5.3.3.4   Write Memory Location (WRITE)

Write data to the memory location specified by the longword address. The address space is defined by BAAR[TT,TM]. Hardware forces low-order address bits to zeros for word and longword accesses to ensure that word addresses are word-aligned and longword addresses are longword-aligned.

Command Formats:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte	0x1				0x8				0x0				0x0			
	A[31:16]															
	A[15:0]															
	X	X	X	X	X	X	X	X	D[7:0]							
Word	0x1				0x8				0x4				0x0			
	A[31:16]															
	A[15:0]															
	D[15:0]															
Longword	0x1				0x8				0x8				0x0			
	A[31:16]															
	A[15:0]															
	D[31:16]															
	D[15:0]															

Figure 26-23. WRITE Command Format

Command Sequence:
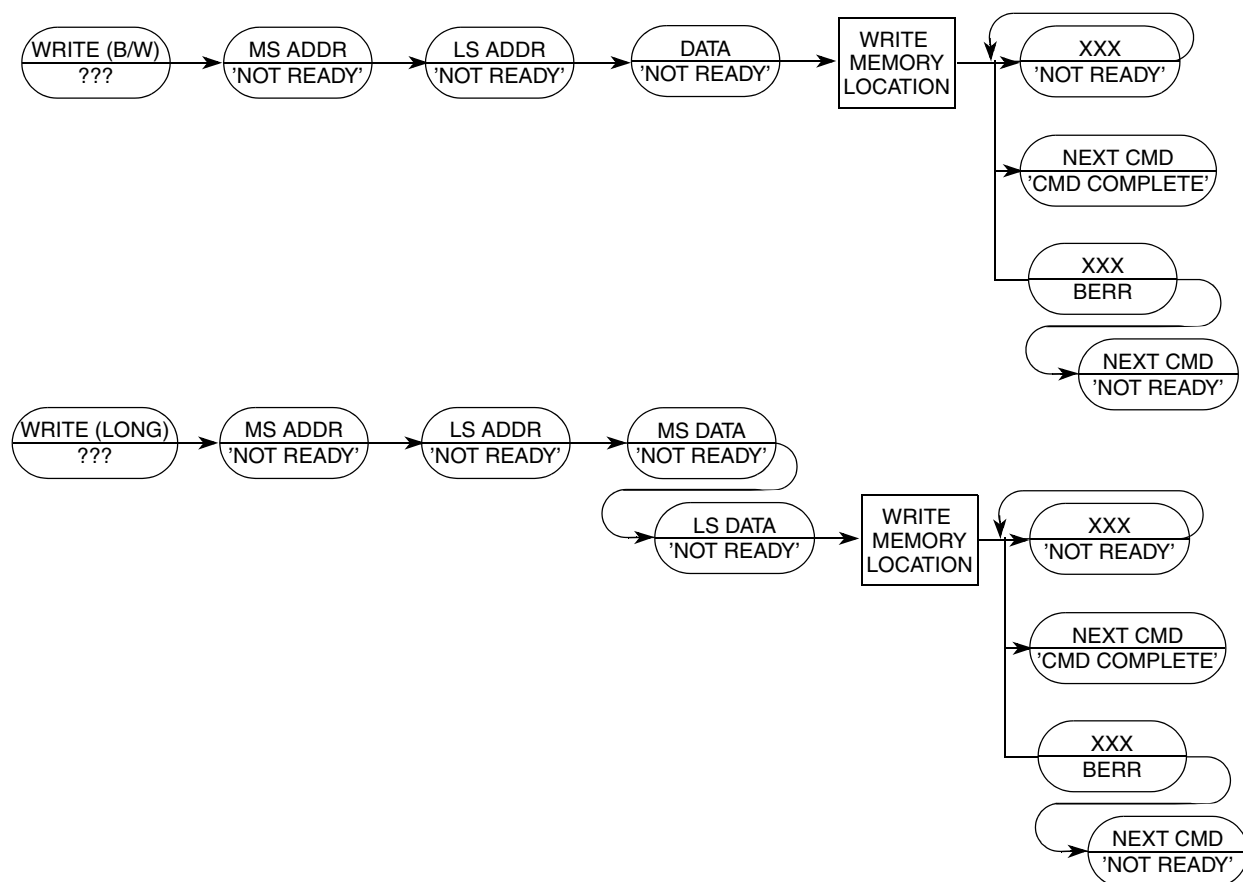
Figure 26-24. WRITE Command Sequence

Operand Data:	This two-operand instruction requires a longword absolute address that specifies a location to which the data operand is to be written. Byte data is sent as a 16-bit word, justified in the LSB; 16- and 32-bit operands are sent as 16 and 32 bits, respectively.
Result Data:	Command complete status is indicated by returning 0xFFFF (with S cleared) when the register write is complete. A value of 0x0001 (with S set) is returned if a bus error occurs.

26.5.3.3.5 Dump Memory Block (DUMP)

DUMP is used with the READ command to access large blocks of memory. An initial READ is executed to set up the starting address of the block and to retrieve the first result. If an initial READ is not executed before the first DUMP, an illegal command response is returned. The DUMP 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 commands use this address, perform the memory read, increment it by the current operand size, and store the updated address in the temporary register.

NOTE

DUMP does not check for a valid address; it is a valid command only when preceded by NOP, READ, or another DUMP command. Otherwise, an illegal command response is returned. NOP can be used for intercommand padding without corrupting the address pointer.

The size field is examined each time a DUMP command is processed, allowing the operand size to be dynamically altered.

Command/Result Formats:

		15			12	11			8	7			4	3			0
Byte	Command		0x1				0xD				0x0				0x0		
	Result	X	X	X	X	X	X	X	X				D[7:0]				
Word	Command		0x1				0xD				0x4				0x0		
	Result								D[15:0]								
Longword	Command		0x1				0xD				0x8				0x0		
	Result								D[31:16]								
									D[15:0]								

Figure 26-25. DUMP Command/Result Formats
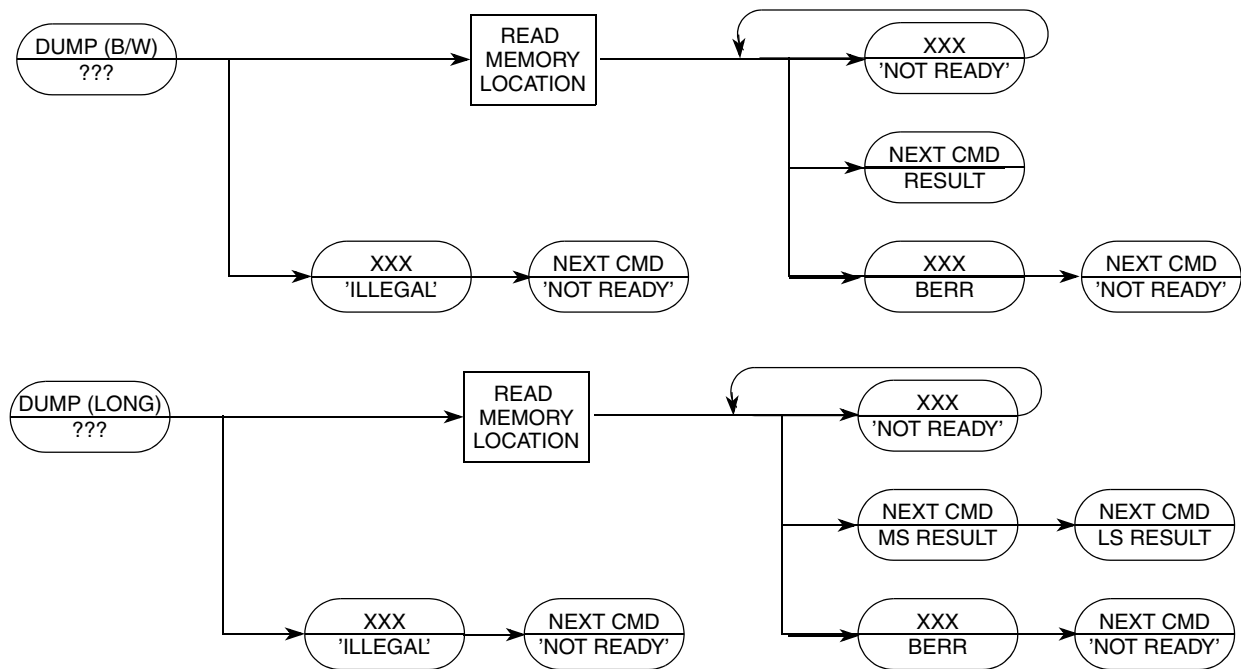
Command Sequence:

Figure 26-26. DUMP Command Sequence

Operand Data: None

Result Data: Requested data is returned as either a word or longword. Byte data is returned in the least-significant byte of a word result. Word results return 16 bits of significant data; longword results return 32 bits. A value of 0x0001 (with S set) is returned if a bus error occurs.

26.5.3.3.6   Fill Memory Block (FILL)

A FILL command is used with the WRITE command to access large blocks of memory. An initial WRITE is executed to set up the starting address of the block and to supply the first operand. The FILL command writes subsequent operands. The initial address is incremented by the operand size (1, 2, or 4) and saved in a temporary register after the memory write. Subsequent FILL commands use this address, perform the write, increment it by the current operand size, and store the updated address in the temporary register.

If an initial WRITE is not executed preceding the first FILL command, the illegal command response is returned.

NOTE

The FILL command does not check for a valid address—FILL is a valid command only when preceded by another FILL, a NOP, or a WRITE command. Otherwise, an illegal command response is returned. The NOP command can be used for intercommand padding without corrupting the address pointer.

The size field is examined each time a FILL command is processed, allowing the operand size to be altered dynamically.

Command Formats:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte	0x1				0xC				0x0				0x0			
	X	X	X	X	X	X	X	X	D[7:0]							
Word	0x1				0xC				0x4				0x0			
	D[15:0]															
Longword	0x1				0xC				0x8				0x0			
	D[31:16]															
	D[15:0]															

Figure 26-27. FILL Command Format

Command Sequence:
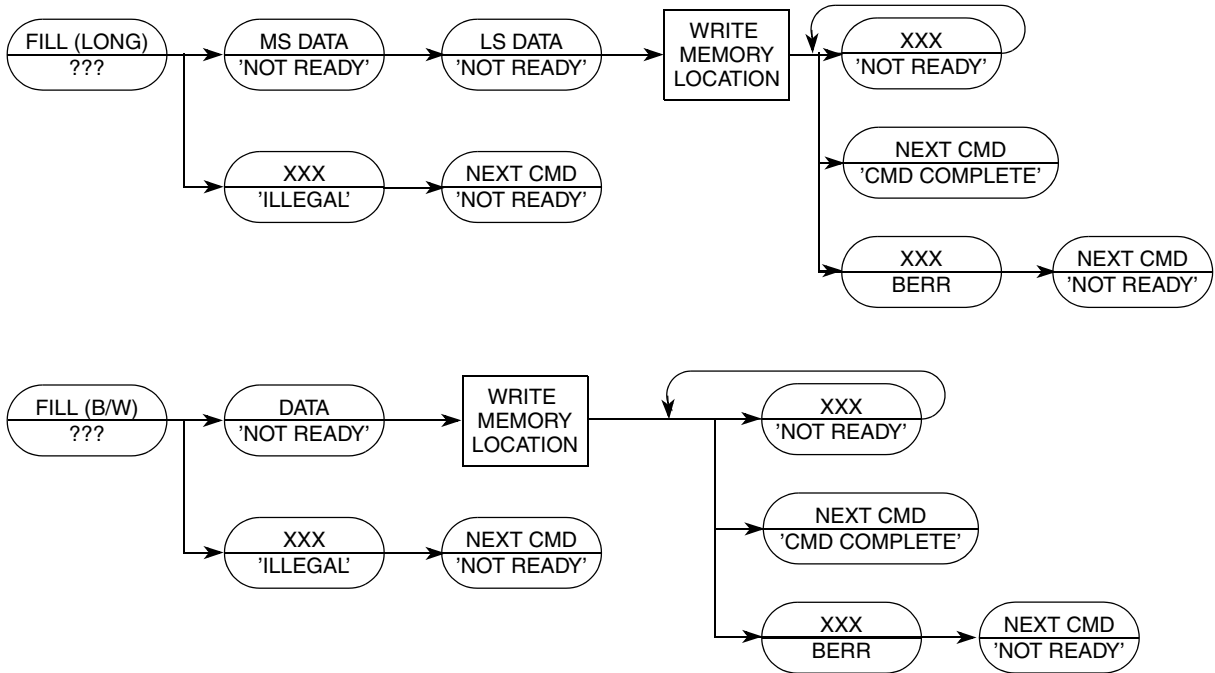
Figure 26-28. FILL Command Sequence

Operand Data: A single operand is data to be written to the memory location. Byte data is sent as a 16-bit word, justified in the least-significant byte; 16- and 32-bit operands are sent as 16 and 32 bits, respectively.

Result Data: Command complete status (0xFFFF) is returned when the register write is complete. A value of 0x0001 (with S set) is returned if a bus error occurs.

26.5.3.3.7   Resume Execution (GO)

The 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.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0				0xC				0x0				0x0			
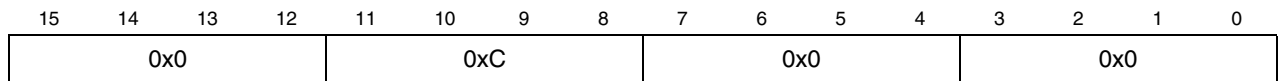
Figure 26-29. GO Command Format

Command Sequence:
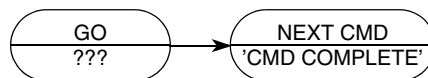
Figure 26-30. GO Command Sequence

Operand Data: None

Result Data: The command-complete response (0xFFFF) is returned during the next shift operation.

26.5.3.3.8   No Operation (NOP)

NOP performs no operation and may be used as a null command where required.

Command Formats:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0x0				0x0				0x0				0x0		
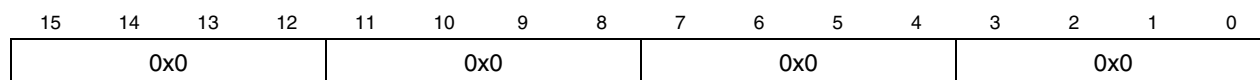
Figure 26-31. NOP Command Format

Command Sequence:
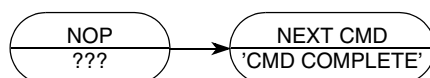
Figure 26-32. NOP Command Sequence

Operand Data: None

Result Data: The command-complete response, 0xFFFF (with S cleared), is returned during the next shift operation.

26.5.3.3.9   Synchronize PC to the PST/DDATA Lines (SYNC_PC)

Capture the current PC (program counter) and display it on the PST/DDATA outputs. After the debug module receives the command, it sends a signal to the ColdFire core that the current PC must be displayed. The core then forces an instruction fetch at the next PC with the address being captured in the DDATA logic under control of the CSR[BTB] bits. The specific sequence of PST and DDATA values is defined below:

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, generates a PST = $5 value indicating a taken branch and signals DDATA. DDATA captures the instruction address corresponding to the PC. DDATA generates a PST marker ($9–$B) as defined by CSR[BTB] and displays the captured PC address.

This command can be used to dynamically access the PC for performance monitoring. The execution of this command is considerably less obtrusive to the real-time operation of an application than a halt-CPU/read-PC/resume command sequence.

Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0x0				0x0				0x0				0x1		
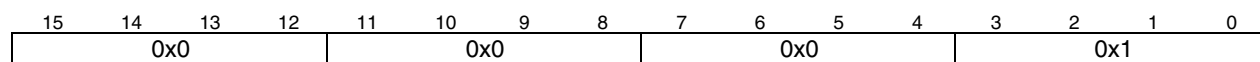
Command Sequence:
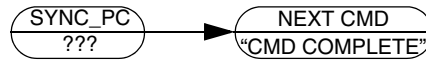
Operand Data:	None
Result Data:	The command complete response, $FFFF (with the status bit cleared), is returned during the next shift operation.

26.5.3.3.10 Read Control Register (RCREG)

Reads the selected control register and returns the 32-bit result. Accesses to the processor/memory control registers are always 32 bits wide, regardless of register width. The second and third words of the command form a 32-bit address, which the debug module uses to generate a special bus cycle to access the specified control register. The 12-bit Rc field is the same as that used by the MOVEC instruction.

Command/Result Formats:

	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Command	0x2	0x9	0x8	0x0
	0x0	0x0	0x0	0x0
	0x0	Rc		
Result	D[31:16]			
	D[15:0]			

Figure 26-33. RCREG **Command/Result Formats**

Rc encoding:

Table 26-20. Control Register Map

Rc	Register Definition
0x004	Access Control Register (ACR0)
0x005	Access Control Register (ACR1)
0x800	Other Stack Pointer (OTHER_A7)
0x801	Vector Base Register (VBR)
0x804	MAC Status Register (MACSR)
0x805	MAC Mask Register (MASK)
0x806	MAC Accumulator 0 (ACC0)
0x80E	Status Register (SR)
0x80F	Program Register (PC)
0xC04	Flash Base Address Register (FLASHBAR)
0xC05	RAM Base Address Register (RAMBAR)

Command Sequence:
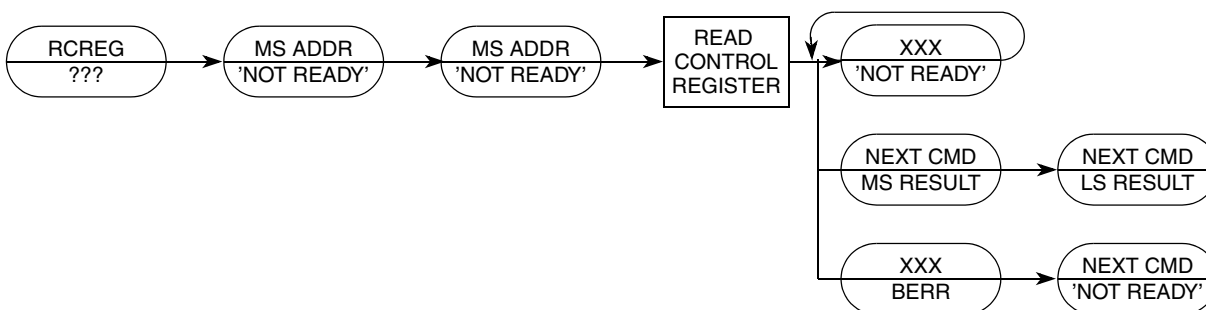
Figure 26-34. RCREG Command Sequence

Operand Data: The only operand is the 32-bit Rc control register select field.

Result Data: Control register contents are returned as a longword, most-significant word first.
                       The implemented portion of registers smaller than 32 bits is guaranteed correct;
                       other bits are undefined.

BDM Accesses of the Stack Pointer Registers (A7: SSP, USP)

The V2 core supports two unique stack pointer (A7) registers: the supervisor stack pointer (SSP) and the user stack pointer (USP). The hardware implementation of these two programmable-visible 32-bit registers does not uniquely identify one as the SSP and the other as the USP. Rather, the hardware uses one 32-bit register as the currently-active A7 and the other register is named simply the "other_A7". Thus, the contents of the two hardware registers is a function of the operating mode of the processor:

```
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 reads and writes to the A7 and other_A7 registers directly. It is the responsibility of the external development system to determine the mapping of the two hardware registers (A7, other_A7) to the two program-visible definitions (supervisor and user stack pointers), based on the supervisor bit of the status register.

26.5.3.3.11   Write Control Register (WCREG)

The operand (longword) data is written to the specified control register. The write alters all 32 register bits.

Command/Result Formats:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Command	0x2				0x8				0x8				0x0			
	0x0				0x0				0x0				0x0			
	0x0				Rc											
Result	D[31:16]															
	D[15:0]															

Figure 26-35. WCREG Command/Result Formats
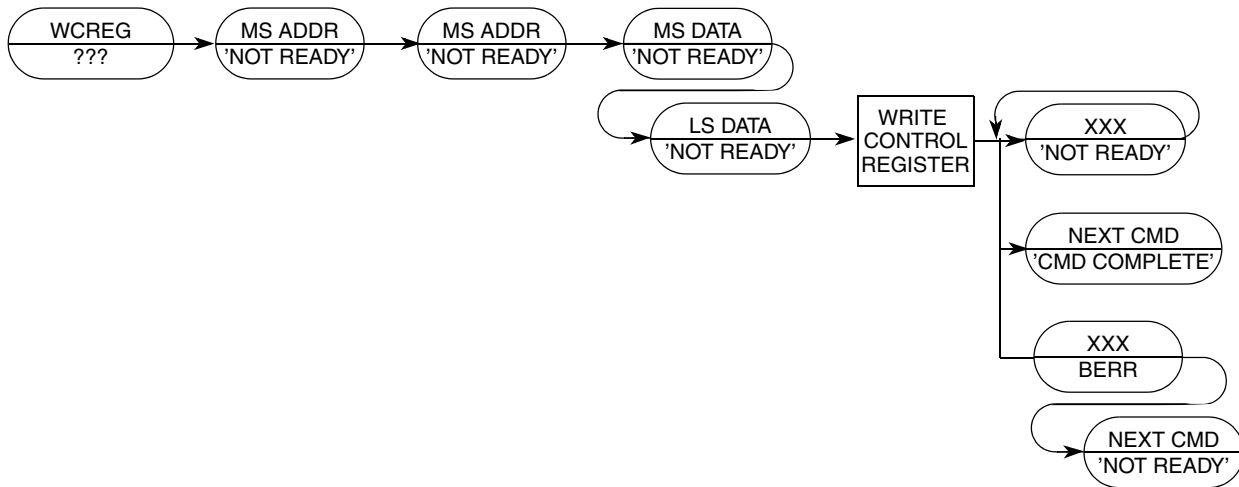
Command Sequence:

Figure 26-36. WCREG Command Sequence

Operand Data: This instruction requires two longword operands. The first selects the register to which the operand data is to be written; the second contains the data.

Result Data: Successful write operations return 0xFFFF. Bus errors on the write cycle are indicated by the setting of bit 16 in the status message and by a data pattern of 0x0001.

26.5.3.3.12   Read Debug Module Register (RDMREG)

Read the selected debug module register and return the 32-bit result. The only valid register selection for the RDMREG command is CSR (DRc = 0x00). Note that this read of the CSR clears CSR[FOF, TRG, HALT, BKPT]; as well as the trigger status bits (CSR[BSTAT]) if either a level-2 breakpoint has been triggered or a level-1 breakpoint has been triggered and no level-2 breakpoint has been enabled.

Command/Result Formats:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Command		0x2				0xD				100				DRc		
Result						D[31:16]										
						D[15:0]										

Figure 26-37. RDMREG Command/Result Formats

Table 26-21 shows the definition of DRc encoding.

Table 26-21. Definition of DRc Encoding—Read

DRc[4:0]	Debug Register Definition	Mnemonic	Initial State	Page
0x00	Configuration/Status	CSR	0x0	p. 26-7
0x01–0x1F	Reserved	—	—	—

Command Sequence:
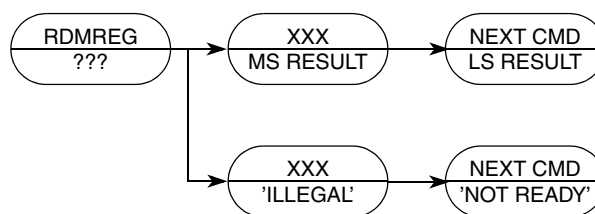
Figure 26-38. RDMREG Command Sequence

Operand Data: None

Result Data: The contents of the selected debug register are returned as a longword value. The data is returned most-significant word first.

26.5.3.3.13   Write Debug Module Register (WDMREG)

The operand (longword) data is written to the specified debug module register. All 32 bits of the register are altered by the write. DSCLK must be inactive while the debug module register writes from the CPU accesses are performed using the WDEBUG instruction.

Command Format:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		0x2				0xC				100				DRc		
						D[31:16]										
						D[15:0]										

Figure 26-39. WDMREG BDM Command Format

Table 26-3 shows the definition of the DRc write encoding.

Command Sequence:
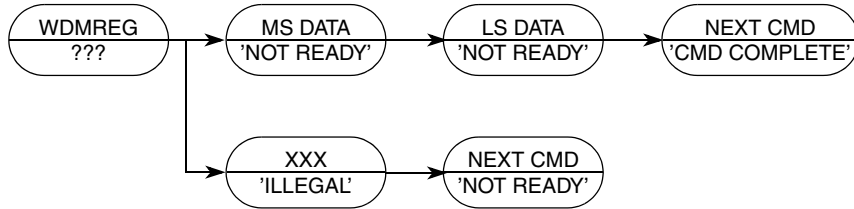
Figure 26-40. WDMREG Command Sequence

Operand Data: Longword data is written into the specified debug register. The data is supplied most-significant word first.

Result Data: Command complete status (0xFFFF) is returned when register write is complete.

26.6   Real-Time Debug Support

The ColdFire Family provides support 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 small intrusions into the real-time operation.

The debug module provides four types of breakpoints—PC with mask, PC without mask, operand address range, and data with mask. These breakpoints can be configured into one- or two-level triggers with the exact trigger response also programmable. The debug module programming model can be written from either the external development system using the debug serial interface or from the processor's supervisor programming model using the WDEBUG instruction. Only CSR is readable using the external development system.

26.6.1   Theory of Operation

Breakpoint hardware can be configured to respond to triggers in several ways. The desired response is programmed into TDR. As shown in Table 26-22, when a breakpoint is triggered, an indication (CSR[BSTAT]) is provided on the DDATA output port when it is not displaying captured processor status, operands, or branch addresses.

Table 26-22. DDATA[3:0]/CSR[BSTAT] Breakpoint Response

DDATA[3:0][1]	CSR[BSTAT] [1]	Breakpoint Status
0000	0000	No breakpoints enabled
0010	0001	Waiting for level-1 breakpoint
0100	0010	Level-1 breakpoint triggered
1010	0101	Waiting for level-2 breakpoint
1100	0110	Level-2 breakpoint triggered

[1] Encodings not shown are reserved for future use.

The breakpoint status is also posted in the CSR. Note that CSR[BSTAT] is cleared by a CSR read when either a level-2 breakpoint is triggered or a level-1 breakpoint is triggered and a level-2 breakpoint is not enabled. Status is also cleared by writing to TDR.

BDM instructions use the appropriate registers to load and configure breakpoints. As the system operates, a breakpoint trigger generates the response defined in TDR.

PC breakpoints are treated in a precise manner—exception recognition and processing are initiated before the excepting instruction is executed. All other breakpoint events are recognized on the processor's local bus, but are made pending to the processor and sampled like other interrupt conditions. As a result, these interrupts are imprecise.

In systems that tolerate the processor being halted, a BDM-entry can be used. With TDR[TRC] = 01, a breakpoint trigger causes the core to halt (PST = 0xF).

If the processor core cannot be halted, the debug interrupt can be used. With the configuration TDR[TRC] = 10 the breakpoint trigger becomes a debug interrupt to the processor, which is treated higher than the nonmaskable level-7 interrupt request. As with all interrupts, it is made pending until the processor reaches a sample point, which occurs once per instruction. Again, the hardware forces the PC breakpoint to occur before the targeted instruction executes. This is possible because the PC breakpoint is enabled when interrupt sampling occurs. For address and data breakpoints, reporting is considered imprecise because several instructions may execute after the triggering address or data is detected.

As soon as the debug interrupt is recognized, the processor aborts execution and initiates exception processing. This event is signaled externally by the assertion of a unique PST value (PST = 0xD) for multiple cycles. The core enters emulator mode when exception processing begins. After the standard 8-byte exception stack is created, the processor fetches a unique exception vector, 12, from the vector table. (Refer to the *ColdFire Programmer's Reference Manual*).

Execution continues at the instruction address in the vector corresponding to the breakpoint triggered. All interrupts are ignored while the processor is in emulator mode. The debug interrupt handler can use supervisor instructions to save the necessary context, such as the state of all program-visible registers into a reserved memory area.

When debug interrupt operations complete, the RTE instruction executes and the processor exits emulator mode. After the debug interrupt handler completes execution, the external development system can use BDM commands to read the reserved memory locations.

If a hardware breakpoint such as a PC trigger is left unmodified by the debug interrupt service routine, another debug interrupt is generated after the completion of the RTE instruction. Therefore the debug interrupt service routine should clear the respective TDR setting.

In the Rev. A implementation, if a hardware breakpoint (e.g., a PC trigger) is left unmodified by the debug interrupt service routine, another debug interrupt is generated after the RTE instruction completes execution. In the Rev. B design, the hardware has been modified to inhibit the generation of another debug interrupt during the first instruction after the RTE exits emulator mode. This behavior is consistent with the existing logic involving trace mode, where the execution of the first instruction occurs before another trace exception is generated. This Rev. B enhancement disables all hardware breakpoints until the first instruction after the RTE has completed execution, regardless of the programmed trigger response.

26.6.1.1 Emulator Mode

Emulator mode is used to facilitate non-intrusive emulator functionality. This mode can be entered in three different ways:

- Setting CSR[EMU] forces the processor into emulator mode. EMU is examined only if $\overline{\text{RSTI}}$ is negated and the processor begins reset exception processing. It can be set while the processor is halted before reset exception processing begins. See Section 26.5.1, "CPU Halt."

- A debug interrupt always puts the processor in emulation mode when debug interrupt exception processing begins.

- Setting CSR[TRC] forces the processor into emulation mode when trace exception processing begins.

While operating in emulation mode, the processor exhibits the following properties:

- All interrupts are ignored, including level-7 interrupts.

- If CSR[MAP] = 1, all caching of memory and the SRAM module are disabled. All memory accesses are forced into a specially mapped address space signaled by TT = 0x2, TM = 0x5 or 0x6. This includes stack frame writes and the vector fetch for the exception that forced entry into this mode.

The RTE instruction exits emulation mode. The processor status output port provides a unique encoding for emulator mode entry (0xD) and exit (0x7).

26.6.2 Concurrent BDM and Processor Operation

The debug module supports concurrent operation of both the processor and most BDM commands. BDM commands may be executed while the processor is running, except those following operations that access processor/memory registers:

- Read/write address and data registers

- Read/write control registers

For BDM commands that access memory, the debug module requests the processor's local bus. The processor responds by stalling the instruction fetch pipeline and waiting for current bus activity to complete before freeing the local bus for the debug module to perform its access. After the debug module bus cycle, the processor reclaims the bus.

Breakpoint registers must be carefully configured in a development system if the processor is executing. The debug module contains no hardware interlocks, so TDR should be disabled while breakpoint registers are loaded, after which TDR can be written to define the exact trigger. This prevents spurious breakpoint triggers.

Because there are no hardware interlocks in the debug unit, no BDM operations are allowed while the CPU is writing the debug's registers (DSCLK must be inactive).

26.7 Processor Status, DDATA 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 = 0x1, {PST = [0x89B], DDATA= operand}

where the {...} definition is optional operand information defined by the setting of the CSR.

The CSR provides capabilities to display operands based on reference type (read, write, or both). A PST value {0x8, 0x9, or 0xB} identifies the size and presence of valid data to follow on the DDATA output {1, 2, or 4 bytes}. Additionally, for certain change-of-flow branch instructions, CSR[BTB] provides the capability to display the target instruction address on the DDATA output {2, 3, or 4 bytes} using a PST value of {0x9, 0xA, or 0xB}.

26.7.1 User Instruction Set

Table 26-23 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 26-23. PST/DDATA Specification for User-Mode Instructions

Instruction	Operand Syntax	PST/DDATA
add.l	<ea>y,Rx	PST = 0x1, {PST = 0xB, DD = source operand}
add.l	Dy,<ea>x	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}
addi.l	#imm,Dx	PST = 0x1
addq.l	#imm,<ea>x	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}
addx.l	Dy,Dx	PST = 0x1
and.l	<ea>y,Dx	PST = 0x1, {PST = 0xB, DD = source operand}
and.l	Dy,<ea>x	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}
andi.l	#imm,Dx	PST = 0x1
asl.l	{Dy,#imm},Dx	PST = 0x1
asr.l	{Dy,#imm},Dx	PST = 0x1
bitrev.l	Dx	PST = 0x1
byterev.l	Dx	PST = 0x1
bcc.{b,w}		if taken, then PST = 0x5, else PST = 0x1
bchg	#imm,<ea>x	PST = 0x1, {PST = 0x8, DD = source}, {PST = 0x8, DD = destination}
bchg	Dy,<ea>x	PST = 0x1, {PST = 0x8, DD = source}, {PST = 0x8, DD = destination}
bclr	#imm,<ea>x	PST = 0x1, {PST = 0x8, DD = source}, {PST = 0x8, DD = destination}
bclr	Dy,<ea>x	PST = 0x1, {PST = 0x8, DD = source}, {PST = 0x8, DD = destination}

MCF5213 Reference Manual, Rev. 1.1

Table 26-23. PST/DDATA Specification for User-Mode Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
bra.{b,w}		PST = 0x5
bset	#imm,<ea>x	PST = 0x1, {PST = 0x8, DD = source}, {PST = 0x8, DD = destination}
bset	Dy,<ea>x	PST = 0x1, {PST = 0x8, DD = source}, {PST = 0x8, DD = destination}
bsr.{b,w}		PST = 0x5, {PST = 0xB, DD = destination operand}
btst	#imm,<ea>x	PST = 0x1, {PST = 0x8, DD = source operand}
btst	Dy,<ea>x	PST = 0x1, {PST = 0x8, DD = source operand}
clr.b	<ea>x	PST = 0x1, {PST = 0x8, DD = destination operand}
clr.l	<ea>x	PST = 0x1, {PST = 0xB, DD = destination operand}
clr.w	<ea>x	PST = 0x1, {PST = 0x9, DD = destination operand}
cmp.l	<ea>y,Rx	PST = 0x1, {PST = 0xB, DD = source operand}
cmpi.l	#imm,Dx	PST = 0x1
divs.l	<ea>y,Dx	PST = 0x1, {PST = 0xB, DD = source operand}
divs.w	<ea>y,Dx	PST = 0x1, {PST = 0x9, DD = source operand}
divu.l	<ea>y,Dx	PST = 0x1, {PST = 0xB, DD = source operand}
divu.w	<ea>y,Dx	PST = 0x1, {PST = 0x9, DD = source operand}
eor.l	Dy,<ea>x	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}
eori.l	#imm,Dx	PST = 0x1
ext.l	Dx	PST = 0x1
ext.w	Dx	PST = 0x1
extb.l	Dx	PST = 0x1
ff1.l	Dx	PST = 0x1
jmp	<ea>x	PST = 0x5, {PST = [0x9AB], DD = target address} [1]
jsr	<ea>x	PST = 0x5, {PST = [0x9AB], DD = target address}, {PST = 0xB , DD = destination operand}[1]
lea	<ea>y,Ax	PST = 0x1
link.w	Ay,#imm	PST = 0x1, {PST = 0xB, DD = destination operand}
lsl.l	{Dy,#imm},Dx	PST = 0x1
lsr.l	{Dy,#imm},Dx	PST = 0x1
move.b	<ea>y,<ea>x	PST = 0x1, {PST = 0x8, DD = source}, {PST = 0x8, DD = destination}
move.l	<ea>y,<ea>x	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}
move.w	<ea>y,<ea>x	PST = 0x1, {PST = 0x9, DD = source}, {PST = 0x9, DD = destination}
move.w	CCR,Dx	PST = 0x1
move.w	{Dy,#imm},CCR	PST = 0x1

Table 26-23. PST/DDATA Specification for User-Mode Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
movem.l	#list,<ea>x	PST = 0x1, {PST = 0xB, DD = destination},... [2]
movem.l	<ea>y,#list	PST = 0x1, {PST = 0xB, DD = source},... [2]
moveq	#imm,Dx	PST = 0x1
muls.l	<ea>y,Dx	PST = 0x1, {PST = 0xB, DD = source operand}
muls.w	<ea>y,Dx	PST = 0x1, {PST = 0x9, DD = source operand}
mulu.l	<ea>y,Dx	PST = 0x1, {PST = 0xB, DD = source operand}
mulu.w	<ea>y,Dx	PST = 0x1, {PST = 0x9, DD = source operand}
neg.l	Dx	PST = 0x1
negx.l	Dx	PST = 0x1
nop		PST = 0x1
not.l	Dx	PST = 0x1
or.l	<ea>y,Dx	PST = 0x1, {PST = 0xB, DD = source operand}
or.l	Dy,<ea>x	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}
ori.l	#imm,Dx	PST = 0x1
pea	<ea>y	PST = 0x1, {PST = 0xB, DD = destination operand}
pulse		PST = 0x4
rems.l	<ea>y,Dx:Dw	PST = 0x1, {PST = 0xB, DD = source operand}
remu.l	<ea>y,Dx:Dw	PST = 0x1, {PST = 0xB, DD = source operand}
rts		PST = 0x1, {PST = 0xB, DD = source operand}, PST = 0x5, {PST = [0x9AB], DD = target address}
scc	Dx	PST = 0x1
sub.l	<ea>y,Rx	PST = 0x1, {PST = 0xB, DD = source operand}
sub.l	Dy,<ea>x	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}
subi.l	#imm,Dx	PST = 0x1
subq.l	#imm,<ea>x	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}
subx.l	Dy,Dx	PST = 0x1
swap	Dx	PST = 0x1
trap	#imm	PST = 0x1[3]
trapf		PST = 0x1
tst.b	<ea>x	PST = 0x1, {PST = 0x8, DD = source operand}
tst.l	<ea>x	PST = 0x1, {PST = 0xB, DD = source operand}
tst.w	<ea>x	PST = 0x1, {PST = 0x9, DD = source operand}
unlk	Ax	PST = 0x1, {PST = 0xB, DD = destination operand}

Table 26-23. PST/DDATA Specification for User-Mode Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
wddata.b	<ea>y	PST = 0x4, {PST = 0x8, DD = source operand
wddata.l	<ea>y	PST = 0x4, {PST = 0xB, DD = source operand
wddata.w	<ea>y	PST = 0x4, {PST = 0x9, DD = source operand

1 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).

2 For Move Multiple instructions (MOVEM), the processor automatically generates line-sized transfers if the operand address reaches a 0-modulo-16 boundary and there are four or more registers to be transferred. For these line-sized transfers, the operand data is never captured nor displayed, regardless of the CSR value.
The automatic line-sized burst transfers are provided to maximize performance during these sequential memory access operations.

3 During normal exception processing, the PST output is driven to a 0xC 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 ProcessingPST = 0xC,{PST = 0xB,DD = destination},// stack frame
                {PST = 0xB,DD = destination},// stack frame
                {PST = 0xB,DD = source},// vector read
         PST = 0x5,{PST = [0x9AB],DD = target}// handler PC
```

The PST/DDATA specification for the reset exception is shown below:

```
Exception ProcessingPST = 0xC,
        PST = 0x5,{PST = [0x9AB],DD = target}//  handler PC
```

The initial references at address 0 and 4 are never captured nor displayed since these accesses are treated as instruction fetches.

For all types of exception processing, the PST = 0xC value is driven at all times, unless the PST output is needed for one of the optional marker values or for the taken branch indicator (0x5).

Table 26-24 shows the PST/DDATA specification for multiply-accumulate instructions.

Table 26-24. PST/DDATA Specification for MAC Instructions

Instruction	Operand Syntax	PST/DDATA
mac.l	Ry,Rx,Accx	PST = 0x1
mac.l	RyRx,<ea>,Rw,Accx	PST = 0x1, {PST = 0xB, DD = source operand}
mac.w	Ry,Rx,Accx	PST = 0x1
mac.w	Ry,Rx,<ea>,Rw,Accx	PST = 0x1, {PST = 0xB, DD = source operand}
move.l	<ea>y,Accx	PST = 0x1
move.l	Accy,Accx	PST = 0x1
move.l	<ea>y,MACR	PST = 0x1
move.l	<ea>y,MASK	PST = 0x1
move.l	<ea>y,Accext01	PST = 0x1
move.l	<ea>y,Accext23	PST = 0x1

Table 26-24. PST/DDATA Specification for MAC Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
move.l	Accy,Rx	PST = 0x1
move.l	MACSR,CCR	PST = 0x1
move.l	MACSR,Rx	PST = 0x1
move.l	MASK,Rx	PST = 0x1
move.l	Accext01,Rx	PST = 0x1
move.l	Accext23,Rx	PST = 0x1
msac.l	Ry,Rx,Accx	PST = 0x1
msac.l	Ry,Rx,<ea>,Rw,Accx	PST = 0x1, {PST = 0xB, DD = source operand}
msac.w	Ry,Rx,Accx	PST = 0x1
msac.w	Ry,Rx,<ea>,Rw,Accx	PST = 0x1, {PST = 0xB, DD = source operand}

26.7.2 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 26-25.

Table 26-25. PST/DDATA Specification for Supervisor-Mode Instructions

Instruction	Operand Syntax	PST/DDATA
cpushl		PST = 0x1
halt		PST = 0x1, PST = 0xF
move.w	SR,Dx	PST = 0x1
move.w	{Dy,#imm},SR	PST = 0x1, {PST = 0x3}
movec	Ry,Rc	PST = 0x1
rte		PST = 0x7, {PST = 0xB, DD = source operand}, {PST = 3},{ PST =0xB, DD =source operand}, PST = 0x5, {[PST = 0x9AB], DD = target address}
stldsr.w	#imm	PST = 0x1, {PST = 0xA, DD = destination operand, PST = 0x3}
stop	#imm	PST = 0x1, PST = 0xE
wdebug	<ea>y	PST = 0x1, {PST = 0xB, DD = source, PST = 0xB, DD = source}

The move-to-SR, STLDSR, and RTE instructions include an optional PST = 0x3 value, indicating an entry into user mode. Additionally, if the execution of a RTE instruction returns the processor to emulator mode, a multiple-cycle status of 0xD is signaled.

Similar to the exception processing mode, the stopped state (PST = 0xE) and the halted state (PST = 0xF) display this status throughout the entire time the ColdFire processor is in the given mode.

26.8 Freescale-Recommended BDM Pinout

The ColdFire BDM connector, Figure 26-41, is a 26-pin Berg connector arranged 2×13.

Developer reserved [1]	1	2	$\longrightarrow$	$\overline{\text{BKPT}}$
GND	3	4	$\longrightarrow$	DSCLK
GND	5	6	$\longleftarrow$	Developer reserved [1]
$\overline{\text{RSTI}}$	7	8	$\longrightarrow$	DSI
Pad-Voltage[2]	9	10	$\longleftarrow$	DSO
GND	11	12	$\longleftarrow$	PST3
PST2	13	14	$\longleftarrow$	PST1
PST0	15	16	$\longleftarrow$	DDATA3
DDATA2	17	18	$\longleftarrow$	DDATA1
DDATA0	19	20	$\longleftarrow$	GND
Freescale reserved	21	22		Freescale reserved
GND	23	24	$\longleftarrow$	PSTCLK
Core-Voltage	25	26	$\longrightarrow$	$\overline{\text{TA}}$
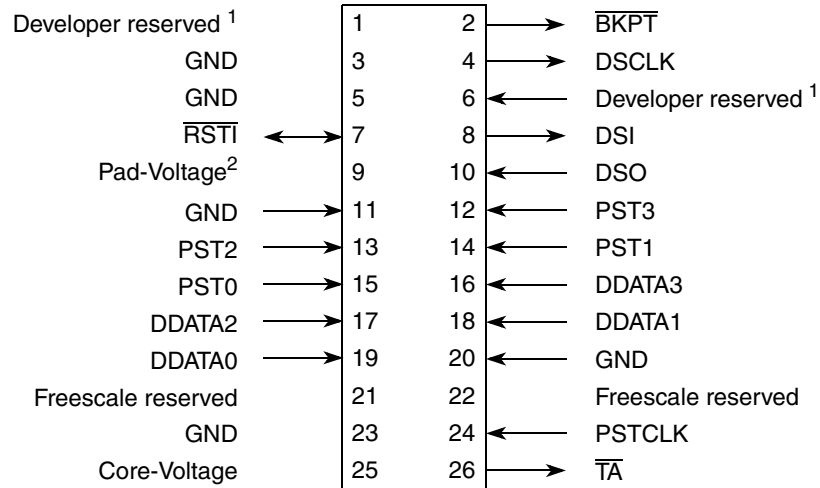
[1]Pins reserved for BDM developer use.
[2]Supplied by target

Figure 26-41. Recommended BDM Connector

Chapter 27
IEEE 1149.1 Test Access Port (JTAG)

27.1 Introduction

The joint test action group or JTAG is a dedicated user-accessible test logic, that complies with the IEEE 1149.1 standard for boundary-scan testability, to help with system diagnostic and manufacturing testing.

This architecture provides access to all data and chip control pins from the board-edge connector through the standard four-pin test access port (TAP) and the JTAG reset pin, $\overline{\text{TRST}}$.

27.1.1 Block Diagram

Figure 27-1 shows the block diagram of the JTAG module.
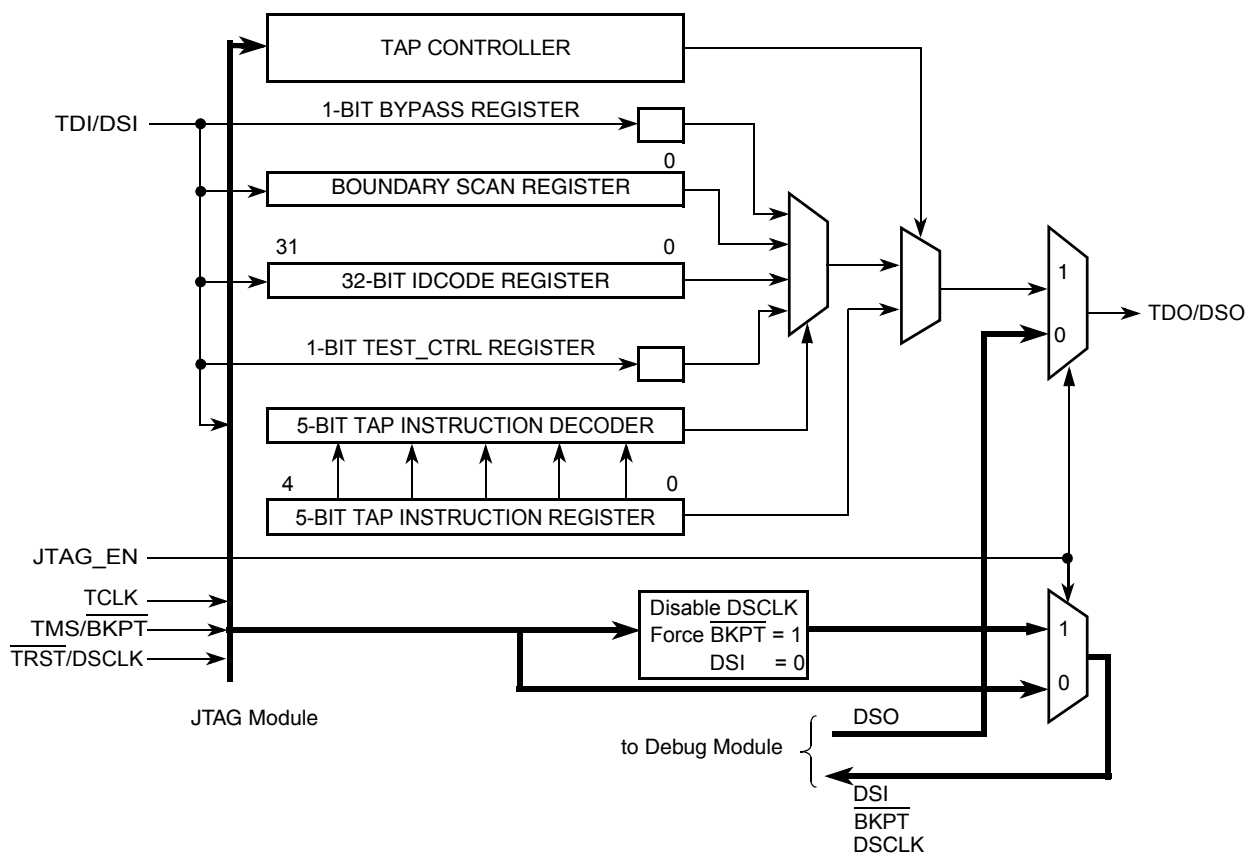
Figure 27-1. JTAG Block Diagram

27.1.2 Features

The basic features of the JTAG module are the following:

- Performs boundary-scan operations to test circuit board electrical continuity
- Bypasses instruction to reduce the shift register path to a single cell
- Sets chip output pins to safety states while executing the bypass instruction
- Samples the system pins during operation and transparently shift out the result
- Selects between JTAG TAP controller and background debug module (BDM) using a dedicated JTAG_EN pin

27.1.3 Modes of Operation

The JTAG_EN pin can select between the following modes of operation:

- JTAG mode (JTAG_EN = 1)
- BDM - background debug mode (For more information, refer to Section 26.5, "Background Debug Mode (BDM)") (JTAG_EN = 0)

27.2 External Signal Description

The JTAG module has five input and one output external signals, as described in Table 27-1.

Table 27-1. Signal Properties

Name	Direction	Function	Reset State	Pull up
JTAG_EN	Input	JTAG/BDM selector input	—	—
TCLK	Input	JTAG Test clock input	—	Active
TMS/$\overline{\text{BKPT}}$	Input	JTAG Test mode select / BDM Breakpoint	—	Active
TDI/DSI	Input	JTAG Test data input / BDM Development serial input	—	Active
$\overline{\text{TRST}}$/DSCLK	Input	JTAG Test reset input / BDM Development serial clock	—	Active
TDO/DSO	Output	JTAG Test data output / BDM Development serial output	Hi-Z / 0	—

27.2.1 JTAG Enable (JTAG_EN)

The JTAG_EN pin selects between the debug module and JTAG. If JTAG_EN is low, the debug module is selected; if it is high, the JTAG is selected. Table 27-2 summarizes the pin function selected depending upon JTAG_EN logic state.

Table 27-2. Pin Function Selected

	JTAG_EN = 0	JTAG_EN = 1	Pin Name
Module selected	BDM	JTAG	—
Pin Function	— $\overline{\text{BKPT}}$ DSI DSO DSCLK	TCLK TMS TDI TDO $\overline{\text{TRST}}$	TCLK $\overline{\text{BKPT}}$ DSI DSO DSCLK

When one module is selected, the inputs into the other module are disabled or forced to a known logic level as shown in Table 27-3, in order to disable the corresponding module.

Table 27-3. Signal State to the Disable Module

	JTAG_EN = 0	JTAG_EN = 1
Disabling JTAG	$\overline{\text{TRST}}$ = 0 TMS = 1	—
Disabling BDM	—	Disable DSCLK DSI = 0 $\overline{\text{BKPT}}$ = 1

NOTE

The JTAG_EN does not support dynamic switching between JTAG and BDM modes.

27.2.2 Test Clock Input (TCLK)

The TCLK pin is a dedicated JTAG clock input to synchronize the test logic. Pulses on TCLK shift data and instructions into the TDI pin on the rising edge and out of the TDO pin on the falling edge. TCLK is independent of the processor clock. The TCLK pin has an internal pull-up resistor and holding TCLK high or low for an indefinite period does not cause JTAG test logic to lose state information.

27.2.3 Test Mode Select/Breakpoint (TMS/$\overline{\text{BKPT}}$)

The TMS pin is the test mode select input that sequences the TAP state machine. TMS is sampled on the rising edge of TCLK. The TMS pin has an internal pull-up resistor.

The $\overline{\text{BKPT}}$ pin is used to request an external breakpoint. Assertion of $\overline{\text{BKPT}}$ puts the processor into a halted state after the current instruction completes.

27.2.4 Test Data Input/Development Serial Input (TDI/DSI)

The TDI pin receives serial test and data, which is sampled on the rising edge of TCLK. Register values are shifted in least significant bit (lsb) first. The TDI pin has an internal pull-up resistor.

The DSI pin provides data input for the debug module serial communication port.

27.2.5 Test Reset/Development Serial Clock ($\overline{\text{TRST}}$/DSCLK)

The $\overline{\text{TRST}}$ pin is an active low asynchronous reset input with an internal pull-up resistor that forces the TAP controller to the test-logic-reset state.

The DSCLK pin clocks the serial communication port to the debug module. Maximum frequency is 1/5 the processor clock speed. At the rising edge of DSCLK, the data input on DSI is sampled and DSO changes state.

27.2.6 Test Data Output/Development Serial Output (TDO/DSO)

The TDO pin is the lsb-first data output. Data is clocked out of TDO on the falling edge of TCLK. TDO is tri-stateable and is actively driven in the shift-IR and shift-DR controller states.

The DSO pin provides serial output data in BDM mode.

27.3 Memory Map/Register Definition

The JTAG module registers are not memory mapped and are only accessible through the TDO/DSO pin.

27.3.1 Instruction Shift Register (IR)

The JTAG module uses a -bit shift register with no parity. The IR transfers its value to a parallel hold register and applies an instruction on the falling edge of TCLK when the TAP state machine is in the update-IR state. To load an instruction into the shift portion of the IR, place the serial data on the TDI pin before each rising edge of TCLK. The msb of the IR is the bit closest to the TDI pin, and the lsb is the bit closest to the TDO pin. See Section 27.4.3, "JTAG Instructions" for a list of possible instruction codes.

27.3.2 IDCODE Register

The IDCODE is a read-only register; its value is chip dependent. For more information, see Section 27.4.3.1, "IDCODE Instruction."

IR[4:0]: 0_0001 (IDCODE)   Access: User read-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	PRN				DC						PIN										JEDEC											ID
W																																
Reset	0	0	0	0	0	1	1	1	0	1			Device Dependent								0	0	0	0	0	0	0	1	1	1	0	1
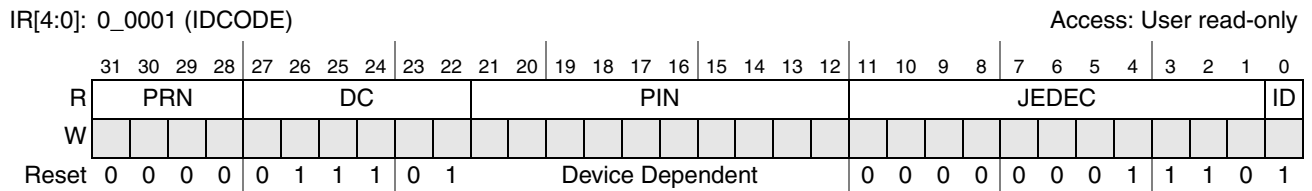
Figure 27-2. IDCODE Register

Table 27-4. IDCODE Field Descriptions

Field	Description
31–28 PRN	Part revision number. Indicate the revision number of the device.
27–22 DC	Freescale Design Center number.
21–12 PIN	Part identification number. Indicate the device number.
11–1 JEDEC	Joint Electron Device Engineering Council ID bits. Indicate the reduced JEDEC ID for Freescale.
0 ID	IDCODE register ID. This bit is set to 1 to identify the register as the IDCODE register and not the bypass register according to the IEEE standard 1149.1.

27.3.3 Bypass Register

The bypass register is a single-bit shift register path from TDI to TDO when the BYPASS, CLAMP, or HIGHZ instructions are selected. After entry into the capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

27.3.4 TEST_CTRL Register

The TEST_CTRL register is a 1-bit shift register path from TDI to TDO when the ENABLE_TEST_CTRL instruction is selected. The TEST_CTRL transfers its value to a parallel hold register on the rising edge of TCLK when the TAP state machine is in the update-DR state. The DSE bit selects the drive strength used in JTAG mode.

IR[4:0]: 0_0110   Access: User
                          read-only

	0
R	DSE
W	

Reset   0

Figure 27-3. 1-Bit TEST_CTRL Register

27.3.5 Boundary Scan Register

The boundary scan register is connected between TDI and TDO when the EXTEST or SAMPLE/PRELOAD instruction is selected. It captures input pin data, forces fixed values on output pins, and selects a logic value and direction for bidirectional pins or high impedance for tri-stated pins.

The boundary scan register contains bits for bonded-out and non bonded-out signals excluding JTAG signals, analog signals, power supplies, compliance enable pins, device configuration pins, and clock signals.

MCF5213 Reference Manual, Rev. 1.1

Preliminary

27.4 Functional Description

27.4.1 JTAG Module

The JTAG module consists of a TAP controller state machine, which is responsible for generating all control signals that execute the JTAG instructions and read/write data registers.

27.4.2 TAP Controller

The TAP controller is a state machine that changes state based on the sequence of logical values on the TMS pin. Figure 27-4 shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCLK signal.

Asserting the $\overline{\text{TRST}}$ signal asynchronously resets the TAP controller to the test-logic-reset state. As Figure 27-4 shows, holding TMS at logic 1 while clocking TCLK through at least five rising edges also causes the state machine to enter the test-logic-reset state, whatever the initial state.
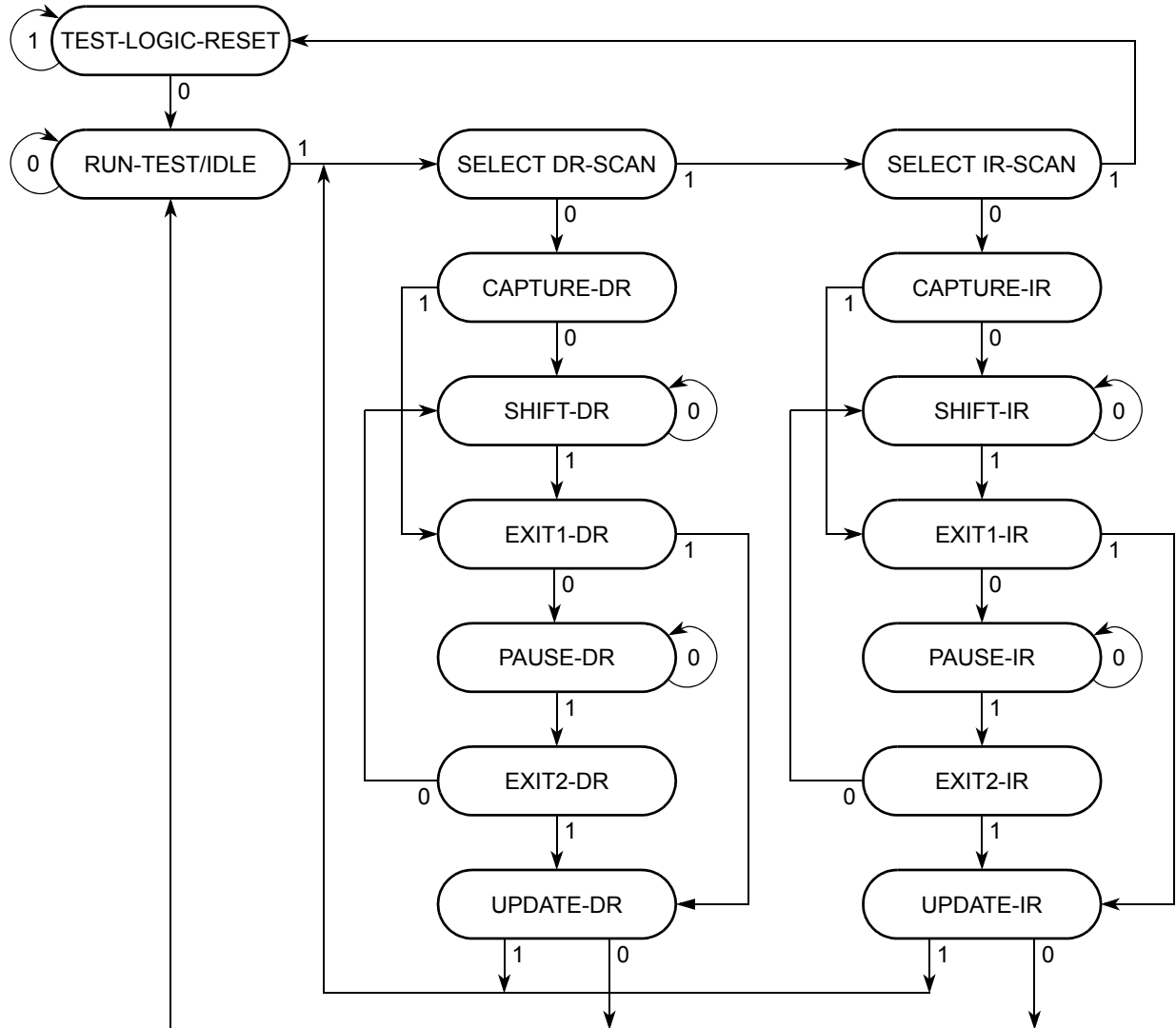
Figure 27-4. TAP Controller State Machine Flow

27.4.3   JTAG Instructions

Table 27-5 describes public and private instructions.

27.4.3.1   IDCODE Instruction

The IDCODE instruction selects the 32-bit IDCODE register for connection as a shift path between the TDI and TDO pin. This instruction allows interrogation of the MCU to determine its version number and other part identification data. The shift register lsb is forced to logic 1 on the rising edge of TCLK following entry into the capture-DR state. Therefore, the first bit to be shifted out after selecting the IDCODE register is always a logic 1. The remaining 31 bits are also forced to fixed values on the rising edge of TCLK following entry into the capture-DR state.

IDCODE is the default instruction placed into the instruction register when the TAP resets. Thus, after a TAP reset, the IDCODE register is selected automatically.

27.4.3.2   SAMPLE/PRELOAD Instruction

The SAMPLE/PRELOAD instruction has two functions:

- PRELOAD - initialize the boundary scan register update cells before selecting EXTEST or CLAMP. This is achieved by ignoring the data shifting out on the TDO pin and shifting in initialization data. The update-DR state and the falling edge of TCLK can then transfer this data to the update cells. The data is applied to the external output pins by the EXTEST or CLAMP instruction.

27.4.3.3   EXTEST Instruction

The external test (EXTEST) instruction selects the boundary scan register. It forces all output pins and bidirectional pins configured as outputs to the values preloaded with the SAMPLE/PRELOAD instruction and held in the boundary scan update registers. EXTEST can also configure the direction of bidirectional pins and establish high-impedance states on some pins. EXTEST asserts internal reset for the MCU system logic to force a predictable internal state while performing external boundary scan operations.

27.4.3.4   ENABLE_TEST_CTRL Instruction

The ENABLE_TEST_CTRL instruction selects a -bit shift register (TEST_CTRL) for connection as a shift path between the TDI and TDO pin. When the user transitions the TAP controller to the UPDATE_DR state, the register transfers its value to a parallel hold register.

27.4.3.5   HIGHZ Instruction

The HIGHZ instruction eliminates the need to backdrive the output pins during circuit-board testing. HIGHZ turns off all output drivers, including the 2-state drivers, and selects the bypass register. HIGHZ also asserts internal reset for the MCU system logic to force a predictable internal state.

27.4.3.6   CLAMP Instruction

The CLAMP instruction selects the 1-bit bypass register and asserts internal reset while simultaneously forcing all output pins and bidirectional pins configured as outputs to the fixed values that are preloaded and held in the boundary scan update register. CLAMP enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register.

27.4.3.7   BYPASS Instruction

The BYPASS instruction selects the bypass register, creating a single-bit shift register path from the TDI pin to the TDO pin. BYPASS enhances test efficiency by reducing the overall shift path when a device other than the ColdFire processor is the device under test on a board design with multiple chips on the overall boundary scan chain. The shift register lsb is forced to logic 0 on the rising edge of TCLK after

entry into the capture-DR state. Therefore, the first bit shifted out after selecting the bypass register is always logic 0. This differentiates parts that support an IDCODE register from parts that support only the bypass register.

27.5 Initialization/Application Information

27.5.1 Restrictions

The test logic is a static logic design, and TCLK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCLK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

Using the EXTEST instruction requires a circuit-board test environment that avoids device-destructive configurations in which MCU output drivers are enabled into actively driven networks.

Low-power stop mode considerations:

- The TAP controller must be in the test-logic-reset state to either enter or remain in the low-power stop mode. Leaving the test-logic-reset state negates the ability to achieve low-power, but does not otherwise affect device functionality.
- The TCLK input is not blocked in low-power stop mode. To consume minimal power, the TCLK input should be externally connected to EV_{DD}.
- The TMS, TDI, and $\overline{TRST}$ pins include on-chip pull-up resistors. For minimal power consumption in low-power stop mode, these three pins should be either connected to EV_{DD} or left unconnected.

27.5.2 Nonscan Chain Operation

Keeping the TAP controller in the test-logic-reset state ensures that the scan chain test logic is transparent to the system logic. It is recommended that TMS, TDI, TCLK, and $\overline{TRST}$ be pulled up. $\overline{TRST}$ could be connected to ground. However, since there is a pull-up on $\overline{TRST}$, some amount of current results. The internal power-on reset input initializes the TAP controller to the test-logic-reset state on power-up without asserting $\overline{TRST}$.

Preliminary

Appendix A
Register Memory Map

Table A-1 summarizes the address, name, and byte assignment for registers within the MCF5213 CPU space. Table A-2 lists an overview of the memory map for the on-chip modules, and Table A-3 is a detailed memory map including all of the registers for on-chip modules.

Table A-1. CPU Space Register Memory Map

Address	Name	Mnemonic	Size
CPU @ 0x002	Cache Control Register	CACR	32
CPU @ 0x004	Access Control Register 0	ACR0	32
CPU @ 0x005	Access Control Register 1	ACR1	32
CPU @ 0x800	Other Stack Pointer	OTHER_A7	32
CPU @ 0x801	Vector Base Register	VBR	32
CPU @ 0x804	MAC Status Register	MACSR	8
CPU @ 0x805	MAC Mask Register	MASK	16
CPU @ 0x806	MAC Accumulator 0	ACC0	16
CPU @ 0x80E	Status Register	SR	16
CPU @ 0x80F	Program Counter	PC	32
CPU @ 0xC04	Flash Base Address Register	FLASHBAR	32
CPU @ 0xC05	RAM Base Address Register	RAMBAR	32

Table A-2. Module Memory Map Overview

Address	Module	Size
0x0000_0000	On-chip Flash/RAM Array	1G
IPSBAR + 0x00_0000	System Control Module	64 bytes
IPSBAR + 0x00_0040	Reserved	64 bytes
IPSBAR + 0x00_0080	Reserved	128 bytes
IPSBAR + 0x00_0100	DMA (Channel 0)	64 bytes
IPSBAR + 0x00_0110	DMA (Channel 1)	64 bytes
IPSBAR + 0x00_0120	DMA (Channel 2)	64 bytes
IPSBAR + 0x00_0130	DMA (Channel 3)	64 bytes
IPSBAR + 0x00_0140	Reserved	196 bytes
IPSBAR + 0x00_0200	UART0	64 bytes
IPSBAR + 0x00_0240	UART1	64 bytes
IPSBAR + 0x00_0280	UART2	64 bytes
IPSBAR + 0x00_02C0	Reserved	64 bytes
IPSBAR + 0x00_0300	I^{2}C	64 bytes
IPSBAR + 0x00_0340	QSPI	64 bytes
IPSBAR + 0x00_0140	Reserved	128 bytes
IPSBAR + 0x00_0400	DMA Timer 0	64 bytes
IPSBAR + 0x00_0440	DMA Timer 1	64 bytes
IPSBAR + 0x00_0480	DMA Timer 2	64 bytes
IPSBAR + 0x00_04C0	DMA Timer 3	64 bytes
IPSBAR + 0x00_0500	Reserved	1792
IPSBAR + 0x00_0C00	Interrupt Controller 0	256 bytes
IPSBAR + 0x00_0D00	Reserved	256
IPSBAR + 0x00_0E00	Reserved	256
IPSBAR + 0x00_0F00	Global Interrupt Acknowledge Cycles	256 bytes
IPSBAR + 0x00_1000	Reserved	1M-4K
IPSBAR + 0x10_0000	Ports	64K
IPSBAR + 0x11_0000	Reset Controller, Chip Configuration, and Power Management	64K
IPSBAR + 0x12_0000	Clock Module	64K
IPSBAR + 0x13_0000	Edge Port	64K
IPSBAR + 0x14_0000	Reserved	64K
IPSBAR + 0x15_0000	Programmable Interval Timer 0	64K
IPSBAR + 0x16_0000	Programmable Interval Timer 1	64K
IPSBAR + 0x17_0000	Reserved	64K
IPSBAR + 0x18_0000	Reserved	64K

Table A-2. Module Memory Map Overview (continued)

Address	Module	Size
IPSBAR + 0x19_0000	ADC	64K
IPSBAR + 0x1A_0000	General Purpose Timer A	64K
IPSBAR + 0x1B_0000	PWM	64K
IPSBAR + 0x1C_0000	FlexCAN	64K
IPSBAR + 0x1D_0000	CFM (Flash) Control Registers	64K
IPSBAR + 0x1E_0000	Reserved	63M+128K
IPSBAR + 0x400_0000	CFM (Flash) Memory for IPS Reads and Writes	512K

Table A-3. Register Memory Map

Address	Name	Mnemonic	Size
SCM Registers			
IPSBAR + 0x000	Internal Peripheral System Base Address Register	IPSBAR	32
IPSBAR + 0x008	Copy of RAMBAR	RAMBAR	32
IPSBAR + 0x00C	Reserved	—	
IPSBAR + 0x010	Core Reset Status Register	CRSR	8
IPSBAR + 0x011	Core Watchdog Control Register	CWCR	8
IPSBAR + 0x012	Low-Power Interrupt Control Register	LPICR	8
IPSBAR + 0x013	Core Watchdog Service Register	CWSR	8
IPSBAR + 0x014	DMA Request Control Register	DMAREQC	32
IPSBAR + 0x01C	Default Bus Master Park Register	MPARK	32
IPSBAR + 0x020	Master Privilege Register	MPR	8
IPSBAR + 0x024	Peripheral Access Control Register 0	PACR0	8
IPSBAR + 0x025	Peripheral Access Control Register 1	PACR1	8
IPSBAR + 0x026	Peripheral Access Control Register 2	PACR2	8
IPSBAR + 0x027	Peripheral Access Control Register 3	PACR3	8
IPSBAR + 0x028	Peripheral Access Control Register 4	PACR4	8
IPSBAR + 0x02A	Peripheral Access Control Register 5	PACR5	8
IPSBAR + 0x02B	Peripheral Access Control Register 6	PACR6	8
IPSBAR + 0x02C	Peripheral Access Control Register 7	PACR7	8
IPSBAR + 0x02E	Peripheral Access Control Register 8	PACR8	8
IPSBAR + 0x030	Grouped Peripheral Access Control Register 0	GPACR0	8
IPSBAR + 0x031	Grouped Peripheral Access Control Register 1	GPACR1	8
DMA Registers			

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x100	Source Address Register 0	SAR0	32
IPSBAR + 0x104	Destination Address Register 0	DAR0	32
IPSBAR + 0x108	Byte Count Register 0	BCR0	32
IPSBAR + 0x10C	DMA Status Register 0	DSR0	8
IPSBAR + 0x120	Source Address Register 1	SAR1	32
IPSBAR + 0x124	Destination Address Register 1	DAR1	32
IPSBAR + 0x128	Byte Count Register 1	BCR1	32
IPSBAR + 0x12C	DMA Status Register 1	DSR1	8
IPSBAR + 0x130	Source Address Register 2	SAR2	32
IPSBAR + 0x134	Destination Address Register 2	DAR2	32
IPSBAR + 0x138	Byte Count Register 2	BCR2	32
IPSBAR + 0x13C	DMA Status Register 2	DSR2	8
IPSBAR + 0x140	Source Address Register 3	SAR3	32
IPSBAR + 0x144	Destination Address Register 3	DAR3	32
IPSBAR + 0x148	Byte Count Register 3	BCR3	32
IPSBAR + 0x14C	DMA Status Register 3	DSR3	8
UART Registers			
IPSBAR + 0x200	UART Mode Register 0[1]	UMR10, UMR20	8
IPSBAR + 0x204	(Read) UART Status Register 0	USR0	8
	(Write) UART Clock Select Register 0[1]	UCSR0	8
IPSBAR + 0x208	(Read) Reserved		8
	(Write) UART Command Register 0	UCR0	8
IPSBAR + 0x20C	(Read) UART Receive Buffer 0	URB0	8
	(Write) UART Transmit Buffer 0	UTB0	8
IPSBAR + 0x210	(Read) UART Input Port Change Register 0	UIPCR0	8
	(Write) UART Auxiliary Control Register 0[1]	UACR0	8
IPSBAR + 0x214	(Read) UART Interrupt Status Register 0	UISR0	8
	(Write) UART Interrupt Mask Register 0	UIMR0	8
IPSBAR + 0x218	(Read) Reserved		8
	UART Baud Rate Generator Register 10	UBG10	8
IPSBAR + 0x21C	(Read) Reserved		8
	UART Baud Rate Generator Register 20	UBG20	8

Preliminary

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x234	(Read) UART Input Port Register 0	UIP0	8
	(Write) Reserved		8
IPSBAR + 0x238	(Read) Reserved		8
	(Write) UART Output Port Bit Set Command Register 0	UOP10	8
IPSBAR + 0x23C	(Read) Reserved		8
	(Write) UART Output Port Bit Reset Command Register 0	UIP00	8
IPSBAR + 0x240	UART Mode Registers 1[1]	UMR11, UMR21	8
IPSBAR + 0x244	(Read) UART Status Register 1	USR1	8
	(Write) UART Clock Select Register 1[1]	UCSR1	8
IPSBAR + 0x248	(Read) Reserved		8
	(Write) UART Command Register 1	UCR1	8
IPSBAR + 0x24C	(UART/Read) UART Receive Buffer 1	URB1	8
	(UART/Write) UART Transmit Buffer 1	UTB1	8
IPSBAR + 0x250	(Read) UART Input Port Change Register 1	UIPCR1	8
	(Write) UART Auxiliary Control Register 1[1]	UACR1	8
IPSBAR + 0x254	(Read) UART Interrupt Status Register 1	UISR1	8
	(Write) UART Interrupt Mask Register 1	UIMR1	8
IPSBAR + 0x258	(Read) Reserved		8
	UART Baud Rate Generator Register 11	UBG11	8
IPSBAR + 0x25C	(Read) Reserved		8
	UART Baud Rate Generator Register 21	UBG21	8
IPSBAR + 0x274	(Read) UART Input Port Register 1	UIP1	8
	(Write) Reserved		8
IPSBAR + 0x278	(Read) Reserved		8
	(Write) UART Output Port Bit Set Command Register 1	UOP11	8
IPSBAR + 0x27C	(Read) Reserved		8
	(Write) UART Output Port Bit Reset Command Register 1	UIP01	8
IPSBAR + 0x280	UART Mode Register 2[1]	UMR12, UMR22	8
IPSBAR + 0x284	(Read) UART Status Register 2	USR2	8
	(Write) UART Clock Select Register 2[1]	UCSR2	8

MCF5213 Reference Manual, Rev. 1.1

Preliminary

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x288	(Read) Reserved		8
	(Write) UART Command Register 2	UCR2	8
IPSBAR + 0x28C	(Read) UART Receive Buffer 2	URB2	8
	(Write) UART Transmit Buffer 2	UTB2	8
IPSBAR + 0x290	(Read) UART Input Port Change Register 2	UIPCR2	8
	(Write) UART Auxiliary Control Register 2[1]	UACR2	8
IPSBAR + 0x294	(Read) UART Interrupt Status Register 2	UISR2	8
	(Write) UART Interrupt Mask Register 2	UIMR2	8
IPSBAR + 0x298	(Read) Reserved		8
	UART Baud Rate Generator Register 12	UBG12	8
IPSBAR + 0x29C	(Read) Reserved		8
	UART Baud Rate Generator Register 22	UBG22	8
IPSBAR + 0x2B4	(Read) UART Input Port Register 2	UIP2	8
	(Write) Reserved		8
IPSBAR + 0x2B8	(Read) Reserved		8
	(Write) UART Output Port Bit Set Command Register 2	UOP12	8
IPSBAR + 0x2BC	(Read) Reserved		8
	(Write) UART Output Port Bit Reset Command Register 2	UIP02	8
I^2C Registers			
IPSBAR + 0x300	I^2C Address Register	I2ADR	8
IPSBAR + 0x304	I^2C Frequency Divider Register	I2FDR	8
IPSBAR + 0x308	I^2C Control Register	I2CR	8
IPSBAR + 0x30C	I^2C Status Register	I2SR	8
IPSBAR + 0x310	I^2C Data I/O Register	I2DR	8
QSPI Registers			
IPSBAR + 0x340	QSPI Mode Register	QMR	16
IPSBAR + 0x344	QSPI Delay Register	QDLYR	16
IPSBAR + 0x348	QSPI Wrap Register	QWR	16
IPSBAR + 0x34C	QSPI Interrupt Register	QIR	16
IPSBAR + 0x350	QSPI Address Register	QAR	16
IPSBAR + 0x354	QSPI Data Register	QDR	16
DMA Timer Registers			

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x400	DMA Timer Mode Register 0	DTMR0	16
IPSBAR + 0x402	DMA Timer Extended Mode Register 0	DTXMR0	8
IPSBAR + 0x403	DMA Timer Event Register 0	DTER0	8
IPSBAR + 0x404	DMA Timer Reference Register 0	DTRR0	32
IPSBAR + 0x408	DMA Timer Capture Register 0	DTCR0	32
IPSBAR + 0x40C	DMA Timer Counter Register 0	DTCN0	32
IPSBAR + 0x440	DMA Timer Mode Register 1	DTMR1	16
IPSBAR + 0x442	DMA Timer Extended Mode Register 1	DTXMR1	8
IPSBAR + 0x443	DMA Timer Event Register 1	DTER1	8
IPSBAR + 0x444	DMA Timer Reference Register 1	DTRR1	32
IPSBAR + 0x448	DMA Timer Capture Register 1	DTCR1	32
IPSBAR + 0x44C	DMA Timer Counter Register 1	DTCN1	32
IPSBAR + 0x480	DMA Timer Mode Register 2	DTMR2	16
IPSBAR + 0x482	DMA Timer Extended Mode Register 2	DTXMR2	8
IPSBAR + 0x483	DMA Timer Event Register 2	DTER2	8
IPSBAR + 0x484	DMA Timer Reference Register 2	DTRR2	32
IPSBAR + 0x488	DMA Timer Capture Register 2	DTCR2	32
IPSBAR + 0x48C	DMA Timer Counter Register 2	DTCN2	32
IPSBAR + 0x4C0	DMA Timer Mode Register 3	DTMR3	16
IPSBAR + 0x4C2	DMA Timer Extended Mode Register 3	DTXMR3	8
IPSBAR + 0x4C3	DMA Timer Event Register 3	DTER3	8
IPSBAR + 0x4C4	DMA Timer Reference Register 3	DTRR3	32
IPSBAR + 0x4C8	DMA Timer Capture Register 3	DTCR3	32
IPSBAR + 0x4CC	DMA Timer Counter Register 3	DTCN3	32
Interrupt Controller 0			
IPSBAR + 0xC00	Interrupt Pending Register High 0	IPRH0	32
IPSBAR + 0xC04	Interrupt Pending Register Low 0	IPRL0	32
IPSBAR + 0xC08	Interrupt Mask Register High 0	IMRH0	32
IPSBAR + 0xC0C	Interrupt Mask Register Low 0	IMRL0	32
IPSBAR + 0xC10	Interrupt Force Register High 0	INTFRCH0	32
IPSBAR + 0xC14	Interrupt Force Register Low 0	INTFRCL0	32
IPSBAR + 0xC18	Interrupt Level Request Register 0	ILRR0	8

MCF5213 Reference Manual, Rev. 1.1

Preliminary

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0XC19	Interrupt Acknowledge Level and Priority Register 0	IACKLPR0	8
IPSBAR + 0xC41	Interrupt Control Register 0-01	ICR001	8
IPSBAR + 0xC42	Interrupt Control Register 0-02	ICR002	8
IPSBAR + 0xC43	Interrupt Control Register 0-03	ICR003	8
IPSBAR + 0xC44	Interrupt Control Register 0-04	ICR004	8
IPSBAR + 0xC45	Interrupt Control Register 0-05	ICR005	8
IPSBAR + 0xC46	Interrupt Control Register 0-06	ICR006	8
IPSBAR + 0xC47	Interrupt Control Register 0-07	ICR007	8
IPSBAR + 0xC48	Interrupt Control Register 0-08	ICR008	8
IPSBAR + 0xC49	Interrupt Control Register 0-09	ICR009	8
IPSBAR + 0xC4A	Interrupt Control Register 0-10	ICR010	8
IPSBAR + 0xC4B	Interrupt Control Register 0-11	ICR011	8
IPSBAR + 0xC4C	Interrupt Control Register 0-12	ICR012	8
IPSBAR + 0xC4D	Interrupt Control Register 0-13	ICR013	8
IPSBAR + 0xC4E	Interrupt Control Register 0-14	ICR014	8
IPSBAR + 0xC4F	Interrupt Control Register 0-15	ICR015	8
IPSBAR + 0xC51	Interrupt Control Register 0-17	ICR017	8
IPSBAR + 0xC52	Interrupt Control Register 0-18	ICR018	8
IPSBAR +0xC53	Interrupt Control Register 0-19	ICR019	8
IPSBAR + 0xC54	Interrupt Control Register 0-20	ICR020	8
IPSBAR + 0xC55	Interrupt Control Register 0-21	ICR021	8
IPSBAR + 0xC56	Interrupt Control Register 0-22	ICR022	8
IPSBAR + 0xC57	Interrupt Control Register 0-23	ICR023	8
IPSBAR + 0xC58	Interrupt Control Register 0-24	ICR024	8
IPSBAR + 0xC59	Interrupt Control Register 0-25	ICR025	8
IPSBAR + 0xC5A	Interrupt Control Register 0-26	ICR026	8
IPSBAR + 0xC5B	Interrupt Control Register 0-27	ICR027	8
IPSBAR + 0xC5C	Interrupt Control Register 0-28	ICR028	8
IPSBAR + 0xC5D	Interrupt Control Register 0-29	ICR029	8
IPSBAR + 0xC5E	Interrupt Control Register 0-30	ICR030	8
IPSBAR + 0xC5F	Interrupt Control Register 0-31	ICR031	8
IPSBAR + 0xC60	Interrupt Control Register 0-32	ICR032	8
IPSBAR + 0xC61	Interrupt Control Register 0-33	ICR033	8

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
ISPBAR + 0xC62	Interrupt Control Register 0-34	ICR034	8
IPSBAR + 0xC63	Interrupt Control Register 0-35	ICR035	8
IPSBAR + 0xC64	Interrupt Control Register 0-36	ICR036	8
IPSBAR + 0xC65	Interrupt Control Register 0-37	ICR037	8
IPSBAR + 0xC66	Interrupt Control Register 0-38	ICR038	8
IPSBAR + 0xC67	Interrupt Control Register 0-39	ICR039	8
IPSBAR + 0xC68	Interrupt Control Register 0-40	ICR040	8
IPSBAR + 0xC69	Interrupt Control Register 0-41	ICR041	8
IPSBAR + 0xC6A	Interrupt Control Register 0-42	ICR042	8
IPSBAR + 0xC6B	Interrupt Control Register 0-43	ICR043	8
IPSBAR + 0xC6C	Interrupt Control Register 0-44	ICR044	8
IPSBAR + 0xC6D	Interrupt Control Register 0-45	ICR45	8
IPSBAR + 0xC6E	Interrupt Control Register 0-46	ICR046	8
IPSBAR + 0xC6F	Interrupt Control Register 0-47	ICR047	8
IPSBAR + 0xC70	Interrupt Control Register 0-48	ICR048	8
IPSBAR + 0xC71	Interrupt Control Register 0-49	ICR049	8
IPSBAR + 0xC72	Interrupt Control Register 0-50	ICR050	8
IPSBAR + 0xC73	Interrupt Control Register 0-51	ICR051	8
IPSBAR + 0xC74	Interrupt Control Register 0-52	ICR052	8
IPSBAR + 0xC75	Interrupt Control Register 0-53	ICR053	8
IPSBAR + 0xC76	Interrupt Control Register 0-54	ICR054	8
IPSBAR + 0xC77	Interrupt Control Register 0-55	ICR055	8
IPSBAR + 0xC78	Interrupt Control Register 0-56	ICR056	8
IPSBAR + 0xC79	Interrupt Control Register 0-57	ICR057	8
IPSBAR + 0xC7A	Interrupt Control Register 0-58	ICR058	8
IPSBAR + 0xC7B	Interrupt Control Register 0-59	ICR059	8
IPSBAR + 0xC7C	Interrupt Control Register 0-60	ICR060	8
IPSBAR + 0xC7D	Interrupt Control Register 0-61	ICR061	8
IPSBAR + 0xC7E	Interrupt Control Register 0-62	ICR062	8
IPSBAR + 0xCE0	Software Interrupt Acknowledge Register 0	SWACKR0	8
IPSBAR + 0xCE4	Level 1 Interrupt Acknowledge Register 0	L1IACKR0	8
IPSBAR + 0xCE8	Level 2 Interrupt Acknowledge Register 0	L2IACKR0	8
IPSBAR + 0xCEC	Level 3 Interrupt Acknowledge Register 0	L3IACKR0	8

MCF5213 Reference Manual, Rev. 1.1

Preliminary

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0xCF0	Level 4 Interrupt Acknowledge Register 0	L4IACKR0	8
IPSBAR + 0xCF4	Level 5 Interrupt Acknowledge Register 0	L5IACKR0	8
IPSBAR + 0xCF8	Level 6 Interrupt Acknowledge Register 0	L6IACKR0	8
IPSBAR + 0xCFC	Level 7 Interrupt Acknowledge Register 0	L7IACKR0	8
Global Interrupt Acknowledge Cycle Registers			
IPSBAR + 0xFE4	Global Level 1 Interrupt Acknowledge Register	GL1IACKR	8
IPSBAR + 0xFE8	Global Level 2 Interrupt Acknowledge Register	GL2IACKR	8
IPSBAR + 0xFEC	Global Level 3 Interrupt Acknowledge Register	GL3IACKR	8
IPSBAR + 0xFF0	Global Level 4 Interrupt Acknowledge Register	GL4IACKR	8
IPSBAR + 0xFF4	Global Level 5 Interrupt Acknowledge Register	GL5IACKR	8
IPSBAR + 0xFF8	Global Level 6 Interrupt Acknowledge Register	GL6IACKR	8
IPSBAR + 0xFFC	Global Level 7 Interrupt Acknowledge Register	GL7IACKR	8
GPIO Registers			
IPSBAR + 0x10_0000	Reserved	—	8
IPSBAR + 0x10_0001	Reserved	—	8
IPSBAR + 0x10_0002	Reserved	—	8
IPSBAR + 0x10_0003	Reserved	—	8
IPSBAR + 0x10_0004	Reserved	—	8
IPSBAR + 0x10_0005	Reserved	—	8
IPSBAR + 0x10_0006	Reserved	—	8
IPSBAR + 0x10_0007	Reserved	—	8
IPSBAR + 0x10_0008	Port NQ Out Data Register	PORTNQ	8
IPSBAR + 0x10_0009	Port DD Output Data Register	PORTDD	8
IPSBAR + 0x10_000A	Port AN Output Data Register	PORTAN	8
IPSBAR + 0x10_000B	Port AS Output Data Register	PORTAS	8
IPSBAR + 0x10_000C	Reserved	—	8
IPSBAR + 0x10_000D	Port QS Output Data Register	PORTQS	8
IPSBAR + 0x10_000E	Port TA Output Data Register	PORTTA	8
IPSBAR + 0x10_000F	Port TC Output Data Register	PORTTC	8
IPSBAR + 0x10_0010	Port TD Output Data Register	PORTTD	8
IPSBAR + 0x10_0011	Port UA Output Data Register	PORTUA	8
IPSBAR + 0x10_0012	Port UB Output Data Register	PORTUB	8

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x10_0013	Port UC Output Data Register	PORTUC	8
IPSBAR + 0x10_0014	Reserved	—	8
IPSBAR + 0x10_0015	Reserved	—	8
IPSBAR + 0x10_0016	Reserved	—	8
IPSBAR + 0x10_0017	Reserved	—	8
IPSBAR + 0x10_0018	Reserved	—	8
IPSBAR + 0x10_0019	Reserved	—	8
IPSBAR + 0x10_001A	Reserved	—	8
IPSBAR + 0x10_001B	Reserved	—	8
IPSBAR + 0x10_001C	Port NQ Data Direction Register	DDRNQ	8
IPSBAR + 0x10_001D	Port DD Data Direction Register	DDRDD	8
IPSBAR + 0x10_001E	Port AN Data Direction Register	DDRAN	8
IPSBAR + 0x10_001F	Port AS Data Direction Register	DDRAS	8
IPSBAR + 0x10_0020	Reserved	—	8
IPSBAR + 0x10_0021	Port QS Data Direction Register	DDRQS	8
IPSBAR + 0x10_0022	Port TA Data Direction Register	DDRTA	8
IPSBAR + 0x10_0023	Port TC Data Direction Register	DDRTC	8
IPSBAR + 0x10_0024	Port TD Data Direction Register	DDRTD	8
IPSBAR + 0x10_0025	Port UA Data Direction Register	DDRUA	8
IPSBAR + 0x10_0026	Port UB Data Direction Register	DDRUB	8
IPSBAR + 0x10_0027	Port UC Data Direction Register	DDRUC	8
IPSBAR + 0x10_0028	Reserved	—	8
IPSBAR + 0x10_0029	Reserved	—	8
IPSBAR + 0x10_002A	Reserved	—	8
IPSBAR + 0x10_002B	Reserved	—	8
IPSBAR + 0x10_002C	Reserved	—	8
IPSBAR + 0x10_002D	Reserved	—	8
IPSBAR + 0x10_002E	Reserved	—	8
IPSBAR + 0x10_002F	Reserved	—	8
IPSBAR + 0x10_0030	Port NQ Pin Data/Set Data Register	PORTNQ/ SETNQ	8
IPSBAR + 0x10_0031	Port DD Pin Data/Set Data Register	PORTDDP/ SETDD	8

MCF5213 Reference Manual, Rev. 1.1

Preliminary

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x10_0032	Port AN Pin Data/Set Data Register	PORTDDP/ SETAN	8
IPSBAR + 0x10_0033	Port AS Pin Data/Set Data Register	PORTELP/ SETAS	8
IPSBAR + 0x10_0034	Reserved	—	8
IPSBAR + 0x10_0035	Port QS Pin Data/Set Data Register	PORTQSP/ SETQS	8
IPSBAR + 0x10_0036	Port TA Pin Data/Set Data Register	PORTSDP/ SETTA	8
IPSBAR + 0x10_0037	Port TC Pin Data/Set Data Register	PORTTCP/ SETTC	8
IPSBAR + 0x10_0038	Port TD Pin Data/Set Data Register	PORTTDP/ SETTD	8
IPSBAR + 0x10_0039	Port UA Pin Data/Set Data Register	PORTUAP/ SETUA	8
IPSBAR + 0x10_003A	Port UB Pin Data/Set Data Register	PORTUAP/ SETUB	8
IPSBAR + 0x10_003B	Port UC Pin Data/Set Data Register	PORTUAP/ SETUC	8
IPSBAR + 0x10_003C	Reserved	—	8
IPSBAR + 0x10_003D	Reserved	—	8
IPSBAR + 0x10_003E	Reserved	—	8
IPSBAR + 0x10_003F	Reserved	—	8
IPSBAR + 0x10_0040	Reserved	—	8
IPSBAR + 0x10_0041	Reserved	—	8
IPSBAR + 0x10_0042	Reserved	—	8
IPSBAR + 0x10_0043	Reserved	—	8
IPSBAR + 0x10_0044	Port NQ Clear Output Data Register	CLRNQ	8
IPSBAR + 0x10_0045	Port DD Clear Output Data Register	CLRDD	8
IPSBAR + 0x10_0046	Port AN Clear Output Data Register	CLRAN	8
IPSBAR + 0x10_0047	Port AS Clear Output Data Register	CLRAS	8
IPSBAR + 0x10_0048	Reserved	—	8
IPSBAR + 0x10_0049	Port QS Clear Output Data Register	CLRQS	8
IPSBAR + 0x10_004A	Port TA Clear Output Data Register	CLRTA	8
IPSBAR + 0x10_004B	Port TC Clear Output Data Register	CLRTC	8
IPSBAR + 0x10_004C	Port TD Clear Output Data Register	CLRTD	8

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x10_004D	Port UA Clear Output Data Register	CLRUA	8
IPSBAR + 0x10_004E	Port UB Clear Output Data Register	CLRUB	8
IPSBAR + 0x10_004F	Port UC Clear Output Data Register	CLRUC	8
IPSBAR + 0x10_0050	Port NQ Pin Assignment Register	PNQPAR	8
IPSBAR + 0x10_0051	Port DD Pin Assignment Register	PDDPAR	8
IPSBAR + 0x10_0052	Port AN Pin Assignment Register	PANPAR	8
IPSBAR + 0x10_0054	Port QS Pin Assignment Register	PQSPAR	16
IPSBAR + 0x10_0056	Port TA Pin Assignment Register	PTAPAR	8
IPSBAR + 0x10_0057	Port TC Pin Assignment Register	PTCPAR	8
IPSBAR + 0x10_0058	Port TD Pin Assignment Register	PTDPAR	8
IPSBAR + 0x10_0059	Port UA Pin Assignment Register	PUAPAR	8
IPSBAR + 0x10_005A	Port UB Pin Assignment Register	PUBPAR	8
IPSBAR + 0x10_005B	Port UC Pin Assignment Register	PUCPAR	8
IPSBAR + 0x10_0078	Pin Slew Rate Register	PSSR	32
IPSBAR + 0x10_007C	Pin Drive StrengthRegister	PDSR	32
Reset Control, Chip Configuration, and Power Management Registers			
IPSBAR + 0x11_0000	Reset Control Register	RCR	8
IPSBAR + 0x11_0001	Reset Status Register	RSR	8
IPSBAR + 0x11_0004	Chip Configuration Register	CCR	16
IPSBAR + 0x11_0007	Low-Power Control Register	LPCR	8
IPSBAR + 0x11_0008	Reset Configuration Register	RCON	16
IPSBAR + 0x11_000A	Chip Identification Register	CIR	16
Clock Module Registers			
IPSBAR + 0x12_0000	Synthesizer Control Register	SYNCR	16
IPSBAR + 0x12_0002	Synthesizer Status Register	SYNSR	16
Edge Port Registers			
IPSBAR + 0x13_0000	EPORT Pin Assignment Register	EPPAR	16
IPSBAR + 0x13_0002	EPORT Data Direction Register	EPDDR	8
IPSBAR + 0x13_0003	EPORT Interrupt Enable Register	EPIER	8
0x0013_0004	EPORT Data Register	EPDR	8
IPSBAR + 0x13_0005	EPORT Pin Data Register	EPPDR	8
0x0013_0006	EPORT Flag Register	EPFR	8

MCF5213 Reference Manual, Rev. 1.1

Preliminary

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
Programmable Interrupt Timer 0 Registers			
IPSBAR + 0x15_0000	PIT Control and Status Register 0	PCSR 0	16
IPSBAR + 0x15_0002	PIT Modulus Register 0	PMR 0	16
IPSBAR + 0x15_0004	PIT Count Register 0	PCNTR 0	16
Programmable Interrupt Timer 1 Registers			
IPSBAR + 0x16_0000	PIT Control and Status Register 1	PCSR 1	16
IPSBAR + 0x16_0002	PIT Modulus Register 1	PMR 1	16
IPSBAR + 0x16_0004	PIT Count Register 1	PCNTR 1	16
ADC Registers			
IPSBAR + 0x19_0000	ADC Module Configuration Register	ADCMCR	16
IPSBAR + 0x19_0006	Port QA Data Register	PORTQA	8
IPSBAR + 0x19_0007	Port QB Data Register	PORTQB	8
IPSBAR + 0x19_0008	Port QA Data Direction Register	DDRQA	8
IPSBAR + 0x19_0009	Port QB Data Direction Register	DDRQB	8
IPSBAR + 0x19_000A	ADC Control Register 0	ACR0	16
IPSBAR + 0x19_000C	ADC Control Register 1	ACR1	16
IPSBAR + 0x19_000E	ADC Control Register 2	ACR2	16
IPSBAR + 0x19_0010	ADC Status Register 0	ASR0	16
IPSBAR + 0x19_0012	ADC Status Register 1	ASR1	16
IPSBAR + 0x19_0200– 0x19_027E	Conversion Command Word Table	CCW0– CCW63	64x16
IPSBAR + 0x19_0280– 0x19_02FE	Right Justified, Unsigned Result Register	RJURR0– RJURR63	64x16
IPSBAR + 0x19_0300– 0x19_037E	Left Justified, Signed Result Register	LJSRR0– LJSRR63	64x16
IPSBAR + 0x19_0380– 0x19_03FE	Left Justified, Unsigned Result Register	LJURR0– LJURR63	64x16
General Purpose Timer A Registers			
IPSBAR + 0x1A_0000	GPTA IC/OC Select Register	GPTAIOS	8
IPSABAR + 0x1A_0001	GPTA Compare Force Register	GPTACFORC	8
IPSBAR + 0x1A_0002	GPTA Output Compare 3 Mask Register	GPTAOC3M	8
IPSBAR + 0x1A_0003	GPTA Output Compare 3 Data Register	GPTAOC3D	8
IPSBAR + 0x1A_0004	GPTA Counter Register	GPTACNT	16

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x1A_0006	GPTA System Control Register 1	GPTASCR1	8
IPSBAR + 0x1A_0008	GPTA Toggle-on-Overflow Register	GPTATOV	8
IPSBAR + 0x1A_0009	GPTA Control Register 1	GPTACTL1	8
IPSBAR + 0x1A_000B	GPTA Control Register 2	GPTACTL2	8
IPSBAR + 0x1A_000C	GPTA Interrupt Enable Register	GPTAIE	8
IPSBAR + 0x1A_000D	GPTA System Control Register 2	GPTASCR2	8
IPSBAR + 0x1A_000E	GPTA Flag Register 1	GPTAFLG1	8
IPSBAR + 0x1A_000F	GPTA Flag Register 2	GPTAFLG2	8
IPSBAR + 0x1A_0010	GPTA Channel 0 Register	GPTAC0	16
IPSBAR + 0x1A_0012	GPTA Channel 1 Register	GPTAC1	16
IPSBAR + 0x1A_0014	GPTA Channel 2 Register	GPTAC2	16
IPSBAR + 0x1A_0016	GPTA Channel 3 Register	GPTAC3	16
IPSBAR + 0x1A_0018	Pulse Accumulator Control Register	GPTAPACTL	8
IPSBAR + 0x1A_0019	Pulse Accumulator Flag Register	GPTPAFLG	8
IPSBAR + 0x1A_001a	Pulse Accumulator Counter Register	GPTAPACNT	8
IPSBAR + 0x1A_001D	GPTA Port Data Register	GPTAPORT	8
IPSBAR + 0x1A_001E	GPTA Port Data Direction Register	GPTADDR	8
Pulse Width Modulator			
IPSBAR + 0x1B_0000	PWM Enable Register	PWME	8
IPSBAR + 0x1B_0001	PWM Polarity Register	PWMPOL	8
IPSBAR + 0x1B_0002	PWM Clock Select Register	PWMCLK	8
IPSBAR + 0x1B_0003	PWM Prescale Clock Select Register	PWMPRCLK	8
IPSBAR + 0x1B_0004	PWM Center Align Enable Register	PWMCAE	8
IPSBAR + 0x1B_0005	PWM Control Register	PWMCTL	8
IPSBAR + 0x1B_0008	PWM Scale A Register	PWMSCLA	8
IPSBAR + 0x1B_0009	PWM Scale B Register	PWMSCLB	8
IPSBAR + 0x1B_000C	PWM channel Counter Register 0	PWMCNT0	8
IPSBAR + 0x1B_000D	PWM channel Counter Register 1	PWMCNT1	8
IPSBAR + 0x1B_000E	PWM channel Counter Register 2	PWMCNT2	8
IPSBAR + 0x1B_000F	PWM channel Counter Register 3	PWMCNT3	8
IPSBAR + 0x1B_0010	PWM channel Counter Register 4	PWMCNT4	8
IPSBAR + 0x1B_0011	PWM channel Counter Register 5	PWMCNT5	8

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x1B_0012	PWM channel Counter Register 6	PWMCNT6	8
IPSBAR + 0x1B_0013	PWM channel Counter Register 7	PWMCNT7	8
IPSBAR + 0x1B_0014	PWM Channel Period Register 0	PWMPER0	8
IPSBAR + 0x1B_0015	PWM Channel Period Register 1	PWMPER1	8
IPSBAR + 0x1B_0016	PWM Channel Period Register 2	PWMPER2	8
IPSBAR + 0x1B_0017	PWM Channel Period Register 3	PWMPER3	8
IPSBAR + 0x1B_0018	PWM Channel Period Register 4	PWMPER4	8
IPSBAR + 0x1B_0019	PWM Channel Period Register 5	PWMPER5	8
IPSBAR + 0x1B_001A	PWM Channel Period Register 6	PWMPER6	8
IPSBAR + 0x1B_001B	PWM Channel Period Register 7	PWMPER7	8
FlexCAN Registers			
IPSBAR + 0x1C_0000	Module Configuration Register	CANMCR	16
IPSBAR + 0x1C_0004	FlexCAN Control Register	CANCTRL	32
IPSBAR + 0x1C_0008	Free Running TImer	TIMER	32
IPSBAR + 0x1C_000C	Reserved	---	32
IPSBAR + 0x1C_0010	Rx Global Mask	RXGMASK	32
IPSBAR + 0x1C_0014	Rx Buffer 14 Mask	RX14MASK	32
IPSBAR + 0x1C_0018	Rx Buffer 15 Mask	RX15MASK	32
IPSBAR + 0x1C_001C	Error Counter Register	ERRCNT	32
IPSBAR + 0x1C_0020	Error and Status	ERRSTAT	32
IPSBAR + 0x1C_0024	Reserved	---	32
IPSBAR + 0x1C_0028	Interrupt Mask Register	IMASK	32
IPSBAR + 0x1C_002C	Reserved	---	32
IPSBAR + 0x1C_0030	Interrupt Flag Register	IFLAG	32
IPSBAR + 0x1C_0080	Message Buffer 0 - Message Buffer 15	MBUFF0–MBUFF15	16x16bytes
Flash Registers			
IPSBAR + 0x1D_0000	CFM Configuration Register	CFMMCR	16
IPSBAR + 0x1D_0002	CFM Clock Divider Register	CFMCLKD	8
IPSBAR + 0x1D_0008	CFM Security Register	CFMSEC	32
IPSBAR + 0x1D_0010	CFM Protection Register	CFMPROT	32
IPSBAR + 0x1D_0014	CFM Supervisor Access Register	CFMSACC	32
IPSBAR + 0x1D_0018	CFM Data Access Register	CFMDACC	32

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x1D_0020	CFM User Status Register	CFMUSTAT	8
IPSBAR + 0x1D_0024	CFM Command Register	CFMCMD	8

[1] UMR1n, UMR2n, and UCSRn should be changed only after the receiver/transmitter is issued a software reset command. That is, if channel operation is not disabled, undesirable results may occur.