MCF5282 and MCF5216 ColdFire® Microcontroller User's Manual

Devices Supported:
MCF5214
MCF5216
MCF5280
MCF5281
MCF5282

Document Number: MCF5282UM
Rev. 3
2/2009

How to Reach Us:

Home Page:
www.freescale.com

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

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

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

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

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

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

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

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2009. All rights reserved.

MCF5282UM
Rev. 3
2/2009

Chapter 1
Overview

Chapter 2
ColdFire Core

Chapter 3
Enhanced Multiply-Accumulate Unit (EMAC)

Chapter 4
Cache

Chapter 5
Static RAM (SRAM)

Chapter 6
ColdFire Flash Module (CFM)

Chapter 7
Power Management

Chapter 8
System Control Module (SCM)

Chapter 9
Clock Module

Chapter 10
Interrupt Controller Modules

Chapter 11
Edge Port Module (EPORT)

Chapter 12
Chip Select Module

Chapter 13
External Interface Module (EIM)

Chapter 14
Signal Descriptions

Chapter 15
Synchronous DRAM Controller Module

Chapter 16
DMA Controller Module

Chapter 17
Fast Ethernet Controller (FEC)

Chapter 18
Watchdog Timer Module

Chapter 19
Programmable Interrupt Timers (PIT0–PIT3)

Chapter 20
General Purpose Timer Modules (GPTA and GPTB)

Chapter 21
DMA Timers (DTIM0–DTIM3)

Chapter 22
Queued Serial Peripheral Interface (QSPI)

Chapter 23
UART Modules

Chapter 24
I²C Interface

Chapter 25
FlexCAN

Chapter 26
General Purpose I/O Module

Chapter 27
Chip Configuration Module (CCM)

Chapter 28
Queued Analog-to-Digital Converter (QADC)

Chapter 29
Reset Controller Module

Chapter 30
Debug Support

Chapter 31
IEEE 1149.1 Test Access Port (JTAG)

Chapter 32
Mechanical Data

Chapter 33
Electrical Characteristics

Appendix A
Register Memory Map

Appendix B
Revision History

About This Book

The primary objective of this user's manual is to define the functionality of the MCF5282 processor for use by software and hardware developers.

The information in this book, except for changes to the flash and Ethernet functionality, also applies to the MCF5280, MCF5281, MCF5216, and MCF5214.

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 MCF5282. 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.

Suggested Reading

This section lists additional reading that provides background for the information in this manual as well as general information about the ColdFire architecture.

General Information

The following documentation provides useful information about the ColdFire architecture and computer architecture in general:

- *ColdFire Programmers Reference Manual, R1.0* (MCF5200PRM/AD)
- *Using Microprocessors and Microcomputers: The Motorola Family*, William C. Wray, Ross Bannatyne, Joseph D. Greenfield
- *Computer Architecture: A Quantitative Approach*, Second Edition, by John L. Hennessy and David A. Patterson.
- *Computer Organization and Design: The Hardware/Software Interface*, Second Edition, David A . Patterson and John L. Hennessy.

ColdFire Documentation

ColdFire documentation is available from the sources listed on the back cover of this manual, as well as our web site, http://www.freescale.com/coldfire.

- User’s manuals — These books provide details about individual ColdFire implementations and are intended to be used in conjunction with the *ColdFire Programmers Reference Manual.*
- Data sheets — Data sheets provide specific data regarding pin-out diagrams, bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations.
- Product briefs — Each device has a product brief that provides an overview of its features. This document is roughly equivalent to the overview (Chapter 1) of an device’s reference manual.
- Application notes — These short documents address specific design issues useful to programmers and engineers working with Freescale Semiconductor processors.

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:

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
&	AND logical operator
|	OR 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.

Chapter 1
Overview

This chapter provides an overview of the microprocessor features, including the major functional components.

1.1 Key Features

A block diagram of the MCF528*x* and MCF521*x* is shown in Figure 1-1. The main features are as follows:

- Static Version 2 ColdFire variable-length RISC processor
  - Static operation
  - On-chip 32-bit address and data path
  - Processor core and bus frequency up to 80 MHz
  - Sixteen general-purpose 32-bit data and address registers
  - ColdFire ISA_A with extensions to support the user stack pointer register, and four new instructions for improved bit processing
  - Enhanced Multiply-Accumulate (EMAC) unit with four 48-bit accumulators to support 32-bit signal processing algorithms
  - Illegal instruction decode that allows for 68K emulation support
- System debug support
  - Real-time trace for determining dynamic execution path
  - Background debug mode (BDM) for in-circuit debugging
  - Real time debug support, with one user-visible hardware breakpoint register (PC and address with optional data) that can be configured into a 1- or 2-level trigger
- On-chip memories
  - 2-Kbyte cache, configurable as instruction-only, data-only, or split I-/D-cache
  - 64-Kbyte dual-ported SRAM on CPU internal bus, accessible by core and non-core bus masters (e.g., DMA, FEC) with standby power supply support
  - 512 Kbytes of interleaved Flash memory supporting 2-1-1-1 accesses (256 Kbytes on the MCF5281 and MCF5214, no Flash on MCF5280)
    - This product incorporates SuperFlash® technology licensed from SST.
- 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
- Fast Ethernet Controller (FEC) (not available on the MCF5214 and MCF5216)
  - 10BaseT capability, half- or full-duplex
  - 100BaseT capability, half- or limited-throughput full-duplex
  - On-chip transmit and receive FIFOs
  - Built-in dedicated DMA controller

- Memory-based flexible descriptor rings
- Media-independent interface (MII) to transceiver (PHY)

- FlexCAN 2.0B Module
  - 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
  - Up to 16 message buffers (MBs)
    - Configurable as receive (Rx) or transmit (Tx)
    - Support standard and extended messages
  - Unused message buffer (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
    - 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
  - Programmable I/O modes
  - 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 ($\overline{\text{URTS}}$) and clear-to-send ($\overline{\text{UCTS}}$) lines for two UARTs
  - Transmit and receive FIFO buffers
- I^2C module
  - Interchip bus interface for EEPROMs, LCD controllers, A/D converters, and keypads
  - Fully compatible with industry-standard I^2C 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 master bit rates
  - Up to 16 pre-programmed transfers
- Queued analog-to-digital converter (QADC)
  - 8 direct, or up to 18 multiplexed, analog input channels
  - 10-bit resolution +/- 2 counts accuracy
  - Minimum 7 μS conversion time
  - Internal sample and hold
  - Programmable input sample time for various source impedances
  - Two conversion command queues with a total of 64 entries
  - Sub-queues possible using pause mechanism
  - Queue complete and pause software interrupts available on both queues
  - Queue pointers indicate current location for each queue
  - Automated queue modes initiated by:
    - External edge trigger and gated trigger
    - Periodic/interval timer, within QADC module [Queue 1 and 2]
    - Software command
  - Single-scan or continuous-scan of queues
  - Output data readable in three formats:
    - Right-justified unsigned
    - Left-justified signed
    - Left-justified unsigned
  - Unused analog channels can be used as digital I/O
  - Low pin-count configuration implemented
- Four 32-bit DMA timers
  - 15-ns resolution at 80 MHz (66 MHz for MCF5214 and MCF5216)
  - 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 reference-compare
  - DMA trigger capability on input capture or reference-compare
- Two 4-channel general purpose timers
  - Four 16-bit input capture/output compare channels per timer
  - 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
  - One dual-mode pulse accumulation channel per timer
- Four periodic interrupt timers (PITs)
  - 16-bit counter
  - Selectable as free running or count down
- Software watchdog timer
  - 16-bit counter
  - Low-power mode support
- Phase locked loop (PLL)
  - Crystal or external oscillator reference
  - 2- to 10-MHz reference frequency for normal PLL mode
  - 33- to 80-MHz (66 MHz for MCF5214/16) oscillator reference frequency for 1:1 mode
  - Low-power modes supported
  - Separate clock output pin
- Two interrupt controllers
  - Support for up to 63 interrupt sources per interrupt controller (a total of 126), organized as follows:
    - 56 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)
  - 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 connections between the 11 DMA requesters in the UARTs (3), 32-bit timers (4) plus external logic (4) and the four DMA channels
- External bus interface
  - Glueless connections to external memory devices (e.g., SRAM, Flash, ROM, etc.)
  - SDRAM controller supports 8-, 16-, and 32-bit wide memory devices
  - Glueless interface to SRAM devices with or without byte strobe inputs
  - Programmable wait state generator
  - 32-bit bidirectional data bus
  - 24-bit address bus
  - Up to seven chip selects available
  - Byte/write enables (byte strobes)

  - Ability to boot from internal Flash memory or external memories that are 8, 16, or 32 bits wide
- 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
  - Support for single chip, master, and test modes
  - Selects one of four clock modes
  - Sets boot device and its data port width
  - Configures output pad drive strength
  - Unique part identification number and part revision number
- General purpose I/O interface
  - Up to 142 bits of general purpose I/O for MCF5280/1/2
  - Up to 134 bits of general purpose I/O for MCF5214/6
  - Coherent 32-bit control
  - Bit manipulation supported via set/clear functions
  - Unused peripheral pins may be used as extra GPIO
- JTAG support for system-level board testing

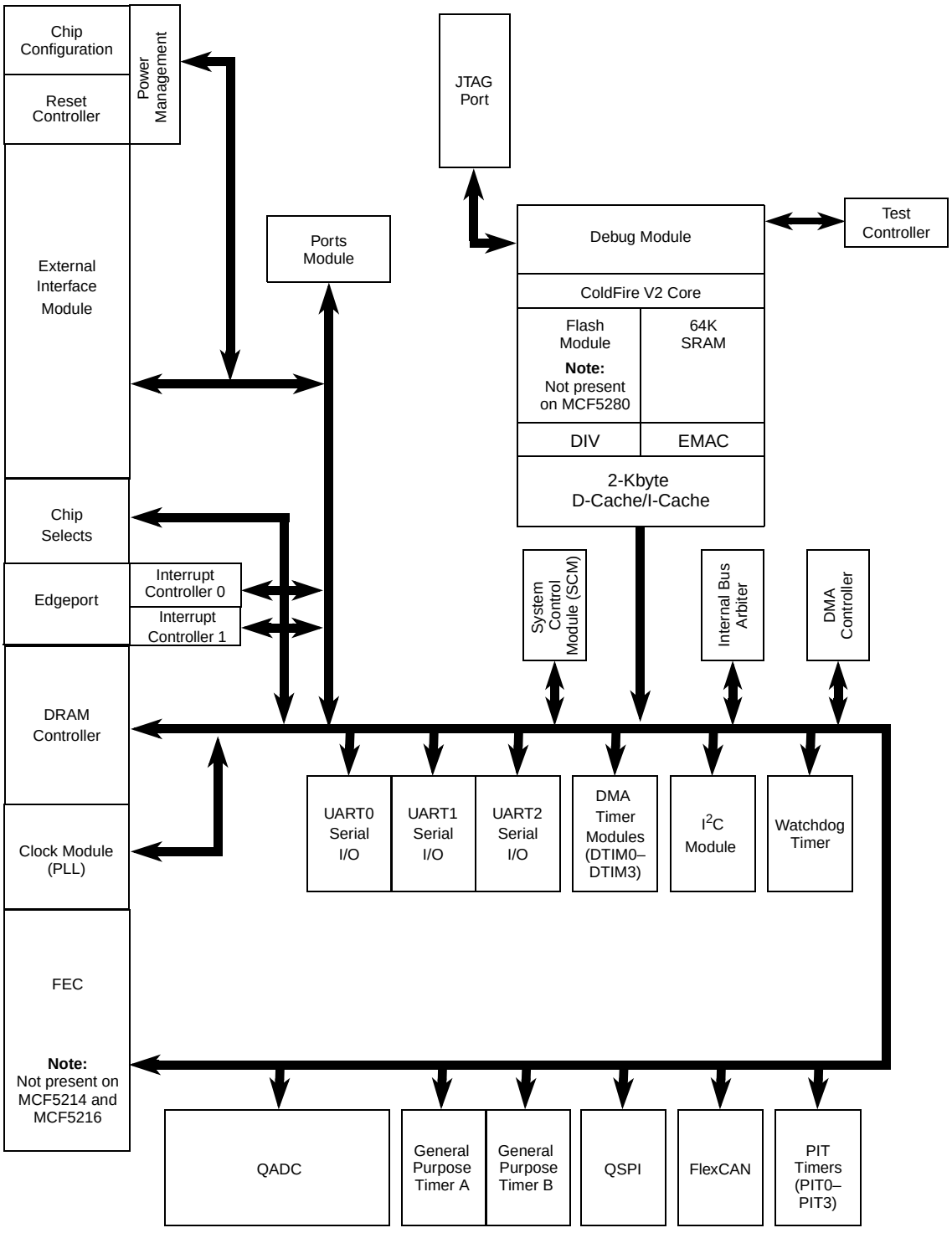


Figure 1-1. MCF528*x* and MCF521*x* Block Diagram

1.1.1 Version 2 ColdFire Core

The 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 MCF5282 core includes the enhanced multiply-accumulate unit (EMAC) for improved signal processing capabilities. The EMAC implements a 4-stage execution pipeline, optimized for 32 x 32 bit operations, with support for four 48-bit accumulators. Supported operands include 16- and 32-bit signed and unsigned integers, signed fractional operands, and a complete set of instructions to process these data types. The EMAC provides superb support for execution of DSP operations within the context of a single processor at a minimal hardware cost.

1.1.1.1 Cache

The 2-Kbyte cache can be configured into one of three possible organizations: a 2-Kbyte instruction cache, a 2-Kbyte data cache or a split 1-Kbyte instruction/1-Kbyte data cache. The configuration is software-programmable by control bits within the privileged cache configuration register (CACR). In all configurations, the cache is a direct-mapped single-cycle memory, organized as 128 lines, each containing 16 bytes of data. The memories consist of a 128-entry tag array (containing addresses and control bits) and a 2-Kbyte data array, organized as 512 x 32 bits. The tag and data arrays are accessed in parallel using the following address bits:

Table 1-1. Cache Configuration

Configuration	Tag Address	Data Array Address
2 Kbyte I-Cache	[10:4]	[10:2]
2 Kbyte D-Cache	[10:4]	[10:2]
Split I-/D-Cache 0 Instruction Fetches Operand Accesses	 0, [9:4] 1, [9:4]	 0, [9:2] 1, [9:2]

If the desired address is mapped into the cache memory, the output of the data array is driven onto the ColdFire core's local data bus, completing the access in a single cycle. If the data is not mapped into the tag memory, a cache miss occurs and the processor core initiates a 16-byte line-sized fetch. The cache module includes a 16-byte line fill buffer used as temporary storage during miss processing. For all data cache configurations, the memory operates in write-through mode and all operand writes generate an external bus cycle.

1.1.1.2 SRAM

The SRAM module provides a general-purpose 64-Kbyte memory block that the ColdFire core can access in a single cycle. The location of the memory block can be set to any 64-Kbyte boundary within the 4-Gbyte address space. The memory is ideal for storing critical code or data structures, for use as the system stack, or for storing FEC data buffers. 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 non-core bus masters, for example the DMA and/or the FEC. 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. As an example, system performance can be increased significantly if Ethernet packets are moved from the FEC into the SRAM (rather than external memory) prior to any processing.

1.1.1.3 Flash

This product incorporates SuperFlash® technology licensed from SST. The ColdFire Flash Module (CFM) is a non-volatile memory (NVM) module for integration with the processor core. The CFM is constructed with eight banks of 32K x 16-bit Flash arrays to generate 512 Kbytes of 32-bit Flash memory

NOTE

The CFM on the MCF5281 and MCF5214 is constructed with four banks of 32K x 16-bit Flash arrays to generate 256 Kbytes of 32-bit Flash memory.

The MCF5280 does not contain a CFM.

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 programming voltage source. The CFM interfaces to the V2 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. It also provides a read datapath for non-core masters (for example, DMA).

1.1.1.4 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 real-time trace and debug information. This allows the processor and system to be debugged at full speed without the need for costly in-circuit emulators. The debug interface is a superset of the BDM interface provided on Freescale’s 683xx family of parts.

The on-chip breakpoint resources include a total of 6 programmable registers—a set of address registers (with two 32-bit registers), a set of data registers (with a 32-bit data register plus a 32-bit data mask register), and one 32-bit PC register plus a 32-bit PC mask register. 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.

To support program trace, the Version 2 debug module provides processor status (PST[3:0]) and debug data (DDATA[3:0]) ports. These buses and the CLKOUT output provide execution status, captured operand data, and branch target addresses defining the dynamic execution path of the processor at the CPU’s clock rate.

1.1.2 System Control Module

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 base address register (IPSBAR), the processor’s dual-port RAM

base address register (RAMBAR), and system control registers that include low-power and core watchdog timer control.

1.1.3 External Interface Module (EIM)

The external interface module handles the transfer of information between the internal core and memory, peripherals, or other processing elements in the external address space.

Programmable chip-select outputs provide signals to enable external memory and peripheral circuits, providing all handshaking and timing signals for automatic wait-state insertion and data bus sizing.

Base memory address and block size are programmable, with some restrictions. For example, the starting address must be on a boundary that is a multiple of the block size. Each chip select can be configured to provide read and write enable signals suitable for use with most popular static RAMs and peripherals. Data bus width (8-bit, 16-bit, or 32-bit) is programmable on all chip selects, and further decoding is available for protection from user mode access or read-only access.

1.1.4 Chip Select

Programmable chip select outputs provide a glueless connection to external memory and peripheral circuits, providing all handshaking and timing signals for automatic wait-state insertion and data bus sizing.

1.1.5 Power Management

The MCF5282 incorporates several low-power modes of operation which are entered under program control and exited by several external trigger events. An integrated Power-On Reset (POR) circuit monitors the input supply and forces an MCU reset as the supply voltage rises. The Low Voltage Detect (LVD) section monitors the supply voltage and is configurable to force a reset or interrupt condition if it falls below the LVD trip point. The RAM standby switch provides power to RAM when the supply voltage is higher than the standby voltage. If the supply voltage to chip falls below the standby battery voltage, the RAM is switched over to the standby supply.

1.1.6 General Input/Output Ports

All of the pins associated with the external bus 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 this function, all 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 on the MCF5282 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.

1.1.7 Interrupt Controllers (INTC0/INTC1)

There are two interrupt controllers on the MCF5282, each of which can support up to 63 interrupt sources for a total of 126. Each interrupt controller is organized as 7 levels with 9 interrupt sources per level. Each interrupt source has a unique interrupt vector, and 56 of the 63 sources of a given controller provide a programmable level [1-7] and priority within the level.

1.1.8 SDRAM Controller

The SDRAM controller provides all required signals for glueless interfacing to a variety of JEDEC-compliant SDRAM devices. SRAS/SCAS address multiplexing is software configurable for different page sizes. To maintain refresh capability without conflicting with concurrent accesses on the address and data buses, SRAS, SCAS, DRAMW, SDRAM_CS[1:0], and SCKE are dedicated SDRAM signals.

1.1.9 Test Access Port

The MCF5282 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 MCF5282 implementation supports the following:

- Perform boundary-scan operations to test circuit board electrical continuity
- Sample MCF5282 system pins during operation and transparently shift out the result in the boundary scan register
- Bypass the MCF5282 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

1.1.10 UART Modules

The MCF5282 contains three full-duplex UARTs that function independently. The three UARTs can be clocked by the system clock, eliminating the need for an external crystal.

Each UART has the following features:

- Each can be clocked by the system 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

1.1.11 DMA Timers (DTIM0-DTIM3)

There are four independent, DMA-transfer-generating 32-bit timers (DTIM0, DTIM1, DTIM2, DTIM3) on the MCF5282. 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*x* signals. If the system clock is selected, it can be divided by 16 or 1. The selected 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.

1.1.12 General-Purpose Timers (GPTA/GPTB)

The two general-purpose timers (GPTA and GPTB) are 4-channel timer modules. Each timer consists of a 16-bit programmable counter driven by a 7-stage programmable prescaler. Each of the four channels for each timer 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.

1.1.13 Periodic Interrupt Timers (PIT0-PIT3)

The four periodic interrupt timers (PIT0, PIT1, PIT2, PIT3) are 16-bit timers that provide precise 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.

1.1.14 Software Watchdog Timer

The watchdog timer is a 16-bit timer that facilitates recovery from runaway code. The watchdog counter is a free-running down-counter that generates a reset on underflow. To prevent a reset, software must periodically restart the countdown.

1.1.15 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.

1.1.16 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 DCR*n*[START] bit or the occurrence of a hardware event from one of the on-chip peripheral devices, such as a capture event or an output reference event in a DMA timer (DTIM*n*) for each channel. The DMA controller supports dual-address mode to on-chip devices.

1.1.17 Reset

The reset controller is provided to determine the cause of reset, assert the appropriate reset signals to the system, and keep track of what caused the last reset. The power management registers for the internal low-voltage detect (LVD) circuit are implemented in the reset module. There are seven sources of reset:

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

External reset on the $\overline{\text{RSTO}}$ pin is software-assertable independent of chip reset state. There are also software-readable status flags indicating the cause of the last reset, and LVD control and status bits for setup and use of LVD reset or interrupt.

1.2 MCF5282-Specific Features

1.2.1 Fast Ethernet Controller (FEC)

The MCF5282's integrated Fast Ethernet Controller (FEC) performs the full set of IEEE 802.3/Ethernet CSMA/CD media access control and channel interface functions. The FEC supports connection and functionality for the 10/100 Mbps 802.3 media independent interface (MII). It requires an external transceiver (PHY) to complete the interface to the media.

NOTE

The MCF5214 and MCF5216 devices do not contain an FEC module.

1.2.2 FlexCAN

The FlexCAN module is a communication controller implementing the CAN protocol. The CAN protocol can be 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. FlexCAN contains 16 message buffers.

1.2.3 I^2C Bus

The I^2C 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.

1.2.4 Queued Serial Peripheral Interface (QSPI)

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

1.2.5 Queued Analog-to-Digital Converter (QADC)

The QADC is a 10-bit, unipolar, successive approximation converter. A maximum of 8 analog input channels can be supported using internal multiplexing. A maximum of 18 input channels can be supported in the internal/external multiplexed mode.

The QADC consists of an analog front-end and a digital control subsystem. The analog section includes input pins, an analog multiplexer, and sample and hold analog circuits. The analog conversion is performed by the digital-to-analog converter (DAC) resistor-capacitor array and a high-gain comparator.

The digital control section contains queue control logic to sequence the conversion process and interrupt generation logic. Also included are the periodic/interval timer, control and status registers, the 64-entry conversion command word (CCW) table, and the 64-entry result table.

Chapter 2
ColdFire Core

2.1 Introduction

This section describes the organization of the Version 2 (V2) 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*.

2.1.1 Overview

As with all ColdFire cores, the V2 ColdFire core is comprised of two separate pipelines decoupled by an instruction buffer.

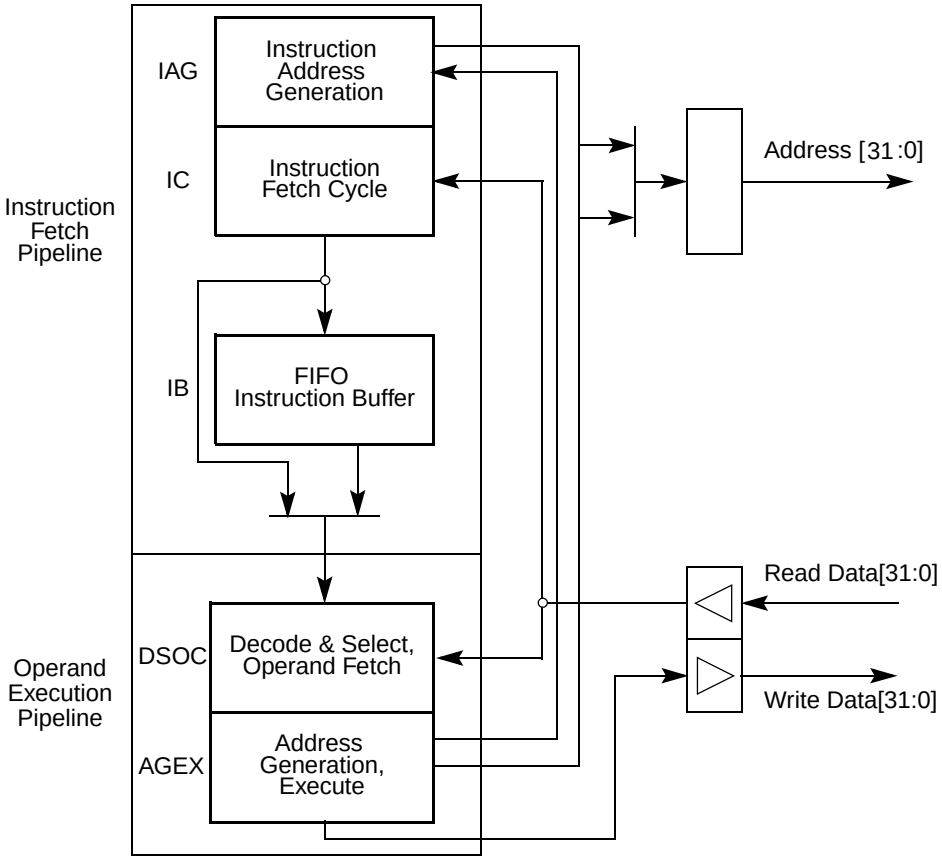


Figure 2-1. V2 ColdFire Core Pipelines

The instruction fetch pipeline (IFP) is a two-stage pipeline for prefetching instructions. The prefetched instruction stream is then gated into the two-stage operand execution pipeline (OEP), which decodes the

instruction, fetches the required operands and then executes the required function. Because the IFP and OEP pipelines are decoupled by an instruction buffer serving as a FIFO queue, the IFP is able to prefetch instructions in advance of their actual use by the OEP thereby minimizing time stalled waiting for instructions.

The V2 ColdFire core pipeline stages include the following:

- Two-stage instruction fetch pipeline (IFP) (plus optional instruction buffer stage)
  - — Instruction address generation (IAG) — Calculates the next prefetch address
  - — Instruction fetch cycle (IC)—Initiates prefetch on the processor's local bus
  - — Instruction buffer (IB) — Optional buffer stage minimizes fetch latency effects using FIFO queue
- Two-stage operand execution pipeline (OEP)
  - — Decode and select/operand fetch cycle (DSOC)—Decodes instructions and fetches the required components for effective address calculation, or the operand fetch cycle
  - — Address generation/execute cycle (AGEX)—Calculates operand address or executes the instruction

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

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

The resulting pipeline and local bus structure allow the V2 ColdFire core to deliver sustained high performance across a variety of demanding embedded applications.

2.2 Memory Map/Register Description

The following sections describe the processor registers in the user and supervisor programming models. The programming model is selected based on the processor privilege level (user mode or supervisor mode) as defined by the S bit of the status register (SR). Table 2-1 lists the processor registers.

The user-programming model consists of the following registers:

- 16 general-purpose 32-bit registers (D0–D7, A0–A7)
- 32-bit program counter (PC)
- 8-bit condition code register (CCR)
- EMAC registers (described fully in Chapter 3, "Enhanced Multiply-Accumulate Unit (EMAC:
  - — Four 48-bit accumulator registers partitioned as follows:
    - – Four 32-bit accumulators (ACC0–ACC3)
    - – Eight 8-bit accumulator extension bytes (two per accumulator). These are grouped into two 32-bit values for load and store operations (ACCEXT01 and ACCEXT23).

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

— One 16-bit mask register (MASK)

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

The supervisor programming model is to be used only by system control software to implement restricted operating system functions, I/O control, and memory management. All accesses that affect the control features of ColdFire processors are in the supervisor programming model, 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)
- 32-bit access control registers (ACR0, ACR1)
- Two 32-bit memory base address registers (RAMBAR, FLASHBAR)

Table 2-1. ColdFire Core Programming Model

BDM[1]	Register	Width (bits)	Access	Reset Value	Written with MOVEC	Section/Page
Supervisor/User Access Registers						
Load: 0x080 Store: 0x180	Data Register 0 (D0)	32	R/W	0xCF20_6080	No	2.2.1/2-4
Load: 0x081 Store: 0x181	Data Register 1 (D1)	32	R/W	0x13B0_1080	No	2.2.1/2-4
Load: 0x082–7 Store: 0x182–7	Data Register 2–7 (D2–D7)	32	R/W	Undefined	No	2.2.1/2-4
Load: 0x088–8E Store: 0x188–8E	Address Register 0–6 (A0–A6)	32	R/W	Undefined	No	2.2.2/2-4
Load: 0x08F Store: 0x18F	Supervisor/User A7 Stack Pointer (A7)	32	R/W	Undefined	No	2.2.3/2-5
0x804	MAC Status Register (MACSR)	32	R/W	0x0000_0000	No	3.2.1/3-3
0x805	MAC Address Mask Register (MASK)	32	R/W	0xFFFF_FFFF	No	3.2.2/3-5
0x806, 0x809, 0x80A, 0x80B	MAC Accumulators 0–3 (ACC0–3)	32	R/W	Undefined	No	3.2.3/3-6
0x807	MAC Accumulator 0,1 Extension Bytes (ACCext01)	32	R/W	Undefined	No	3.2.4/3-7
0x808	MAC Accumulator 2,3 Extension Bytes (ACCext23)	32	R/W	Undefined	No	3.2.4/3-7
0x80E	Condition Code Register (CCR)	8	R/W	Undefined	No	2.2.4/2-6

Table 2-1. ColdFire Core Programming Model (continued)

BDM[1]	Register	Width (bits)	Access	Reset Value	Written with MOVEC	Section/Page
0x80F	Program Counter (PC)	32	R/W	Contents of location 0x0000_0004	No	2.2.5/2-7
Supervisor Access Only Registers						
0x002	Cache Control Register (CACR)	32	R/W	0x0000_0000	Yes	2.2.6/2-7
0x004–5	Access Control Register 0–1 (ACR0–1)	32	R/W	See Section	Yes	2.2.7/2-7
0x800	User/Supervisor A7 Stack Pointer (OTHER_A7)	32	R/W	Contents of location 0x0000_0000	No	2.2.3/2-5
0x801	Vector Base Register (VBR)	32	R/W	0x0000_0000	Yes	2.2.8/2-7
0x80E	Status Register (SR)	16	R/W	0x27--	No	2.2.9/2-8
0xC04	Flash Base Address Register (FLASHBAR)	32	R/W	0x0000_0000	Yes	2.2.10/2-8
0xC05	RAM Base Address Register (RAMBAR)	32	R/W	See Section	Yes	2.2.10/2-8

[1] The values listed in this column represent the Rc field used when accessing the core registers via the BDM port. For more information see Chapter 30, “Debug Support”.

2.2.1 Data Registers (D0–D7)

D0–D7 data registers are for bit (1-bit), byte (8-bit), word (16-bit) and longword (32-bit) operations; they can also be used as index registers.

NOTE

Registers D0 and D1 contain hardware configuration details after reset. See Section 2.3.4.15, “Reset Exception” for more details.

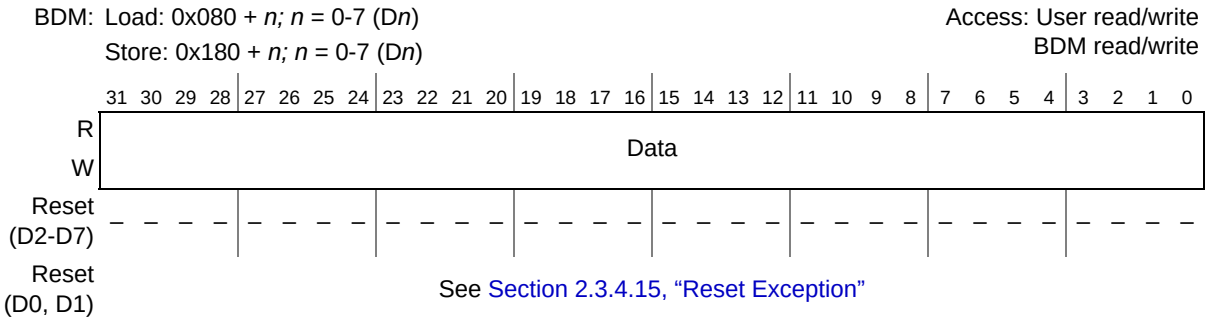


Figure 2-2. Data Registers (D0–D7)

2.2.2 Address Registers (A0–A6)

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

BDM: Load: 0x088 + *n*; *n* = 0–6 (A*n*)
Store: 0x188 + *n*; *n* = 0–6 (A*n*)

Access: User read/write
BDM read/write

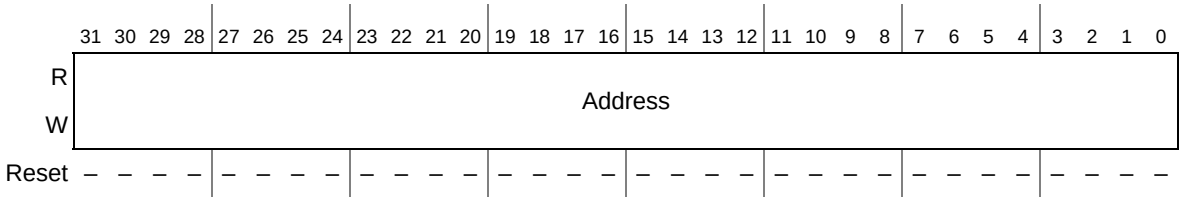


Figure 2-3. Address Registers (A0–A6)

2.2.3 Supervisor/User Stack Pointers (A7 and OTHER_A7)

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

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

The BDM programming model supports direct reads and writes to A7 and OTHER_A7. It is the responsibility of the external development system to determine, based on the setting of SR[S], the mapping of A7 and OTHER_A7 to the two program-visible definitions (SSP and USP). This functionality is enabled by setting the enable user stack pointer bit, CACR[EUSP]. If this bit is cleared, only a single stack pointer (A7), defined for ColdFire ISA_A, is available. EUSP is cleared at reset.

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

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

These instructions are described in the *ColdFire Family Programmer's Reference Manual*. All other instruction references to the stack pointer, explicit or implicit, access the active A7 register.

NOTE

The SSP is loaded during reset exception processing with the contents of location 0x0000_0000.

BDM: Load: 0x08F (A7)
Store: 0x18F (A7)
0x800 (OTHER_A7)

Access: A7: User or BDM read/write
OTHER_A7: Supervisor or BDM read/write

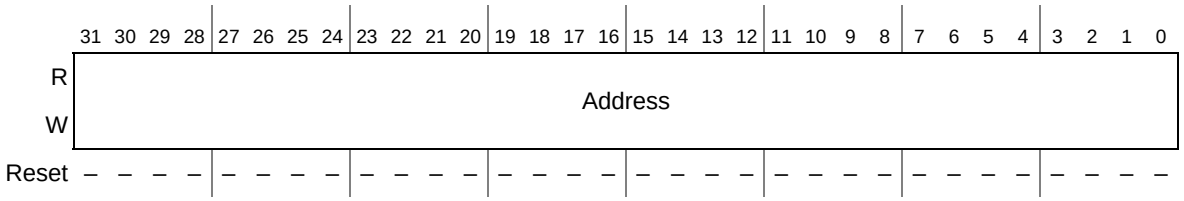


Figure 2-4. Stack Pointer Registers (A7 and OTHER_A7)

2.2.4 Condition Code Register (CCR)

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

BDM: LSB of Status Register (SR)

Access: User read/write
BDM read/write

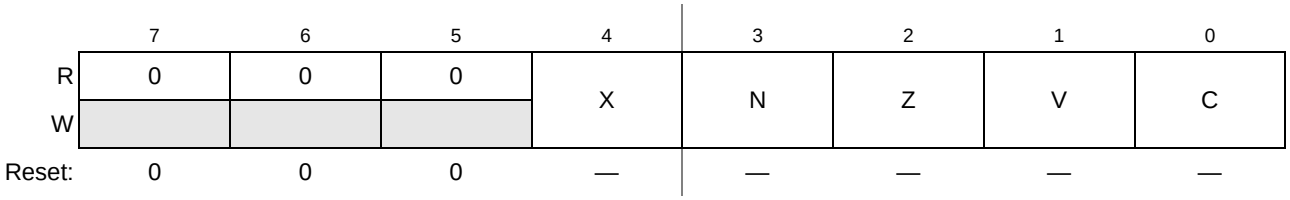


Figure 2-5. Condition Code Register (CCR)

Table 2-2. CCR Field Descriptions

Field	Description
7–5	Reserved, must be cleared.
4 X	Extend condition code bit. Set to the C-bit value for arithmetic operations; otherwise not affected or set to a specified result.
3 N	Negative condition code bit. Set if most significant bit of the result is set; otherwise cleared.
2 Z	Zero condition code bit. Set if result equals zero; otherwise cleared.
1 V	Overflow condition code bit. Set if an arithmetic overflow occurs implying the result cannot be represented in operand size; otherwise cleared.
0 C	Carry condition code bit. Set if a carry out of the operand msb occurs for an addition or if a borrow occurs in a subtraction; otherwise cleared.

2.2.5 Program Counter (PC)

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

The PC is initially loaded during reset exception processing with the contents of location 0x0000_0004.

BDM: 0x80F (PC)

Access: User read/write
BDM read/write

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

Figure 2-6. Program Counter Register (PC)

2.2.6 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 4.2.1, “Cache Control Register (CACR).”

2.2.7 Access Control Registers (ACR*n*)

The access control registers define attributes for user-defined memory regions. These attributes include the definition of cache mode, write protect, and buffer write enables. The ACRs are described in Section 4.2.2, “Access Control Registers (ACR0, ACR1).”

2.2.8 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 MB boundary.

BDM: 0x801 (VBR)

Access: Supervisor read/write
BDM read/write

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

Figure 2-7. Vector Base Register (VBR)

2.2.9 Status Register (SR)

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

BDM: 0x80E (SR)

Access: Supervisor read/write
BDM read/write

	System Byte								Condition Code Register (CCR)							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	T	0	S	M	0	I			0	0	0	X	N	Z	V	C
W																
Reset	0	0	1	0	0	1	1	1	0	0	0	—	—	—	—	—

Figure 2-8. Status Register (SR)

Table 2-3. SR Field Descriptions

Field	Description
15 T	Trace enable. When set, the processor performs a trace exception after every instruction.
14	Reserved, must be cleared.
13 S	Supervisor/user state. 0 User mode 1 Supervisor mode
12 M	Master/interrupt state. Bit is cleared by an interrupt exception and software can set it during execution of the RTE or move to SR instructions.
11	Reserved, must be cleared.
10–8 I	Interrupt level mask. Defines current interrupt level. Interrupt requests are inhibited for all priority levels less than or equal to current level, except edge-sensitive level 7 requests, which cannot be masked.
7–0 CCR	Refer to Section 2.2.4, “Condition Code Register (CCR)”.

2.2.10 Memory Base Address Registers (RAMBAR, FLASHBAR)

The 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. 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 5.3.1, “SRAM Base Address Register (RAMBAR)” and Section 6.3.2, “Flash Base Address Register (FLASHBAR)”.

2.3 Functional Description

2.3.1 Version 2 ColdFire Microarchitecture

From the block diagram in Figure 2-1, the non-Harvard architecture of the processor is readily apparent. The processor interfaces to the local memory subsystem via a single 32-bit address and two unidirectional 32-bit data buses. This structure minimizes the core size without compromising performance to a large degree.

A more detailed view of the hardware structure within the two pipelines is presented in Figure 2-9 and Figure 2-10 below. In these diagrams, the internal structure of the instruction fetch and operand execution pipelines is shown:

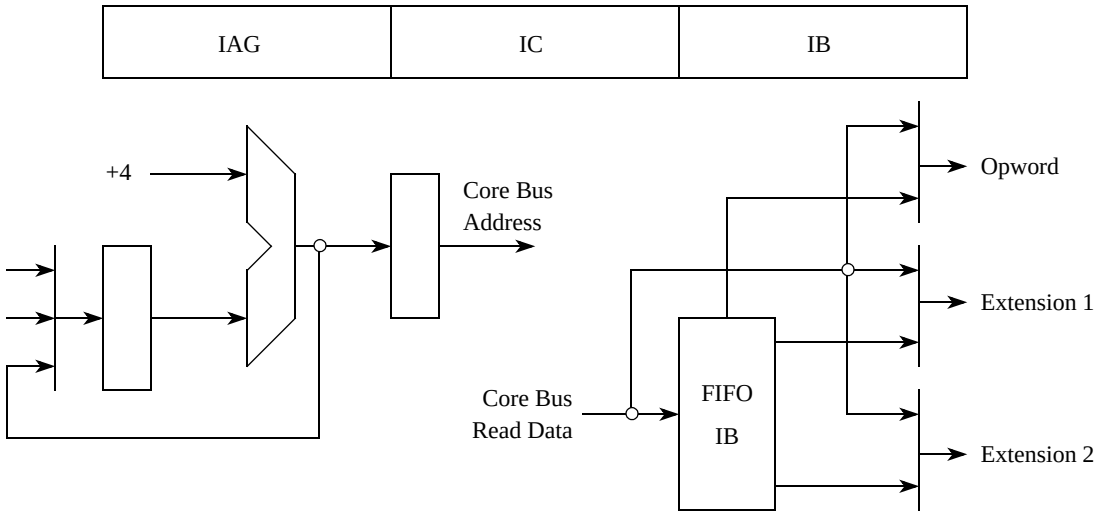


Figure 2-9. Version 2 ColdFire Processor Instruction Fetch Pipeline Diagram

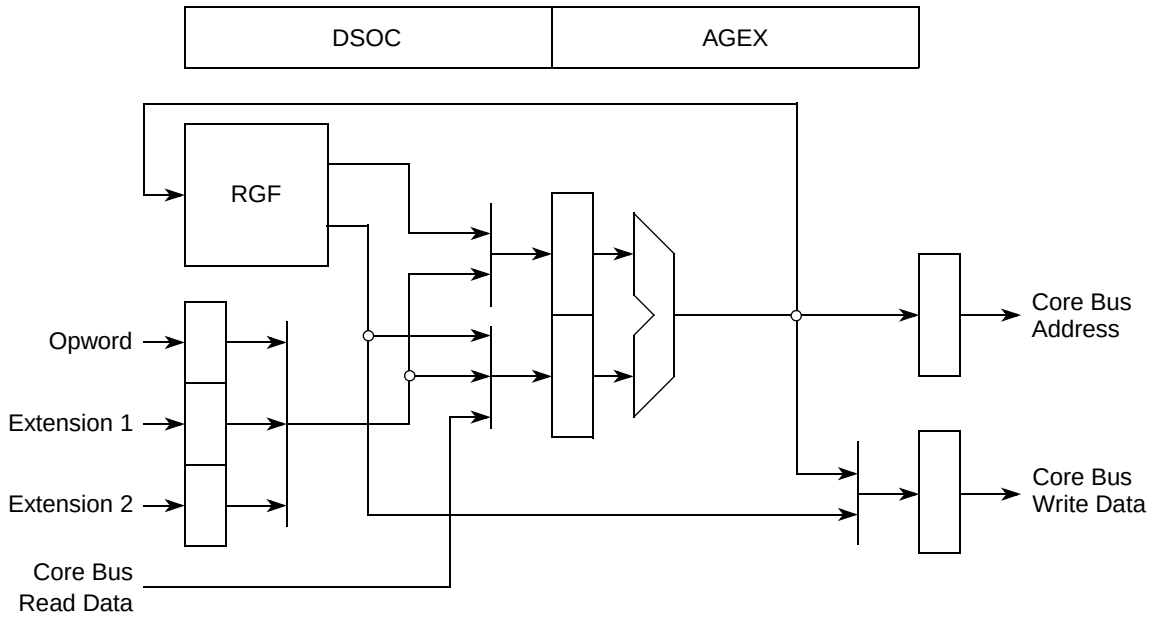


Figure 2-10. Version 2 ColdFire Processor Operand Execution Pipeline Diagram

The instruction fetch pipeline prefetches instructions from local memory using a two-stage structure. For sequential prefetches, the next instruction address is generated by adding four to the last prefetch address. This function is performed during the IAG stage and the resulting prefetch address gated onto the core bus (if there are no pending operand memory accesses assigned a higher priority). After the prefetch address is driven onto the core bus, the instruction fetch cycle accesses the appropriate local memory and returns the instruction read data back to the IFP during the cycle. If the accessed data is not present in a local memory (e.g., an instruction cache miss, or an external access cycle is required), the IFP is stalled in the IC stage until the referenced data is available. As the prefetch data arrives in the IFP, it can be loaded into the FIFO instruction buffer or gated directly into the OEP.

The V2 design uses a simple static conditional branch prediction algorithm (forward-assumed as not-taken, backward-assumed as taken), and all change-of-flow operations are calculated by the OEP and the target instruction address fed back to the IFP.

The IFP and OEP are decoupled by the FIFO instruction buffer, allowing instruction prefetching to occur with the available core bus bandwidth not used for operand memory accesses. For the V2 design, the instruction buffer contains three 32-bit locations.

Consider the operation of the OEP for three basic classes of non-branch instructions:

- Register-to-register:

  ```
  op    Ry,Rx
  ```

- Embedded load:

  ```
  op    <mem>y,Rx
  ```

- Register-to-memory (store)

  ```
  move    Ry,<mem>x
  ```

For simple register-to-register instructions, the first stage of the OEP performs the instruction decode and fetching of the required register operands (OC) from the dual-ported register file, while the actual

instruction execution is performed in the second stage (EX) in one of the execute engines (e.g., ALU, barrel shifter, divider, EMAC). There are no operand memory accesses associated with this class of instructions, and the execution time is typically a single machine cycle. See Figure 2-11.

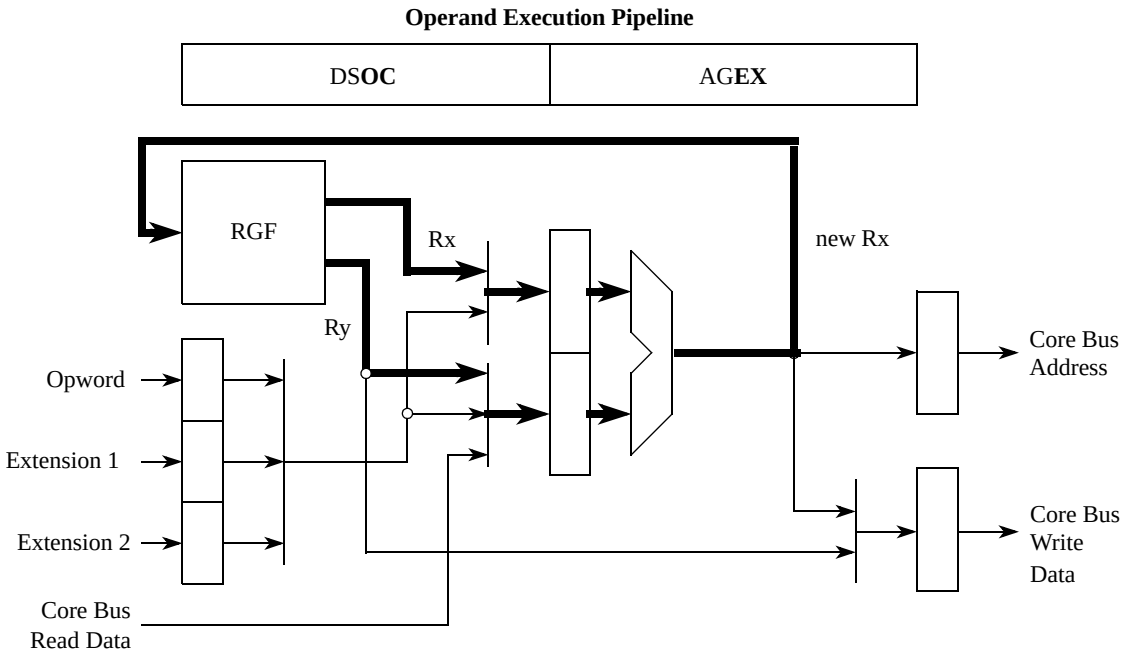


Figure 2-11. V2 OEP Register-to-Register

For memory-to-register (embedded-load) instructions, the instruction is effectively staged through the OEP twice with a basic execution time of three cycles. First, the instruction is decoded and the components of the operand address (base register from the RGF and displacement) are selected (DS). Second, the operand effective address is generated using the ALU execute engine (AG). Third, the memory read operand is fetched from the core bus, while any required register operand is simultaneously fetched (OC) from the RGF. Finally, in the fourth cycle, the instruction is executed (EX). The heavily-used 32-bit load instruction (`move.l <mem>y,Rx`) is optimized to support a two-cycle execution time. The following example in Figure 2-12 shows an effective address of the form <ea>y = (d16,Ay), i.e., a 16-bit signed displacement added to a base register Ay.

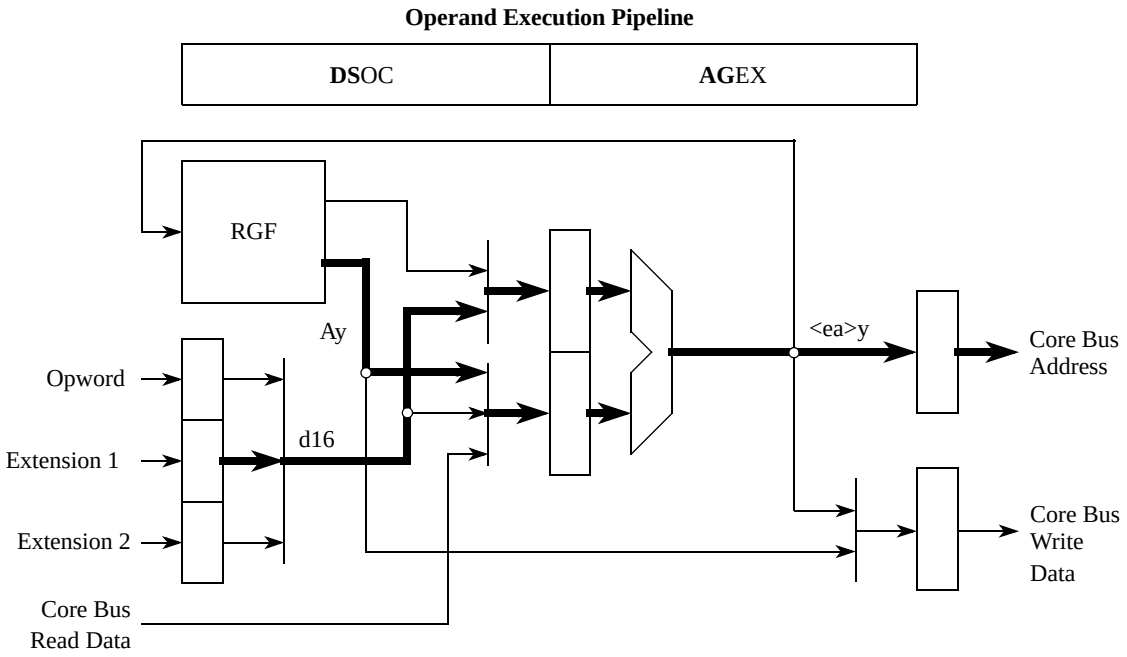


Figure 2-12. V2 OEP Embedded-Load Part 1

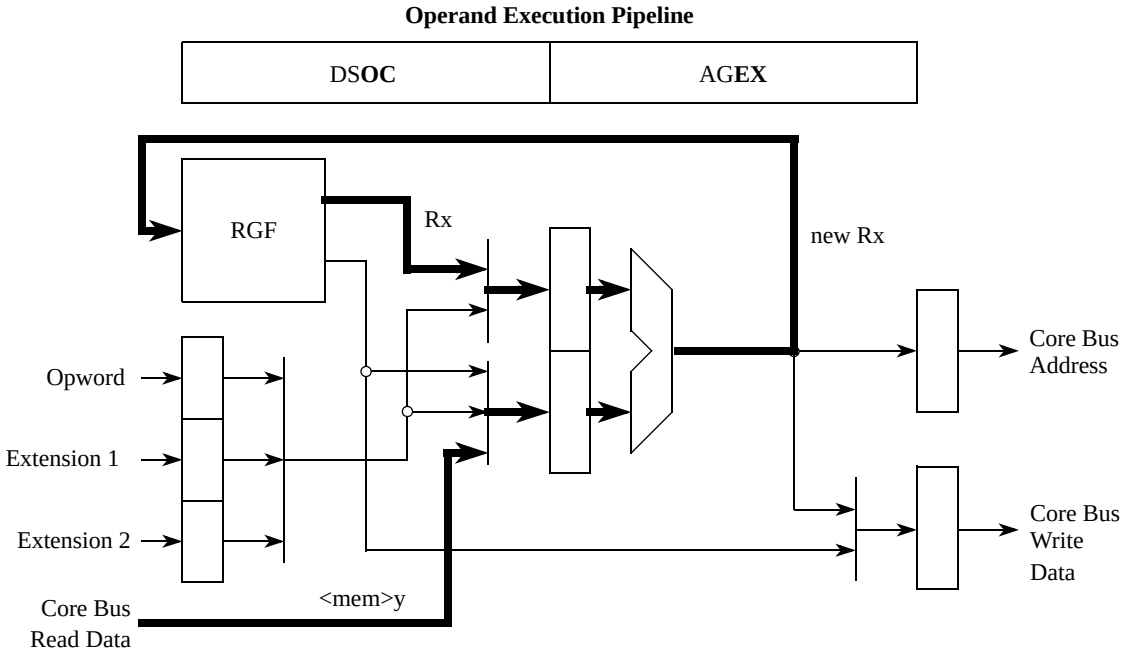


Figure 2-13. V2 OEP Embedded-Load Part 2

For register-to-memory (store) operations, the stage functions (DS/OC, AG/EX) are effectively performed simultaneously allowing single-cycle execution. See Figure 2-14 where the effective address is of the form <ea>x = (d16,Ax), i.e., a 16-bit signed displacement added to a base register Ax.

For read-modify-write instructions, the pipeline effectively combines an embedded-load with a store operation for a three-cycle execution time.

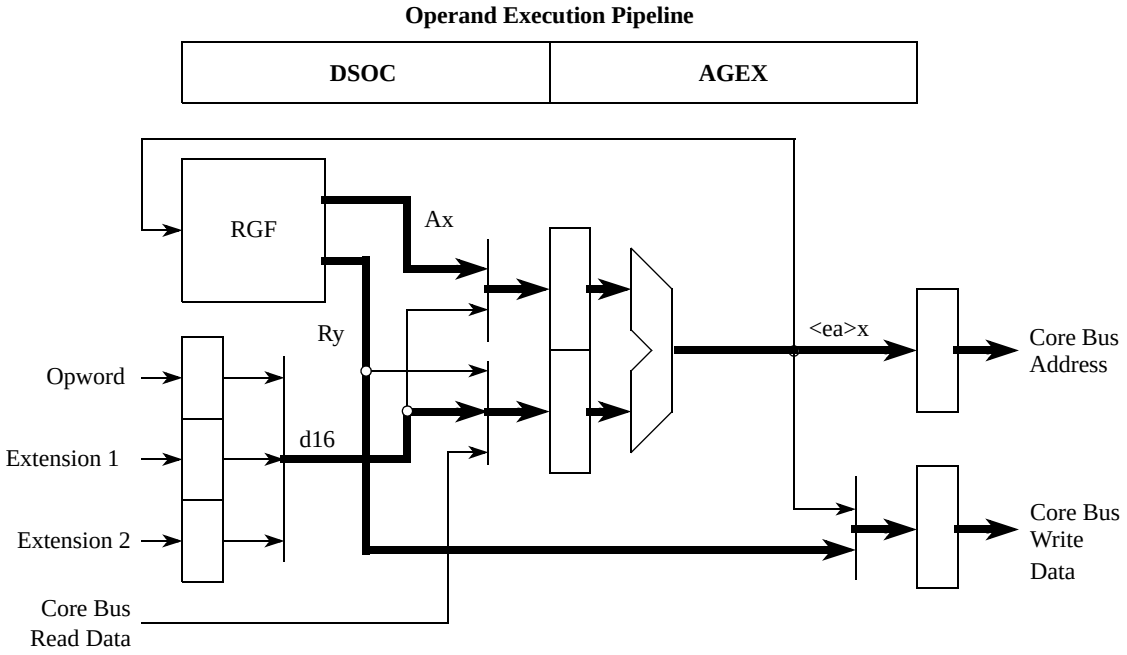


Figure 2-14. V2 OEP Register-to-Memory

The pipeline timing diagrams of Figure 2-15 depict the execution templates for these three classes of instructions. In these diagrams, the x-axis represents time, and the various instruction operations are shown progressing down the operand execution pipeline.

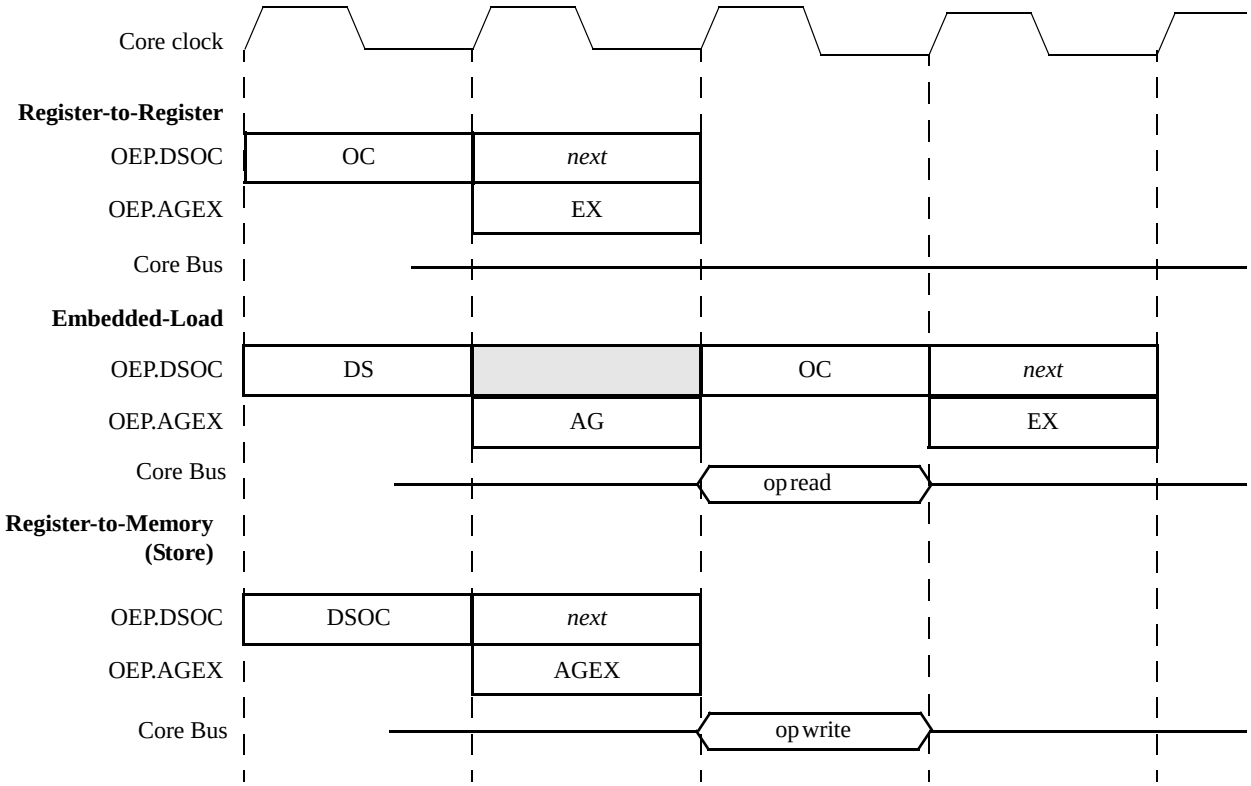


Figure 2-15. V2 OEP Pipeline Execution Templates

2.3.2 Instruction Set Architecture (ISA_A+)

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

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

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

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

Table 2-4 summarizes the instructions added to revision ISA_A to form revision ISA_A+. For more details see the *ColdFire Family Programmer's Reference Manual*.

Table 2-4. Instruction Enhancements over Revision ISA_A

Instruction	Description
BITREV	The contents of the destination data register are bit-reversed; new Dn[31] equals old Dn[0], new Dn[30] equals old Dn[1],..., new Dn[0] equals old Dn[31].
BYTEREV	The contents of the destination data register are byte-reversed; new Dn[31:24] equals old Dn[7:0],..., new Dn[7:0] equals old Dn[31:24].
FF1	The data register, Dn, is scanned, beginning from the most-significant bit (Dn[31]) and ending with the least-significant bit (Dn[0]), searching for the first set bit. The data register is then loaded with the offset count from bit 31 where the first set bit appears.
Move from USP	USP → Destination register
Move to USP	Source register → USP
STLDSR	Pushes the contents of the status register onto the stack and then reloads the status register with the immediate data value.

2.3.3 Exception Processing Overview

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

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

All ColdFire processors use an instruction restart exception model. However, Version 2 ColdFire processors require more software support to recover from certain access errors. See Section 2.3.4.1, "Access Error Exception" for details.

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

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

2. The processor determines the exception vector number. For all faults except interrupts, the processor performs this calculation based on exception type. For interrupts, the processor performs an interrupt-acknowledge (IACK) bus cycle to obtain the vector number from the interrupt controller. The IACK cycle is mapped to special locations within the interrupt controller's address space with the interrupt level encoded in the address.

3. The processor saves the current context by creating an exception stack frame on the system stack. The exception stack frame is created at a 0-modulo-4 address on top of the system stack pointed to by the supervisor stack pointer (SSP). As shown in Figure 2-16, the processor uses a simplified fixed-length stack frame for all exceptions. The exception type determines whether the program counter placed in the exception stack frame defines the location of the faulting instruction (fault) or the address of the next instruction to be executed (next).
4. The processor calculates the address of the first instruction of the exception handler. By definition, the exception vector table is aligned on a 1 MB boundary. This instruction address is generated by fetching an exception vector from the table located at the address defined in the vector base register. The index into the exception table is calculated as (4 × vector number). After the exception vector has been fetched, the vector contents determine the address of the first instruction of the desired handler. After the instruction fetch for the first opcode of the handler has initiated, exception processing terminates and normal instruction processing continues in the handler.

All ColdFire processors support a 1024-byte vector table aligned on any 1 Mbyte address boundary (see Table 2-5).

The table contains 256 exception vectors; the first 64 are defined for the core and the remaining 192 are device-specific peripheral interrupt vectors. See Chapter 10, “Interrupt Controller Modules” for details on the device-specific interrupt sources.

Table 2-5. Exception Vector Assignments

Vector Number(s)	Vector Offset (Hex)	Stacked Program Counter	Assignment
0	0x000	—	Initial supervisor stack pointer
1	0x004	—	Initial program counter
2	0x008	Fault	Access error
3	0x00C	Fault	Address error
4	0x010	Fault	Illegal instruction
5	0x014	Fault	Divide by zero
6–7	0x018–0x01C	—	Reserved
8	0x020	Fault	Privilege violation
9	0x024	Next	Trace
10	0x028	Fault	Unimplemented line-A opcode
11	0x02C	Fault	Unimplemented line-F opcode
12	0x030	Next	Debug interrupt
13	0x034	—	Reserved
14	0x038	Fault	Format error
15–23	0x03C–0x05C	—	Reserved
24	0x060	Next	Spurious interrupt
25–31	0x064–0x07C	—	Reserved

Table 2-5. Exception Vector Assignments (continued)

Vector Number(s)	Vector Offset (Hex)	Stacked Program Counter	Assignment
32–47	0x080–0x0BC	Next	Trap # 0-15 instructions
48–63	0x0C0–0x0FC	—	Reserved
64–255	0x100–0x3FC	Next	Device-specific interrupts

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

All ColdFire processors inhibit interrupt sampling during the first instruction of all exception handlers. This allows any handler to disable interrupts effectively, if necessary, by raising the interrupt mask level contained in the status register. In addition, the ISA_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 that services multiple interrupt requests with different interrupt levels. For more details, see *ColdFire Family Programmer's Reference Manual*.

2.3.3.1 Exception Stack Frame Definition

Figure 2-16 shows exception stack frame. The first longword contains the 16-bit format/vector word (F/V) and the 16-bit status register, and the second longword contains the 32-bit program counter address.

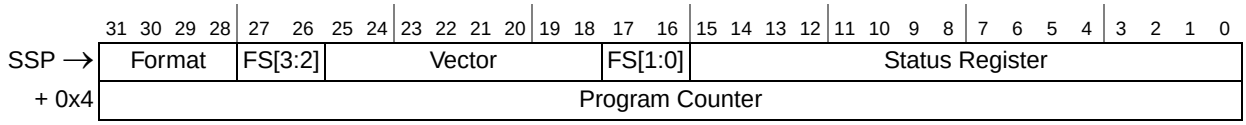


Figure 2-16. Exception Stack Frame Form

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

- A 4-bit format field at the top of the system stack is always written with a value of 4, 5, 6, or 7 by the processor, indicating a two-longword frame format. See Table 2-6.

Table 2-6. Format Field Encodings

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

- There is a 4-bit fault status field, FS[3:0], at the top of the system stack. This field is defined for access and address errors only and written as zeros for all other exceptions. See Table 2-7.

Table 2-7. Fault Status Encodings

FS[3:0]	Definition
00*xx*	Reserved
0100	Error on instruction fetch
0101	Reserved
011x	Reserved
1000	Error on operand write
1001	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 case of an interrupt. See Table 2-5.

2.3.4 Processor Exceptions

2.3.4.1 Access Error Exception

The exact processor response to an access error depends on the memory reference being performed. For an instruction fetch, the processor postpones the error reporting until the faulted reference is needed by an instruction for execution. Therefore, faults during instruction prefetches followed by a change of instruction flow do not generate an exception. When the processor attempts to execute an instruction with a faulted opword and/or extension words, the access error is signaled and the instruction aborted. For this type of exception, the programming model has not been altered by the instruction generating the access error.

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

The V2 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.

2.3.4.2 Address Error Exception

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

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

If an address error occurs on a JSR instruction, the Version 2 ColdFire processor calculates the target address then the return address is pushed onto the stack. If an address error occurs on an RTS instruction, the Version 2 ColdFire processor overwrites the faulting return PC with the address error stack frame.

2.3.4.3 Illegal Instruction Exception

The ColdFire variable-length instruction set architecture supports three instruction sizes: 16, 32, or 48 bits. The first instruction word is known as the operation word (or opword), while the optional words are known as extension word 1 and extension word 2. The opword is further subdivided into three sections: the upper four bits segment the entire ISA into 16 instruction lines, the next 6 bits define the operation mode (opmode), and the low-order 6 bits define the effective address. See Figure 2-17. The opword line definition is shown in Table 2-8.

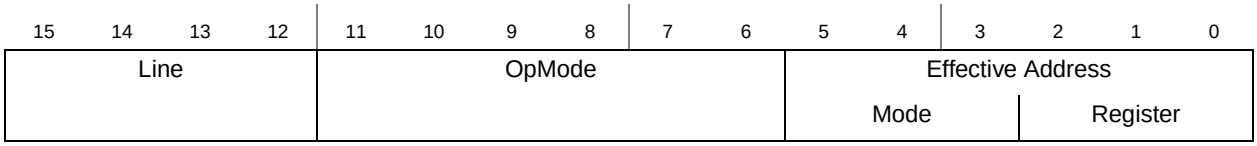


Figure 2-17. ColdFire Instruction Operation Word (Opword) Format

Table 2-8. ColdFire Opword Line Definition

Opword[Line]	Instruction Class
0x0	Bit manipulation, Arithmetic and Logical Immediate
0x1	Move Byte
0x2	Move Long
0x3	Move Word
0x4	Miscellaneous
0x5	Add (ADDQ) and Subtract Quick (SUBQ), Set according to Condition Codes (Scc)
0x6	PC-relative change-of-flow instructions Conditional (Bcc) and unconditional (BRA) branches, subroutine calls (BSR)
0x7	Move Quick (MOVEQ), Move with sign extension (MVS) and zero fill (MVZ)
0x8	Logical OR (OR)
0x9	Subtract (SUB), Subtract Extended (SUBX)

Table 2-8. ColdFire Opword Line Definition (continued)

Opword[Line]	Instruction Class
0xA	EMAC, Move 3-bit Quick (MOV3Q)
0xB	Compare (CMP), Exclusive-OR (EOR)
0xC	Logical AND (AND), Multiply Word (MUL)
0xD	Add (ADD), Add Extended (ADDX)
0xE	Arithmetic and logical shifts (ASL, ASR, LSL, LSR)
0xF	Cache Push (CPUSHL), Write DDATA (WDDATA), Write Debug (WDEBUG)

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

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

2.3.4.4 Divide-By-Zero

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

2.3.4.5 Privilege Violation

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

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

2.3.4.6 Trace Exception

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

The stop instruction has the following effects:

1. The instruction before the stop executes and then generates a trace exception. In the exception stack frame, the PC points to the stop opcode.
2. When the trace handler is exited, the stop instruction executes, loading the SR with the immediate operand from the instruction.

3. The processor then generates a trace exception. The PC in the exception stack frame points to the instruction after the stop, and the SR reflects the value loaded in the previous step.

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

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

2.3.4.7 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.

2.3.4.8 Unimplemented Line-F Opcode

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

2.3.4.9 Debug Interrupt

See Chapter 30, “Debug Support,” for a detailed explanation of this exception, which is generated in response to a hardware breakpoint register trigger. The processor does not generate an IACK cycle, but rather calculates the vector number internally (vector number 12). Additionally, SR[M,I] are unaffected by the interrupt.

2.3.4.10 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.

2.3.4.11 TRAP Instruction Exception

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

2.3.4.12 Unsupported Instruction Exception

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

All ColdFire cores record the processor hardware configuration in the D0 register immediately after the negation of $\overline{\text{RESET}}$. See Section 2.3.4.15, “Reset Exception,” for details.

2.3.4.13 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.

2.3.4.14 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 to exit this state.

2.3.4.15 Reset Exception

Asserting the reset input signal ($\overline{\text{RESET}}$) to the processor causes a reset exception. The reset exception has the highest priority of any exception; it provides for system initialization and recovery from catastrophic failure. Reset also aborts any processing in progress when the reset input is recognized. Processing cannot be recovered.

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

NOTE

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

After the processor is granted the bus, it performs two longword read-bus cycles. The first longword at address 0x0000_0000 is loaded into the supervisor stack pointer and the second longword at address 0x0000_0004 is loaded into the program counter. After the initial instruction is fetched from memory, program execution begins at the address in the PC. If an access error or address error occurs before the first instruction is executed, the processor enters the fault-on-fault state.

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

Information loaded into D0 defines the processor hardware configuration as shown in Figure 2-18.

BDM: Load: 0x080 (D0)
Store: 0x180 (D0)

Access: User read-only
BDM read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PF								VER				REV			
W																
Reset	1	1	0	0	1	1	1	1	0	0	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	0	0	0	0	ISA				DEBUG			
W																
Reset	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0

Figure 2-18. D0 Hardware Configuration Info

Table 2-9. D0 Hardware Configuration Info Field Description

Field	Description
31–24 PF	Processor family. This field is fixed to a hex value of 0xCF indicating a ColdFire core is present.
23–20 VER	ColdFire core version number. Defines the hardware microarchitecture version of ColdFire core. 0001 V1 ColdFire core 0010 V2 ColdFire core (This is the value used for this device.) 0011 V3 ColdFire core 0100 V4 ColdFire core 0101 V5 ColdFire core Else Reserved for future use
19–16 REV	Processor revision number. The default is 0b0000.
15 MAC	MAC present. This bit signals if the optional multiply-accumulate (MAC) execution engine is present in processor core. 0 MAC execute engine not present in core. (This is the value used for this device.) 1 MAC execute engine is present in core.
14 DIV	Divide present. This bit signals if the hardware divider (DIV) is present in the processor core. 0 Divide execute engine not present in core. 1 Divide execute engine is present in core. (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 processor core. 0 EMAC execute engine not present in core. 1 EMAC execute engine is present in core. (This is the value used for this device.)
12 FPU	FPU present. This bit signals if the optional floating-point (FPU) execution engine is present in 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.

Table 2-9. D0 Hardware Configuration Info Field Description (continued)

Field	Description
11–8	Reserved.
7–4 ISA	ISA revision. Defines the instruction-set architecture (ISA) revision level implemented in ColdFire processor core. 0000 ISA_A 0001 ISA_B 0010 ISA_C 1000 ISA_A+ (This is the value used for this device.) Else Reserved
3–0 DEBUG	Debug module revision number. Defines revision level of the debug module used in the ColdFire processor core. 0000 DEBUG_A 0001 DEBUG_B 0010 DEBUG_C 0011 DEBUG_D 0100 DEBUG_E 1001 DEBUG_B+ 1011 DEBUG_D+ 1111 DEBUG_D+PST Buffer Else Reserved

Information loaded into D1 defines the local memory hardware configuration as shown in the figure below.

BDM: Load: 0x1 (D1)
Store: 0x1 (D1)

Access: User read-only
BDM read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLSZ		CCAS		CCSZ				FLASHSZ					0	0	0
W																
Reset	0	0	0	1	0	0	1	1	1	0	1	1	0	0	0	0

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

Figure 2-19. D1 Hardware Configuration Info

Table 2-10. D1 Hardware Configuration Information Field Description

Field	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 CCAS	Configurable cache associativity. 00 Four-way 01 Direct mapped (This is the value used for this device) Else Reserved for future use

Table 2-10. D1 Hardware Configuration Information Field Description (continued)

Field	Description
27–24 CCSZ	Configurable cache size. Indicates the amount of instruction/data cache. The cache configuration options available are 50% instruction/50% data, 100% instruction, or 100% data, and are specified in the CACR register. 0000 No configurable cache 0001 512B configurable cache 0010 1KB configurable cache 0011 2KB configurable cache (This is the value used for this device) 0100 4KB configurable cache 0101 8KB configurable cache 0110 16KB configurable cache 0111 32KB configurable cache Else Reserved
23–19 FLASHSZ	Flash bank size. 0000-0111 No flash 1000 64-KB flash 1001 128-KB flash 1010 256-KB flash 1011 512-KB flash (This is the value used for this device) Else Reserved for future use.
18–16	Reserved
15–14 MBSZ	Bus size. Defines the width of the ColdFire master bus datapath. 00 32-bit system bus datapath (This is the value used for this device) 01 64-bit system bus datapath Else Reserved
13–8	Reserved, resets to 0b010000
7–3 SRAMSZ	SRAM bank size. 00000 No SRAM 00010 512 bytes 00100 1 KB 00110 2 KB 01000 4 KB 01010 8 KB 01100 16 KB 01110 32 KB 10000 64 KB (This is the value used for this device) 10010 128 KB Else Reserved for future use
2–0	Reserved.

2.3.5 Instruction Execution Timing

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

- C is the number of processor clock cycles, including all applicable operand fetches and writes, and all internal core cycles required to complete the instruction execution.

- R/W is the number of operand reads (R) and writes (W) required by the instruction. An operation performing a read-modify-write function is denoted as (1/1).

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

2.3.5.1 Timing Assumptions

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

1. The OEP is loaded with the opword and all required extension words at the beginning of each instruction execution. This implies that the OEP does not wait for the IFP to supply opwords and/or extension words.
2. The OEP does not experience any sequence-related pipeline stalls. The most common example of stall involves consecutive store operations, excluding the MOVEM instruction. For all STORE operations (except MOVEM), certain hardware resources within the processor are marked as busy for two clock cycles after the final decode and select/operand fetch cycle (DSOC) of the store instruction. If a subsequent STORE instruction is encountered within this 2-cycle window, it is stalled until the resource again becomes available. Thus, the maximum pipeline stall involving consecutive STORE operations is two cycles. The MOVEM instruction uses a different set of resources and this stall does not apply.
3. The OEP completes all memory accesses without any stall conditions caused by the memory itself. Thus, the timing details provided in this section assume that an infinite zero-wait state memory is attached to the processor core.
4. All operand data accesses are aligned on the same byte boundary as the operand size; for example, 16-bit operands aligned on 0-modulo-2 addresses, 32-bit operands aligned on 0-modulo-4 addresses.

   The processor core decomposes misaligned operand references into a series of aligned accesses as shown in Table 2-11.

Table 2-11. Misaligned Operand References

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

2.3.5.2 MOVE Instruction Execution Times

Table 2-12 lists execution times for MOVE.{B,W} instructions; Table 2-13 lists timings for MOVE.L.

NOTE

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

ET with {<ea> = (d16,PC)}	equals ET with {<ea> = (d16,An)}
ET with {<ea> = (d8,PC,Xi*SF)}	equals ET with {<ea> = (d8,An,Xi*SF)}

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

Table 2-12. 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)	3(1/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1))	3(1/1)
(Ay)+	3(1/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1))	3(1/1)
-(Ay)	3(1/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1))	3(1/1)
(d16,Ay)	3(1/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	—	—
(d8,Ay,Xi*SF)	4(1/0)	4(1/1)	4(1/1)	4(1/1)	—	—	—
xxx.w	3(1/0)	3(1/1)	3(1/1)	3(1/1)	—	—	—
xxx.l	3(1/0)	3(1/1)	3(1/1)	3(1/1)	—	—	—
(d16,PC)	3(1/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	—	—
(d8,PC,Xi*SF)	4(1/0)	4(1/1)	4(1/1)	4(1/1))	—	—	—
#xxx	1(0/0)	3(0/1)	3(0/1)	3(0/1)	—	—	—

Table 2-13. 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)	3(1/1)	—	—	—
xxx.w	2(1/0)	2(1/1)	2(1/1)	2(1/1)	—	—	—
xxx.l	2(1/0)	2(1/1)	2(1/1)	2(1/1)	—	—	—
(d16,PC)	2(1/0)	2(1/1)	2(1/1)	2(1/1)	2(1/1)	—	—
(d8,PC,Xi*SF)	3(1/0)	3(1/1)	3(1/1)	3(1/1)	—	—	—
#xxx	1(0/0)	2(0/1)	2(0/1)	2(0/1)	—	—	—

2.3.5.3 Standard One Operand Instruction Execution Times

Table 2-14. 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)	—
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)

2.3.5.4 Standard Two Operand Instruction Execution Times

Table 2-15. 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)	—	—	—	—	—	—	—

Table 2-15. 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
ASL.L	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
ASR.L	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
BCHG	Dy,<ea>	2(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)	—
BCHG	#imm,<ea>	2(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	—	—	—
BCLR	Dy,<ea>	2(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)	—
BCLR	#imm,<ea>	2(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	—	—	—
BSET	Dy,<ea>	2(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)	—
BSET	#imm,<ea>	2(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	—	—	—
BTST	Dy,<ea>	2(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	—
BTST	#imm,<ea>	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	—	—	—
CMP.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.L	#imm,Dx	—	—	—	—	—	—	—	1(0/0)
OR.L	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
OR.L	Dy,<ea>	—	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
ORI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
REMS.L	<ea>,Dx	≤35(0/0)	≤38(1/0)	≤38(1/0)	≤38(1/0)	≤38(1/0)	—	—	—
REMU.L	<ea>,Dx	≤35(0/0)	≤38(1/0)	≤38(1/0)	≤38(1/0)	≤38(1/0)	—	—	—
SUB.L	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
SUB.L	Dy,<ea>	—	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
SUBI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
SUBQ.L	#imm,<ea>	1(0/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
SUBX.L	Dy,Dx	1(0/0)	—	—	—	—	—	—	—

2.3.5.5 Miscellaneous Instruction Execution Times

Table 2-16. Miscellaneous Instruction Execution Times

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xn*SF)	xxx.wl	#xxx
CPUSHL	(Ax)	—	11(0/1)	—	—	—	—	—	—
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>,and list	—	1+n(n/0)	—	—	1+n(n/0)	—	—	—
MOVEM.L	and 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)
TPF		1(0/0)	—	—	—	—	—	—	—
TPF.W		1(0/0)	—	—	—	—	—	—	—
TPF.L		1(0/0)	—	—	—	—	—	—	—
UNLK	Ax	2(1/0)	—	—	—	—	—	—	—
WDDATA	<ea>	—	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	—
WDEBUG	<ea>	—	5(2/0)	—	—	5(2/0)	—	—	—

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

[2]If a MOVE.W #imm,SR instruction is executed and imm[13] equals 1, the execution time is 1(0/0).

[3]The execution time for STOP is the time required until the processor begins sampling continuously for interrupts.

[4]PEA execution times are the same for (d16,PC).

[5]PEA execution times are the same for (d8,PC,Xn*SF).

2.3.5.6 EMAC Instruction Execution Times

Table 2-17. EMAC Instruction Execution Times

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

1 Effective address of (d16,PC) not supported

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

NOTE

The execution times for moving the contents of the Racc, Raccext[01,23], MACSR, or Rmask into a destination location <ea>x shown in this table represent the best-case scenario when the store instruction is executed and there are no load or M{S}AC instructions in the EMAC execution pipeline. In general, these store operations require only a single cycle for execution, but if preceded immediately by a load, MAC, or MSAC instruction, the depth of the EMAC pipeline is exposed and the execution time is four cycles.

2.3.5.7 Branch Instruction Execution Times

Table 2-18. 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
BRA		—	—	—	—	2(0/1)	—	—	—
BSR		—	—	—	—	3(0/1)	—	—	—
JMP	<ea>	—	3(0/0)	—	—	3(0/0)	4(0/0)	3(0/0)	—
JSR	<ea>	—	3(0/1)	—	—	3(0/1)	4(0/1)	3(0/1)	—
RTE		—	—	10(2/0)	—	—	—	—	—
RTS		—	—	5(1/0)	—	—	—	—	—

Table 2-19. Bcc Instruction Execution Times

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

Chapter 3
Enhanced Multiply-Accumulate Unit (EMAC)

3.1 Introduction

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

3.1.1 Overview

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

The MAC provides functionality in three related areas:

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

The ColdFire family supports two MAC implementations with different performance levels and capabilities. The original MAC features a three-stage execution pipeline optimized for 16-bit operands, with a 16x16 multiply array and a single 32-bit accumulator. The EMAC features a four-stage pipeline optimized for 32-bit operands, with a fully pipelined 32 × 32 multiply array and four 48-bit accumulators.

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

EMAC improvements target three primary areas:

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

The three areas of functionality are addressed in detail in following sections. The logic required to support this functionality is contained in a MAC module (Figure 3-1).

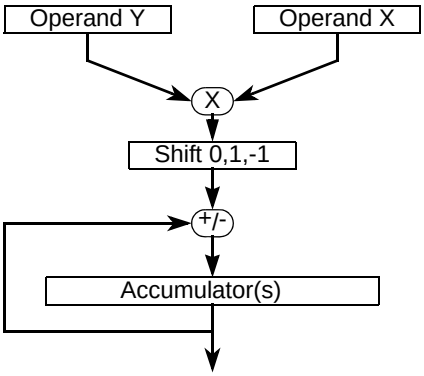


Figure 3-1. Multiply-Accumulate Functionality Diagram

3.1.1.1 Introduction to the MAC

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

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

Consider a typical filtering operation where the filter is defined as in Equation 3-1.

$$y(i) = \sum_{k=1}^{N-1} a(k)y(i-k) + \sum_{k=0}^{N-1} b(k)x(i-k) \qquad \textit{Eqn. 3-1}$$

Here, the output y(i) is determined by past output values and past input values. This is the general form of an infinite impulse response (IIR) filter. A finite impulse response (FIR) filter can be obtained by setting coefficients a(k) to zero. In either case, the operations involved in computing such a filter are multiplies and product summing. To show this point, reduce Equation 3-1 to a simple, four-tap FIR filter, shown in Equation 3-2, in which the accumulated sum is a past data values and coefficients sum.

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

3.2 Memory Map/Register Definition

The following table and sections explain the MAC registers:

Table 3-1. EMAC Memory Map

BDM[1]	Register	Width (bits)	Access	Reset Value	Section/Page
0x804	MAC Status Register (MACSR)	32	R/W	0x0000_0000	3.2.1/3-3
0x805	MAC Address Mask Register (MASK)	32	R/W	0xFFFF_FFFF	3.2.2/3-5
0x806	MAC Accumulator 0 (ACC0)	32	R/W	Undefined	3.2.3/3-6
0x807	MAC Accumulator 0,1 Extension Bytes (ACCext01)	32	R/W	Undefined	3.2.4/3-7
0x808	MAC Accumulator 2,3 Extension Bytes (ACCext23)	32	R/W	Undefined	3.2.4/3-7
0x809	MAC Accumulator 1 (ACC1)	32	R/W	Undefined	3.2.3/3-6
0x80A	MAC Accumulator 2 (ACC2)	32	R/W	Undefined	3.2.3/3-6
0x80B	MAC Accumulator 3 (ACC3)	32	R/W	Undefined	3.2.3/3-6

1 The values listed in this column represent the Rc field used when accessing the core registers via the BDM port. For more information see Chapter 43, “Debug Module.”

3.2.1 MAC Status Register (MACSR)

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

BDM: 0x804 (MACSR)

Access: Supervisor read/write
BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PAV*n*				OMC	S/U	F/I	R/T	N	Z	V	EV
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-2. MAC Status Register (MACSR)

Table 3-2. MACSR Field Descriptions

Field	Description
31–12	Reserved, must be cleared.
11–8 PAV*n*	Product/accumulation overflow flags. Contains four flags, one per accumulator, that indicate if past MAC or MSAC instructions generated an overflow during product calculation or the 48-bit accumulation. When a MAC or MSAC instruction is executed, the PAV*n* flag associated with the destination accumulator forms the general overflow flag, MACSR[V]. Once set, each flag remains set until V is cleared by a `move.l, MACSR` instruction or the accumulator is loaded directly. Bit 11: Accumulator 3 ... Bit 8: Accumulator 0

Table 3-2. MACSR Field Descriptions (continued)

Field	Description
7 OMC	Overflow saturation mode. Enables or disables saturation mode on overflow. If set, the accumulator is set to the appropriate constant (see S/U field description) on any operation that overflows the accumulator. After saturation, the accumulator remains unaffected by any other MAC or MSAC instructions until the overflow bit is cleared or the accumulator is directly loaded.
6 S/U	Signed/unsigned operations. **In integer mode:** S/U determines whether operations performed are signed or unsigned. It also determines the accumulator value during saturation, if enabled. 0 Signed numbers. On overflow, if OMC is enabled, an accumulator saturates to the most positive (0x7FFF_FFFF) or the most negative (0x8000_0000) number, depending on the instruction and the product value that overflowed. 1 Unsigned numbers. On overflow, if OMC is enabled, an accumulator saturates to the smallest value (0x0000_0000) or the largest value (0xFFFF_FFFF), depending on the instruction. **In fractional mode:** S/U controls rounding while storing an accumulator to a general-purpose register. 0 Move accumulator without rounding to a 16-bit value. Accumulator is moved to a general-purpose register as a 32-bit value. 1 The accumulator is rounded to a 16-bit value using the round-to-nearest (even) method when moved to a general-purpose register. See Section 3.3.1.1, “Rounding”. The resulting 16-bit value is stored in the lower word of the destination register. The upper word is zero-filled. This rounding procedure does not affect the accumulator value.
5 F/I	Fractional/integer mode. Determines whether input operands are treated as fractions or integers. 0 Integers can be represented in signed or unsigned notation, depending on the value of S/U. 1 Fractions are represented in signed, fixed-point, two’s complement notation. Values range from -1 to $1 - 2^{-15}$ for 16-bit fractions and -1 to $1 - 2^{-31}$ for 32-bit fractions. See Section 3.3.4, “Data Representation.”
4 R/T	Round/truncate mode. Controls rounding procedure for `move.l ACCx,Rx`, or MSAC.L instructions when in fractional mode. 0 Truncate. The product’s lsbs are dropped before it is combined with the accumulator. Additionally, when a store accumulator instruction is executed (`move.l ACCx,Rx`), the 8 lsbs of the 48-bit accumulator logic are truncated. 1 Round-to-nearest (even). The 64-bit product of two 32-bit, fractional operands is rounded to the nearest 40-bit value. If the low-order 24 bits equal 0x80_0000, the upper 40 bits are rounded to the nearest even (lsb = 0) value. See Section 3.3.1.1, “Rounding”. Additionally, when a store accumulator instruction is executed (`move.l ACCx,Rx`), the lsbs of the 48-bit accumulator logic round the resulting 16- or 32-bit value. If MACSR[S/U] is cleared and MACSR[R/T] is set, the low-order 8 bits are used to round the resulting 32-bit fraction. If MACSR[S/U] is set, the low-order 24 bits are used to round the resulting 16-bit fraction.
3 N	Negative. Set if the msb of the result is set, otherwise cleared. N is affected only by MAC, MSAC, and load operations; it is not affected by MULS and MULU instructions.
2 Z	Zero. Set if the result equals zero, otherwise cleared. This bit is affected only by MAC, MSAC, and load operations; it is not affected by MULS and MULU instructions.

Table 3-2. MACSR Field Descriptions (continued)

Field	Description
1 V	Overflow. Set if an arithmetic overflow occurs on a MAC or MSAC instruction, indicating that the result cannot be represented in the limited width of the EMAC. V is set only if a product overflow occurs or the accumulation overflows the 48-bit structure. V is evaluated on each MAC or MSAC operation and uses the appropriate PAV*n* flag in the next-state V evaluation.
0 EV	Extension overflow. Signals that the last MAC or MSAC instruction overflowed the 32 lsbs in integer mode or the 40 lsbs in fractional mode of the destination accumulator. However, the result remains accurately represented in the combined 48-bit accumulator structure. Although an overflow has occurred, the correct result, sign, and magnitude are contained in the 48-bit accumulator. Subsequent MAC or MSAC operations may return the accumulator to a valid 32/40-bit result.

Table 3-3 summarizes the interaction of the MACSR[S/U,F/I,R/T] control bits.

Table 3-3. Summary of S/U, F/I, and R/T Control Bits

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

3.2.2 Mask Register (MASK)

The 32-bit MASK implements the low-order 16 bits to minimize the alignment complications involved with loading and storing only 16 bits. When the MASK is loaded, the low-order 16 bits of the source operand are actually loaded into the register. When it is stored, the upper 16 bits are all forced to ones.

This register performs a simple AND with the operand address for MAC instructions. The processor calculates the normal operand address and, if enabled, that address is then ANDed with {0xFFFF, MASK[15:0]} to form the final address. Therefore, with certain MASK bits cleared, the operand address can be constrained to a certain memory region. This is used primarily to implement circular queues with the (An)+ addressing mode.

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

```
mac.sz  Ry,RxSF,<ea>yand ,Rw
```

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

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

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

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

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

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

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

BDM: 0x805 (MASK)

Access: User read/write
BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	MASK															
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 3-3. Mask Register (MASK)

Table 3-4. MASK Field Descriptions

Field	Description
31–16	Reserved, must be set.
15–0 MASK	Performs a simple AND with the operand address for MAC instructions.

3.2.3 Accumulator Registers (ACC0–3)

The accumulator registers store 32-bits of the MAC operation result. The accumulator extension registers form the entire 48-bit result.

BDM: 0x806 (ACC0)
0x809 (ACC1)
0x80A (ACC2)
0x80B (ACC3)

Access: User read/write
BDM read/write

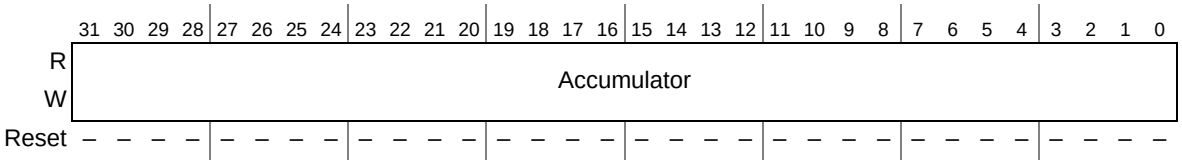


Figure 3-4. Accumulator Registers (ACC0–3)

Table 3-5. ACC0–3 Field Descriptions

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

3.2.4 Accumulator Extension Registers (ACCext01, ACCext23)

Each pair of 8-bit accumulator extension fields are concatenated with the corresponding 32-bit accumulator register to form the 48-bit accumulator. For more information, see Section 3.3, “Functional Description.”

BDM: 0x807 (ACCext01)

Access: User read/write
BDM read/write

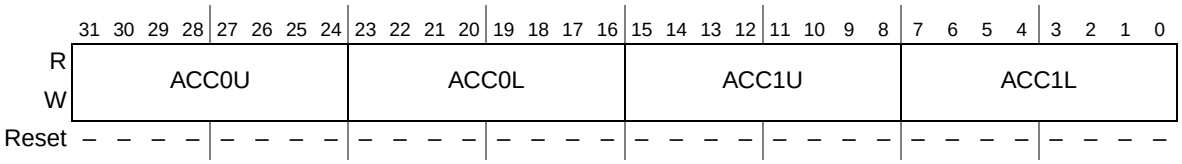


Figure 3-5. Accumulator Extension Register (ACCext01)

Table 3-6. ACCext01 Field Descriptions

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

BDM: 0x808 (ACCext23)

Access: User read/write
BDM read/write

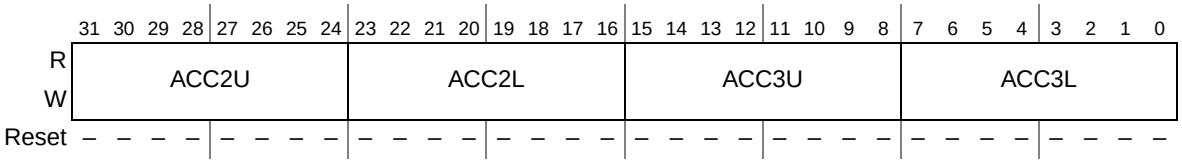


Figure 3-6. Accumulator Extension Register (ACCext23)

Table 3-7. ACCext23 Field Descriptions

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

3.3 Functional Description

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

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

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

The EMAC is optimized for single-cycle, pipelined 32 × 32 multiplications. For word- and longword-sized integer input operands, the low-order 40 bits of the product are formed and used with the destination accumulator. For fractional operands, the entire 64-bit product is calculated and truncated or rounded to the most-significant 40-bit result using the round-to-nearest (even) method before it is combined with the destination accumulator.

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

Figure 3-7 and Figure 3-8 show relative alignment of input operands, the full 64-bit product, the resulting 40-bit product used for accumulation, and 48-bit accumulator formats.

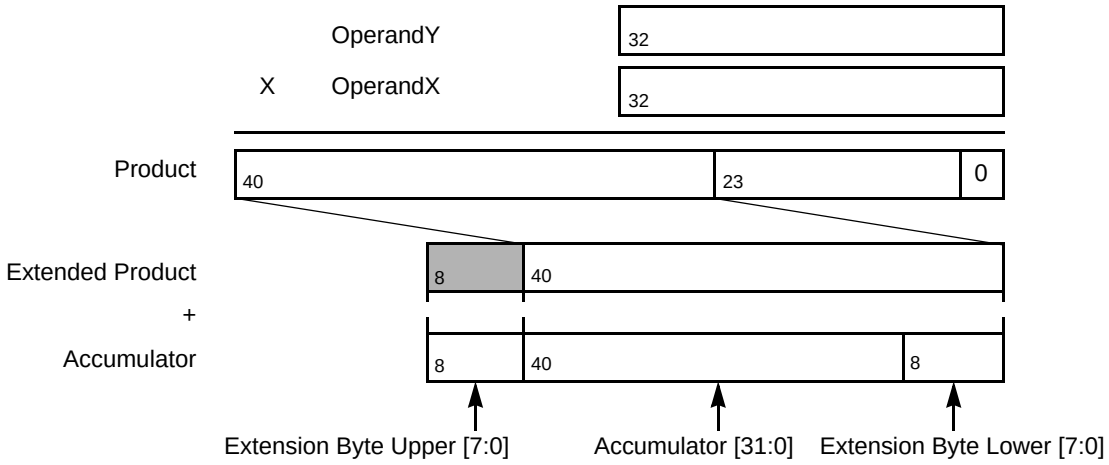


Figure 3-7. Fractional Alignment

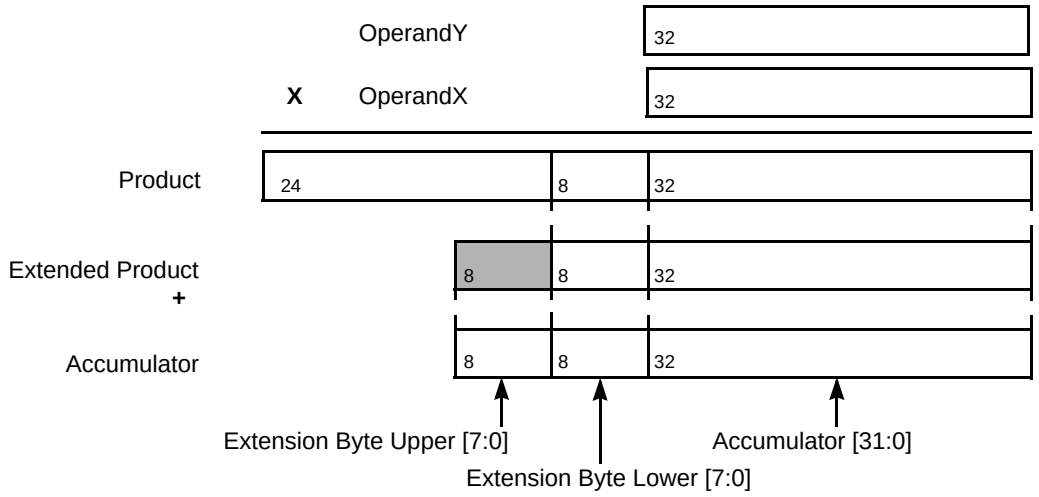


Figure 3-8. Signed and Unsigned Integer Alignment

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

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

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

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

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

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

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

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

3.3.1 Fractional Operation Mode

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

3.3.1.1 Rounding

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

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

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

- If R0.L is less than 0x8000, the result is truncated to the value of R0.U.
- If R0.L is greater than 0x8000, the upper word is incremented (rounded up).

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

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

The rounding algorithm is summarized in the following pseudocode:

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

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

3.3.1.2 Saving and Restoring the EMAC Programming Model

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

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

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

```
EMAC_state_save:
        move.l  macsr,d7          ; save the macsr
        clr.l   d0                ; zero the register to ...
        move.l  d0,macsr          ; disable rounding in the macsr
        move.l  acc0,d0           ; save the accumulators
        move.l  acc1,d1
        move.l  acc2,d2
        move.l  acc3,d3
        move.l  accext01,d4       ; save the accumulator extensions
        move.l  accext23,d5
        move.l  mask,d6           ; save the address mask
        movem.l #0x00ff,(a7)      ; move the state to memory
```

This code performs the EMAC state restore:

```
EMAC_state_restore:
```

```
movem.l (a7),#0x00ff      ; restore the state from memory
move.l  #0,macsr          ; disable rounding in the macsr
move.l  d0,acc0           ; restore the accumulators
move.l  d1,acc1
move.l  d2,acc2
move.l  d3,acc3
move.l  d4,accext01       ; restore the accumulator extensions
move.l  d5,accext23
move.l  d6,mask           ; restore the address mask
move.l  d7,macsr          ; restore the macsr
```

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

3.3.1.3 MULS/MULU

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

3.3.1.4 Scale Factor in MAC or MSAC Instructions

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

3.3.2 EMAC Instruction Set Summary

Table 3-8 summarizes EMAC unit instructions.

Table 3-8. EMAC Instruction Summary

Command	Mnemonic	Description
Multiply Signed	`muls <ea>y,Dx`	Multiplies two signed operands yielding a signed result
Multiply Unsigned	`mulu <ea>y,Dx`	Multiplies two unsigned operands yielding an unsigned result
Multiply Accumulate	`mac Ry,RxSF,ACCx` `msac Ry,RxSF,ACCx`	Multiplies two operands and adds/subtracts the product to/from an accumulator
Multiply Accumulate with Load	`mac Ry,Rx,<ea>y,Rw,ACCx` `msac Ry,Rx,<ea>y,Rw,ACCx`	Multiplies two operands and combines the product to an accumulator while loading a register with the memory operand
Load Accumulator	`move.l {Ry,#imm},ACCx`	Loads an accumulator with a 32-bit operand
Store Accumulator	`move.l ACCx,Rx`	Writes the contents of an accumulator to a CPU register
Copy Accumulator	`move.l ACCy,ACCx`	Copies a 48-bit accumulator
Load MACSR	`move.l {Ry,#imm},MACSR`	Writes a value to MACSR
Store MACSR	`move.l MACSR,Rx`	Write the contents of MACSR to a CPU register
Store MACSR to CCR	`move.l MACSR,CCR`	Write the contents of MACSR to the CCR
Load MAC Mask Reg	`move.l {Ry,#imm},MASK`	Writes a value to the MASK register
Store MAC Mask Reg	`move.l MASK,Rx`	Writes the contents of the MASK to a CPU register
Load Accumulator Extensions 01	`move.l {Ry,#imm},ACCext01`	Loads the accumulator 0,1 extension bytes with a 32-bit operand

Table 3-8. EMAC Instruction Summary (continued)

Command	Mnemonic	Description
Load Accumulator Extensions 23	`move.l {Ry,#imm},ACCext23`	Loads the accumulator 2,3 extension bytes with a 32-bit operand
Store Accumulator Extensions 01	`move.l ACCext01,Rx`	Writes the contents of accumulator 0,1 extension bytes into a CPU register
Store Accumulator Extensions 23	`move.l ACCext23,Rx`	Writes the contents of accumulator 2,3 extension bytes into a CPU register

3.3.3 EMAC Instruction Execution Times

The instruction execution times for the EMAC can be found in Section 2.3.5.6, “EMAC Instruction Execution Times”.

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

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

The MOVE.L instruction that stores the accumulator to an integer register (Rz) stalls until the program-visible copy of the accumulator is available. Figure 3-9 shows EMAC timing.

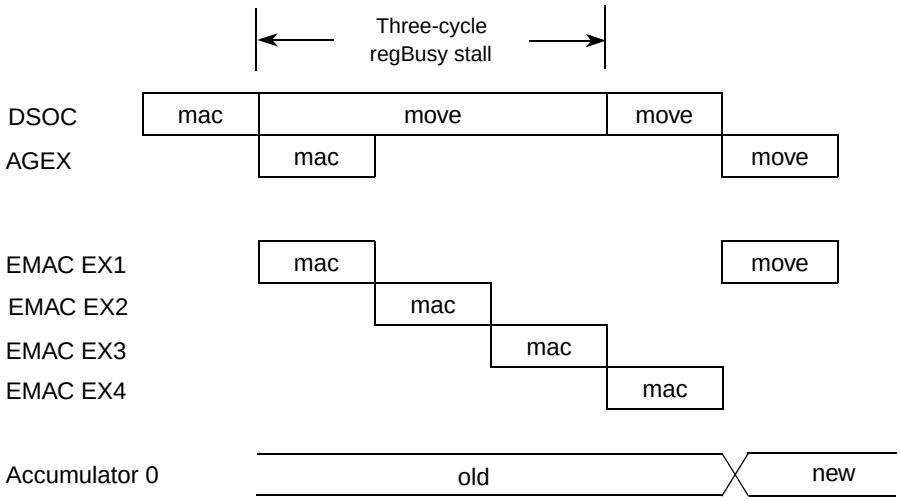


Figure 3-9. EMAC-Specific OEP Sequence Stall

In Figure 3-9, the OEP stalls the store-accumulator instruction for three cycles: the EMAC pipleline depth minus 1. The minus 1 factor is needed because the OEP and EMAC pipelines overlap by a cycle, the AGEX stage. As the store-accumulator instruction reaches the AGEX stage where the operation is performed, the recently updated accumulator 0 value is available.

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

3.3.4 Data Representation

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

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

$$\text{value} = -(1 \cdot a_{N-1}) + \sum_{i=0}^{N-2} 2^{-(i+1-N)} \cdot ai \qquad \textit{Eqn. 3-3}$$

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

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

3.3.5 MAC Opcodes

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

Remember the following:

- Unless otherwise noted, the value of MACSR[N,Z] is based on the result of the final operation that involves the product and the accumulator.
- The overflow (V) flag is managed differently. It is set if the complete product cannot be represented as a 40-bit value (this applies to 32 × 32 integer operations only) or if the combination of the product with an accumulator cannot be represented in the given number of bits. The EMAC design includes an additional product/accumulation overflow bit for each accumulator that are treated as sticky indicators and are used to calculate the V bit on each MAC or MSAC instruction. See Section 3.2.1, "MAC Status Register (MACSR)".
- For the MAC design, the assembler syntax of the MAC (multiply and add to accumulator) and MSAC (multiply and subtract from accumulator) instructions does not include a reference to the single accumulator. For the EMAC, assemblers support this syntax and no explicit reference to an accumulator is interpreted as a reference to ACC0. Assemblers also support syntaxes where the destination accumulator is explicitly defined.

- The optional 1-bit shift of the product is specified using the notation {<< | >>} SF, where <<1 indicates a left shift and >>1 indicates a right shift. The shift is performed before the product is added to or subtracted from the accumulator. Without this operator, the product is not shifted. If the EMAC is in fractional mode (MACSR[F/I] is set), SF is ignored and no shift is performed. Because a product can overflow, the following guidelines are implemented:
  - For unsigned word and longword operations, a zero is shifted into the product on right shifts.
  - For signed, word operations, the sign bit is shifted into the product on right shifts unless the product is zero. For signed, longword operations, the sign bit is shifted into the product unless an overflow occurs or the product is zero, in which case a zero is shifted in.
  - For all left shifts, a zero is inserted into the lsb position.

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

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

               /* perform the multiply */
               product[63:0] = operandY[31:0] * operandX[31:0]

               /* check for product overflow */
      if ((product[63:39] != 0x0000_00_0) and and  (product[63:39] != 0xffff_ff_1))
                  then {        /* product overflow */
                        MACSR.PAVn = 1
                        MACSR.V = 1
                        if (inst == MSAC and and  MACSR.OMC == 1)
                           then if (product[63] == 1)
                                   then result[47:0] = 0x0000_7fff_ffff
                                   else result[47:0] = 0xffff_8000_0000
                           else if (MACSR.OMC == 1)
                                   then /* overflowed MAC,
                                           saturationMode enabled */
                                        if (product[63] == 1)
                                          then result[47:0] = 0xffff_8000_0000
                                          else result[47:0] = 0x0000_7fff_ffff
                  }
```

```
            /* sign-extend to 48 bits before performing any scaling */
                    product[47:40] = {8{product[39]}}    /* sign-extend */

            /* scale product before combining with accumulator */
            switch (SF)     /* 2-bit scale factor */
            {
                case 0:     /* no scaling specified */
                   break;
                case 1:     /* SF = "<< 1" */
                   product[40:0] = {product[39:0], 0}
                   break;
                case 2:     /* reserved encoding */
                   break;
                case 3:     /* SF = ">> 1" */
                   product[39:0] = {product[39], product[39:1]}
                   break;
            }

            if (MACSR.PAVn == 0)
               then {if (inst == MSAC)
                        then result[47:0] = ACCx[47:0] - product[47:0]
                        else result[47:0] = ACCx[47:0] + product[47:0]
               }

            /* check for accumulation overflow */
            if (accumulationOverflow == 1)
               then {MACSR.PAVn = 1
                     MACSR.V = 1
                     if (MACSR.OMC == 1)
                        then /* accumulation overflow,
                                saturationMode enabled */
                             if (result[47] == 1)
                                then result[47:0] = 0x0000_7fff_ffff
                                else result[47:0] = 0xffff_8000_0000
               }
            /* transfer the result to the accumulator */
            ACCx[47:0] = result[47:0]
      }
      MACSR.V = MACSR.PAVn
      MACSR.N = ACCx[47]
      if (ACCx[47:0] == 0x0000_0000_0000)
         then MACSR.Z = 1
         else MACSR.Z = 0
      if ((ACCx[47:31] == 0x0000_0) || (ACCx[47:31] == 0xffff_1))
         then MACSR.EV = 0
         else MACSR.EV = 1
break;
   case 1,3:              /* signed fractionals */
   if (MACSR.OMC == 0 || MACSR.PAVn == 0)
      then {
            MACSR.PAVn = 0
            if (sz == word)
               then {if (U/Ly == 1)
                        then operandY[31:0] = {Ry[31:16], 0x0000}
                        else operandY[31:0] = {Ry[15:0],  0x0000}
                     if (U/Lx == 1)
```

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

```
            then operandX[31:0] = {0x0000, Rx[31:16]}
            else operandX[31:0] = {0x0000, Rx[15:0]}
   }
   else {operandY[31:0] = Ry[31:0]
         operandX[31:0] = Rx[31:0]
   }

/* perform the multiply */
product[63:0] = operandY[31:0] * operandX[31:0]

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

/* zero-fill to 48 bits before performing any scaling */
         product[47:40] = 0    /* zero-fill upper byte */

/* scale product before combining with accumulator */
switch (SF)     /* 2-bit scale factor */
{
    case 0:     /* no scaling specified */
       break;
    case 1:     /* SF = “<< 1” */
       product[40:0] = {product[39:0], 0}
       break;
    case 2:     /* reserved encoding */
       break;
    case 3:     /* SF = “>> 1” */
       product[39:0] = {0, product[39:1]}
       break;
}

/* combine with accumulator */
if (MACSR.PAVn == 0)
   then {if (inst == MSAC)
            then result[47:0] = ACCx[47:0] - product[47:0]
            else result[47:0] = ACCx[47:0] + product[47:0]
   }

/* check for accumulation overflow */
if (accumulationOverflow == 1)
   then {MACSR.PAVn = 1
         MACSR.V = 1
         if (inst == MSAC and and  MACSR.OMC == 1)
            then result[47:0] = 0x0000_0000_0000
            else if (MACSR.OMC == 1)
                 then /* overflowed MAC,
                         saturationMode enabled */
```

```
                                    result[47:0] = 0xffff_ffff_ffff
                  }

               /* transfer the result to the accumulator */
               ACCx[47:0] = result[47:0]
         }
         MACSR.V = MACSR.PAVn
         MACSR.N = ACCx[47]
         if (ACCx[47:0] == 0x0000_0000_0000)
            then MACSR.Z = 1
            else MACSR.Z = 0
         if (ACCx[47:32] == 0x0000)
            then MACSR.EV = 0
            else MACSR.EV = 1
   break;
}
```

Chapter 4
Cache

4.1 Introduction

This chapter describes cache operation on the ColdFire processor.

4.1.1 Features

Features include the following:

- Configurable as instruction, data, or split instruction/data cache
- 2-Kbyte direct-mapped cache
- Single-cycle access on cache hits
- Physically located on the ColdFire core's high-speed local bus
- Nonblocking design to maximize performance
- Separate instruction and data 16-Byte line-fill buffers
- Configurable instruction cache miss-fetch algorithm

4.1.2 Introduction

The cache is a direct-mapped, single-cycle memory. It may be configured as an instruction cache, a write-through data cache, or a split instruction/data cache. The cache storage is organized as 128 lines, each containing 16 bytes. The memory storage consists of a 128-entry tag array (containing addresses and a valid bit), and a data array containing 2 Kbytes, organized as 512×32 bits.

Cache configuration is controlled by bits in the cache control register (CACR), detailed later in this chapter. For the instruction or data-only configurations, only the associated instruction or data line-fill buffer is used. For the split cache configuration, one-half of the tag and storage arrays is used for an instruction cache and one-half is used for a data cache. The split cache configuration uses the instruction and the data line-fill buffers. The core’s local bus is a unified bus used for instruction and data fetches. Therefore, the cache can have only one fetch, instruction or data, active at one time.

For the instruction- or data-only configurations, the cache tag and storage arrays are accessed in parallel: fetch address bits [10:4] addressing the tag array, and fetch address bits [10:2] addressing the storage array. For the split cache configuration, the cache tag and storage arrays are accessed in parallel. The msb of the tag array address is set for instruction fetches and cleared for operand fetches; fetch address bits [9:4] provide the rest of the tag array address. The tag array outputs the address mapped to the given cache location along with the valid bit for the line. This address field is compared to bits [31:11] for instruction- or data-only configurations and to bits [31:10] for a split configuration of the fetch address from the local bus to determine if a cache hit has occurred. If the desired address is mapped into the cache memory, the

output of the storage array is driven onto the ColdFire core's local data bus, thereby completing the access in a single cycle.

The tag array maintains a single valid bit per line entry. Accordingly, only entire 16-byte lines are loaded into the cache.

The cache also contains separate 16-byte instruction and data line-fill buffers that provide temporary storage for the last line fetched in response to a cache miss. With each fetch, the contents of the associated line fill buffer are examined. Thus, each fetch address examines the tag memory array and the associated line fill buffer to see if the desired address is mapped into either hardware resource. A cache hit in the memory array or the associated line-fill buffer is serviced in a single cycle. Because the line fill buffer maintains valid bits on a longword basis, hits in the buffer can be serviced immediately without waiting for the entire line to be fetched.

If the referenced address is not contained in the memory array or the associated line-fill buffer, the cache initiates the required external fetch operation. In most situations, this is a 16-byte line-sized burst reference.

The hardware implementation is a nonblocking design, meaning the ColdFire core's local bus is released after the initial access of a miss. Thus, the cache or the SRAM module can service subsequent requests while the remainder of the line is being fetched and loaded into the fill buffer.

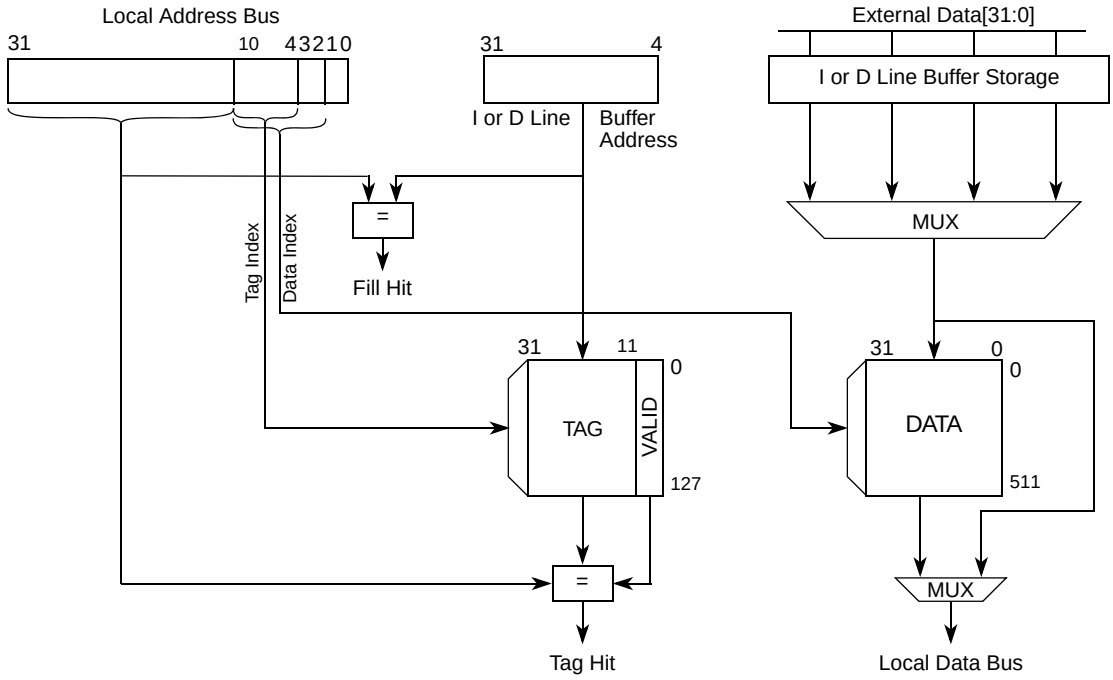


Figure 4-1. 2-Kbyte Cache Block Diagram

4.2 Memory Map/Register Definition

Three supervisor registers define the operation of the cache and local bus controller: the cache control register (CACR) and two access control registers (ACR0, ACR1). Table 4-1 below shows the memory map

of these registers. The CACR and ACRs can only be accessed in supervisor mode using the MOVEC instruction with an Rc value of 0x002, 0x004 and 0x005, respectively.

Table 4-1. Cache Memory Map

BDM[1]	Register	Width (bits)	Access[2]	Reset Value	Section/Page
0x002	Cache Control Register (CACR)	32	W	0x0000_0000	4.2.1/4-3
0x004	Access Control Register 0 (ACR0)	32	W	See Section	4.2.2/4-6
0x005	Access Control Register 1 (ACR1)	32	W	See Section	4.2.2/4-6

1 The values listed in this column represent the Rc field used when accessing the core registers via the BDM port. For more information see Chapter 30, "Debug Support."

2 Readable through debug.

4.2.1 Cache Control Register (CACR)

The CACR controls the operation of the cache. The CACR provides a set of default memory access attributes used when a reference address does not map into the spaces defined by the ACRs.

The CACR is a 32-bit, write-only supervisor control register. It is accessed in the CPU address space via the MOVEC instruction with an Rc encoding of 0x002. The CACR can be read when in background debug mode (BDM). Therefore, the register diagram, Figure 4-2, is shown as read/write. At system reset, the entire register is cleared.

BDM: 0x002 (CACR)

Access: Supervisor write-only
Debug read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CENB	0	0	CPD	CFRZ	0	0	CINV	DISI	DISD	INVI	INVD	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	CEIB	DCM	DBWE	0	0	DWP	EUSP	0	0	CLNF	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-2. Cache Control Register (CACR)

Table 4-2. CACR Field Descriptions

Field	Description
31 CENB	Cache enable. The memory array of the cache is enabled only if CENB is asserted. This bit, along with the DISI (disable instruction caching) and DISD (disable data caching) bits, control the cache configuration. 0 Cache disabled 1 Cache enabled Table 4-3 describes cache configuration.
30–29	Reserved, must be cleared.

Table 4-2. CACR Field Descriptions (continued)

Field	Description
28 CPDI	Disable CPUSHL invalidation. When the privileged CPUSHL instruction is executed, the cache entry defined by bits [10:4] of the address is invalidated if CPDI is cleared. If CPDI is set, no operation is performed. 0 Enable invalidation 1 Disable invalidation
27 CFRZ	Cache freeze. This field allows the user to freeze the contents of the cache. When CFRZ is asserted line fetches can be initiated and loaded into the line-fill buffer, but a valid cache entry can not be overwritten. If a given cache location is invalid, the contents of the line-fill buffer can be written into the memory array while CFRZ is asserted. 0 Normal Operation 1 Freeze valid cache lines
26–25	Reserved, must be cleared.
24 CINV	Cache invalidate. The cache invalidate operation is not a function of the CENB state (this operation is independent of the cache being enabled or disabled). Setting this bit forces the cache to invalidate all, half, or none of the tag array entries depending on the state of the DISI, DISD, INVI, and INVD bits. The invalidation process requires several cycles of overhead plus 128 machine cycles to clear all tag array entries and 64 cycles to clear half of the tag array entries, with a single cache entry cleared per machine cycle. The state of this bit is always read as a zero. After a hardware reset, the cache must be invalidated before it is enabled. 0 No operation 1 Invalidate all cache locations Table 4-4 describes how to set the cache invalidate all bit.
23 DISI	Disable instruction caching. When set, this bit disables instruction caching. This bit, along with the CENB (cache enable) and DISD (disable data caching) bits, control the cache configuration. See the CENB definition for a detailed description. 0 Enable instruction caching 1 Disable instruction caching Table 4-3 describes cache configuration and Table 4-4 describes how to set the cache invalidate all bit.
22 DISD	Disable data caching. When set, this bit disables data caching. This bit, along with the CENB (cache enable) and DISI (disable instruction caching) bits, control the cache configuration. See the CENB definition for a detailed description. 0 Enable data caching 1 Disable data caching Table 4-3 describes cache configuration and Table 4-4 describes how to set the cache invalidate all bit.
21 INVI	CINV instruction cache only. This bit can not be set unless the cache configuration is split (DISI and DISD cleared). For instruction or data cache configurations this bit is a don’t-care. For the split cache configuration, this bit is part of the control for the invalidate all operation. See the CINV definition for a detailed description Table 4-4 describes how to set the cache invalidate all bit.
20 INVD	CINV data cache only. This bit can not be set unless the cache configuration is split (DISI and DISD cleared). For instruction or data cache configurations this bit is a don’t-care. For the split cache configuration, this bit is part of the control for the invalidate all operation. See the CINV definition for a detailed description Table 4-4 describes how to set the cache invalidate all bit.
19–11	Reserved, must be cleared.
10 CEIB	Cache enable non-cacheable instruction bursting. Setting this bit enables the line-fill buffer to be loaded with burst transfers under control of CLNF[1:0] for non-cacheable accesses. Non-cacheable accesses are never written into the memory array. See Table 4-7. 0 Disable burst fetches on non-cacheable accesses 1 Enable burst fetches on non-cacheable accesses

Table 4-2. CACR Field Descriptions (continued)

Field	Description
9 DCM	Default cache mode. This bit defines the default cache mode. For more information on the selection of the effective memory attributes, see Section 4.3.2, “Memory Reference Attributes. 0 Caching enabled 1 Caching disabled
8 DBWE	Default buffered write enable. This bit defines the default value for enabling buffered writes. If DBWE = 0, the termination of an operand write cycle on the processor's local bus is delayed until the external bus cycle is completed. If DBWE = 1, the write cycle on the local bus is terminated immediately and the operation buffered in the bus controller. In this mode, operand write cycles are effectively decoupled between the processor's local bus and the external bus. Generally, enabled buffered writes provide higher system performance but recovery from access errors can be more difficult. For the ColdFire core, reporting access errors on operand writes is always imprecise and enabling buffered writes further decouples the write instruction and the signaling of the fault 0 Disable buffered writes 1 Enable buffered writes
7–6	Reserved, must be cleared.
5 DWP	Default write protection 0 Read and write accesses permitted 1 Only read accesses permitted
4 EUSP	Enable user stack pointer. See Section 2.2.3, “Supervisor/User Stack Pointers (A7 and OTHER_A7)”for more information on the dual stack pointer implementation. 0 Disable the processor’s use of the User Stack Pointer 1 Enable the processor’s use of the User Stack Pointer
3–2	Reserved, must be cleared.
1–0 CLNF	Cache line fill. These bits control the size of the memory request the cache issues to the bus controller for different initial instruction line access offsets. See Table 4-6 for external fetch size based on miss address and CLNF.

Table 4-3 shows the relationship between CACR[CENB, DISI, & DISD] bits and the cache configuration.

Table 4-3. Cache Configuration as Defined by CACR

CACR [CENB]	CACR [DISI]	CACR [DISD]	Configuration	Description
0	x	x	N/A	Cache is completely disabled
1	0	0	Split Instruction/ Data Cache	1 KByte direct-mapped instruction cache (uses upper half of tag and storage arrays) and 1 KByte direct-mapped write-through data cache (uses lower half of tag and storage arrays)
1	0	1	Instruction Cache	2 KByte direct-mapped instruction cache (uses all of tag and storage arrays)
1	1	0	Data Cache	2 KByte direct-mapped write-through data cache (uses all of tag and storage arrays)

Table 4-4 shows the relationship between CACR[DISI, DISD, INVI, & INVD] and setting the cache invalidate all bit (CACR[CINV]).

Table 4-4. Cache Invalidate All as Defined by CACR

CACR [DISI]	CACR [DISD]	CACR [INVI]	CACR [INVD]	Configuration	Operation
0	0	0	0	Split Instruction/ Data Cache	Invalidate all entries in 1-KByte instruction cache and 1-KByte data cache
0	0	0	1	Split Instruction/ Data Cache	Invalidate only 1 KByte data cache
0	0	1	0	Split Instruction Data Cache	Invalidate only 1 KByte instruction cache
0	0	1	1	Split Instruction/ Data Cache	No invalidate
1	0	x	x	Instruction Cache	Invalidate 2 KByte instruction cache
0	1	x	x	Data Cache	Invalidate 2 KByte data cache

4.2.2 Access Control Registers (ACR0, ACR1)

The ACRs provide a definition of memory reference attributes for two memory regions (one per ACR). This set of effective attributes is defined for every memory reference using the ACRs or the set of default attributes contained in the CACR. The ACRs are examined for every processor memory reference not mapped to the flash or SRAM memories.

The ACRs are 32-bit, write-only supervisor control register. They are accessed in the CPU address space via the MOVEC instruction with an Rc encoding of 0x004 and 0x005. The ACRs can be read when in background debug mode (BDM). Therefore, the register diagram, Figure 4-3, is shown as read/write. At system reset, both registers are disabled with ACR*n*[EN] cleared.

NOTE

IPSBAR space should not be cached. The combination of the CACR defaults and the two ACR*n* registers must define the non-cacheable attribute for this address space.

BDM: 0x004 (ACR0)
0x005 (ACR1)

Access: Supervisor write-only
BDM read/write

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

Figure 4-3. Access Control Registers (ACR*n*)

Table 4-5. ACR*n* Field Descriptions

Field	Description
31–24 AB	Address base. This 8-bit field is compared to address bits [31:24] from the processor's local bus under control of the ACR address mask. If the address matches, the attributes for the memory reference are sourced from the given ACR.
23–16 AM	Address mask. Masks any AB bit. If a bit in the AM field is set, the corresponding bit of the address field comparison is ignored.
15 EN	ACR Enable. Hardware reset clears this bit, disabling the ACR. 0 ACR disabled 1 ACR enabled
14–13 SM	Supervisor mode. Allows the given ACR to be applied to references based on operating privilege mode of the ColdFire processor. The field uses the ACR for user references only, supervisor references only, or all accesses. 00 Match if user mode 01 Match if supervisor mode 1*x* Match always—ignore user/supervisor mode
12–7	Reserved, must be cleared.
6 CM	Cache mode. 0 Caching enabled 1 Caching disabled
5 BWE	Buffered write enable. Defines the value for enabling buffered writes. If BWE is cleared, the termination of an operand write cycle on the processor's local bus is delayed until the system bus cycle is completed. Setting BWE terminates the write cycle on the local bus immediately and the operation is then buffered in the bus controller. In this mode, operand write cycles are effectively decoupled between the processor's local bus and the system bus. Generally, the enabling of buffered writes provides higher system performance but recovery from access errors may be more difficult. For the V2 ColdFire core, the reporting of access errors on operand writes is always imprecise, and enabling buffered writes simply decouples the write instruction from the signaling of the fault even more. 0 Writes are not buffered. 1 Writes are buffered.
4–3	Reserved, must be cleared.
2 WP	Write protect. Defines the write-protection attribute. If the effective memory attributes for a given access select the WP bit, an access error terminates any attempted write with this bit set. 0 Read and write accesses permitted 1 Only read accesses permitted
1–0	Reserved, must be cleared.

4.3 Functional Description

The cache is physically connected to the ColdFire core's local bus, allowing it to service all fetches from the ColdFire core and certain memory fetches initiated by the debug module. Typically, the debug module's memory references appear as supervisor data accesses but the unit can be programmed to generate user-mode accesses and/or instruction fetches. The cache processes any fetch access in the normal manner.

4.3.1 Interaction with Other Modules

Because the cache and high-speed SRAM module are connected to the ColdFire core's local data bus, certain user-defined configurations can result in simultaneous fetch processing.

If the referenced address is mapped into the SRAM module, that module services the request in a single cycle. In this case, data accessed from the cache is simply discarded and no external memory references are generated. If the address is not mapped into the SRAM space, the cache handles the request in the normal fashion.

4.3.2 Memory Reference Attributes

For every memory reference the ColdFire core or the debug module generates, a set of effective attributes is determined based on the address and the access control registers (ACRs). This set of attributes includes the cacheable/non-cacheable definition, the precise/imprecise handling of operand write, and the write-protect capability.

In particular, each address is compared to the values programmed in the ACRs. If the address matches one of the ACR values, the access attributes from that ACR are applied to the reference. If the address does not match either ACR, then the default value defined in the cache control register (CACR) is used. The specific algorithm is as follows:

```
if (address == ACR0_address including mask)
        Effective Attributes = ACR0 attributes
    else if (address == ACR1_address including mask)
            Effective Attributes = ACR1 attributes
        else Effective Attributes = CACR default attributes
```

4.3.3 Cache Coherency and Invalidation

The cache does not monitor data references for accesses to cached instructions. Therefore, software must maintain instruction cache coherency by invalidating the appropriate cache entries after modifying code segments if instructions are cached.

The cache invalidation can be performed in several ways. For the instruction- or data-only configurations, setting CACR[CINV] forces the entire cache to be marked as invalid. The invalidation operation requires 128 cycles because the cache sequences through the entire tag array, clearing a single location each cycle. For the split configuration, CACR[INVI] and CACR[INVD] can be used in addition to CACR[CINV] to clear the entire cache, only the instruction half, or only the data half. Any subsequent fetch accesses are postponed until the invalidation sequence is complete.

The privileged CPUSHL instruction can invalidate a single cache line. When this instruction is executed, the cache entry defined by bits [10:4] of the source address register is invalidated, provided CACR[CPDI] is cleared. For the split data/instruction cache configuration, software directly controls bit 10 that selects whether an instruction cache or data cache line is being accessed.

These invalidation operations can be initiated from the ColdFire core or the debug module.

4.3.4 Reset

A hardware reset clears the CACR and disables the cache. The contents of the tag array are not affected by the reset. Accordingly, the system startup code must explicitly perform a cache invalidation by setting CACR[CINV] before the cache can be enabled.

4.3.5 Cache Miss Fetch Algorithm/Line Fills

As discussed in Section 4.1.2, “Introduction,” the cache hardware includes a 16-byte, line-fill buffer for providing temporary storage for the last fetched line.

With the cache enabled as defined by CACR[CENB], a cacheable fetch that misses in the tag memory and the line-fill buffer generates an external fetch. For data misses, the size of the external fetch is always 16 bytes. For instruction misses, the size of the external fetch is determined by the value contained in the 2-bit CLNF field of the CACR and the miss address. Table 4-6 shows the relationship between the CLNF bits, the miss address, and the size of the external fetch.

Table 4-6. Initial Fetch Offset vs. CLNF Bits

CLNF[1:0]	Longword Address Bits[3:2]			
	00	01	10	11
00	Line	Line	Line	Longword
01	Line	Line	Longword	Longword
1X	Line	Line	Line	Line

Depending on the runtime characteristics of the application and the memory response speed, overall performance may be increased by programming the CLNF bits to values 00 or 01.

For all cases of a line-sized fetch, the critical longword defined by bits [3:2] of the miss address is accessed first followed by the remaining three longwords that are accessed by incrementing the longword address in a modulo-16 fashion as shown below:

```
if miss address[3:2] = 00
   fetch sequence = 0x0, 0x4, 0x8, 0xC
if miss address[3:2] = 01
   fetch sequence = 0x4, 0x8, 0xC, 0x0
if miss address[3:2] = 10
   fetch sequence = 0x8, 0xC, 0x0, 0x4
if miss address[3:2] = 11
   fetch sequence = 0xC, 0x0, 0x4, 0x8
```

After an external fetch has been initiated and the data is loaded into the line-fill buffer, the cache maintains a special most-recently-used indicator that tracks the contents of the associated line-fill buffer versus its corresponding cache location. At the time of the miss, the hardware indicator is set, marking the line-fill buffer as most recently used. If a subsequent access occurs to the cache location defined by bits [10:4] (or bits [9:4] for split configurations of the fill buffer address), the data in the cache memory array is now most recently used, so the hardware indicator is cleared. In all cases, the indicator defines whether the contents of the line-fill buffer or the memory data array are most recently used. At the time of the next cache miss, the contents of the line-fill buffer are written into the memory array if the entire line is present, and the line-fill buffer data is most recently used compared to the memory array.

Generally, longword references are used for sequential instruction fetches. If the processor branches to an odd word address, a word-sized instruction fetch is generated.

For instruction fetches, the fill buffer can also be used as temporary storage for line-sized bursts of non-cacheable references under control of CACR[CEIB]. With this bit set, a non-cacheable instruction fetch is processed, as defined by Table 4-7. For this condition, the line-fill buffer is loaded and subsequent references can hit in the buffer, but the data is never loaded into the memory array.

Table 4-7 shows the relationship between CACR bits CENB and CEIB and the type of instruction fetch.

Table 4-7. Instruction Cache Operation as Defined by CACR

CACR [CENB]	CACR [CEIB]	Type of Instruction Fetch	Description
0	0	N/A	Cache is completely disabled; all instruction fetches are word or longword in size.
0	1	N/A	All instruction fetches are word or longword in size
1	X	Cacheable	Fetch size is defined by Table 4-6 and contents of the line-fill buffer can be written into the memory array
1	0	Non-cacheable	All instruction fetches are word or longword in size, and not loaded into the line-fill buffer
1	1	Non-cacheable	Instruction fetch size is defined by Table 4-6 and loaded into the line-fill buffer, but are never written into the memory array.

Chapter 5
Static RAM (SRAM)

5.1 SRAM Features

- One 64-Kbyte SRAM
- Single-cycle access
- Physically located on processor's high-speed local bus
- Memory location programmable on any 0-modulo-64 Kbyte address
- Byte, word, longword address capabilities

5.2 SRAM 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-64K address within the 4-GByte address space. 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 access or memory-referencing commands from the debug module.

Depending on configuration information, instruction fetches may be sent to both the cache and the SRAM block simultaneously. If the reference is mapped into the region defined by the SRAM, the SRAM provides the data back to the processor, and the cache data discarded. Accesses from the SRAM module are not cached.

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.

5.3 SRAM Programming Model

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

5.3.1 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 in a similar manner.
- 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.

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

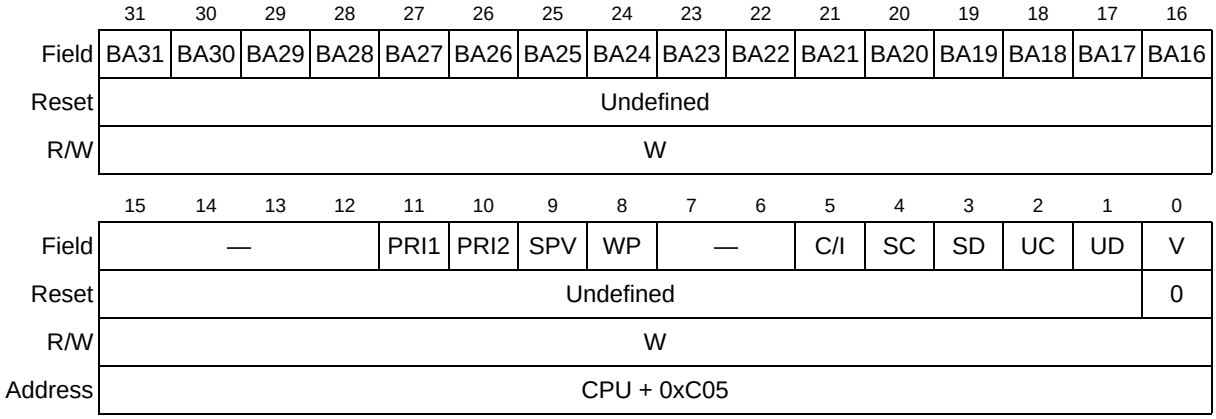


Figure 5-1. SRAM Base Address Register (RAMBAR)

Table 5-1. SRAM Base Address Register

Bits	Name	Description
31–16	BA	Base address. Defines the 0-modulo-64K base address of the SRAM module. By programming this field, the SRAM may be located on any 64-Kbyte boundary within the processor's 4-Gbyte address space.
15–12	—	Reserved, should be cleared.
11–10	PRI1, PRI2	Priority bit. PRI1 determines if DMA or CPU has priority in upper 32k bank of memory. PRI2 determines if DMA or CPU has priority in lower 32k bank of memory. If bit is set, CPU has priority. If bit is cleared, DMA has priority. Priority is determined according to the following table. **PRI[1:2]** / **Upper Bank Priority** / **Lower Bank Priority** 00 / DMA Accesses / DMA Accesses 01 / DMA Accesses / CPU Accesses 10 / CPU Accesses / DMA Accesses 11 / CPU Accesses / CPU Accesses NOTE: The Freescale-recommended setting for the priority bits is 00.
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.

Table 5-1. SRAM Base Address Register (continued)

Bits	Name	Description
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 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 5.3.4, “Power Management.”
0	V	Valid. A hardware reset clears this bit. When set, this bit enables the SRAM module; otherwise, the module is disabled. 0 Contents of RAMBAR are not valid 1 Contents of RAMBAR are valid

5.3.2 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 emulator using the debug module can perform these initialization functions.

5.3.3 SRAM Initialization Code

The following code segment describes how to initialize the SRAM. The code sets the base address of the SRAM at 0x20000000 and then initializes the SRAM to zeros.

```
RAMBASE         EQU $20000000             ;set this variable to $20000000

RAMVALID        EQU $00000001

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          #16384,D0               ;load loop counter into D0


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
```

5.3.4 Power Management

As noted previously, depending on the configuration defined by the RAMBAR, instruction fetch and operand read accesses may be sent to the SRAM and cache simultaneously. If the access is mapped to the SRAM module, it sources the read data and the unified cache access is discarded. If the SRAM is used only for data operands, asserting 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]
Code Only	0x2B
Data Only	0x35
Both Code And Data	0x21

Chapter 6
ColdFire Flash Module (CFM)

The MCF5282 incorporates SuperFlash® technology licensed from SST. The ColdFire Flash Module (CFM) is constructed with eight banks of 32K x 16-bit Flash to generate a 512-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 array used 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

The MCF5281 and MCF5214 implements only 256 Kbytes of Flash; half that of the MCF5282 and MCF5216.

The MCF5280 does not contain a Flash module.

6.1 Features

Features of the CFM include:

- 512-Kbytes of Flash memory on the MCF5282 and MCF5216
- 256-Kbytes of Flash memory on the MCF5281 and MCF5214
- 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.

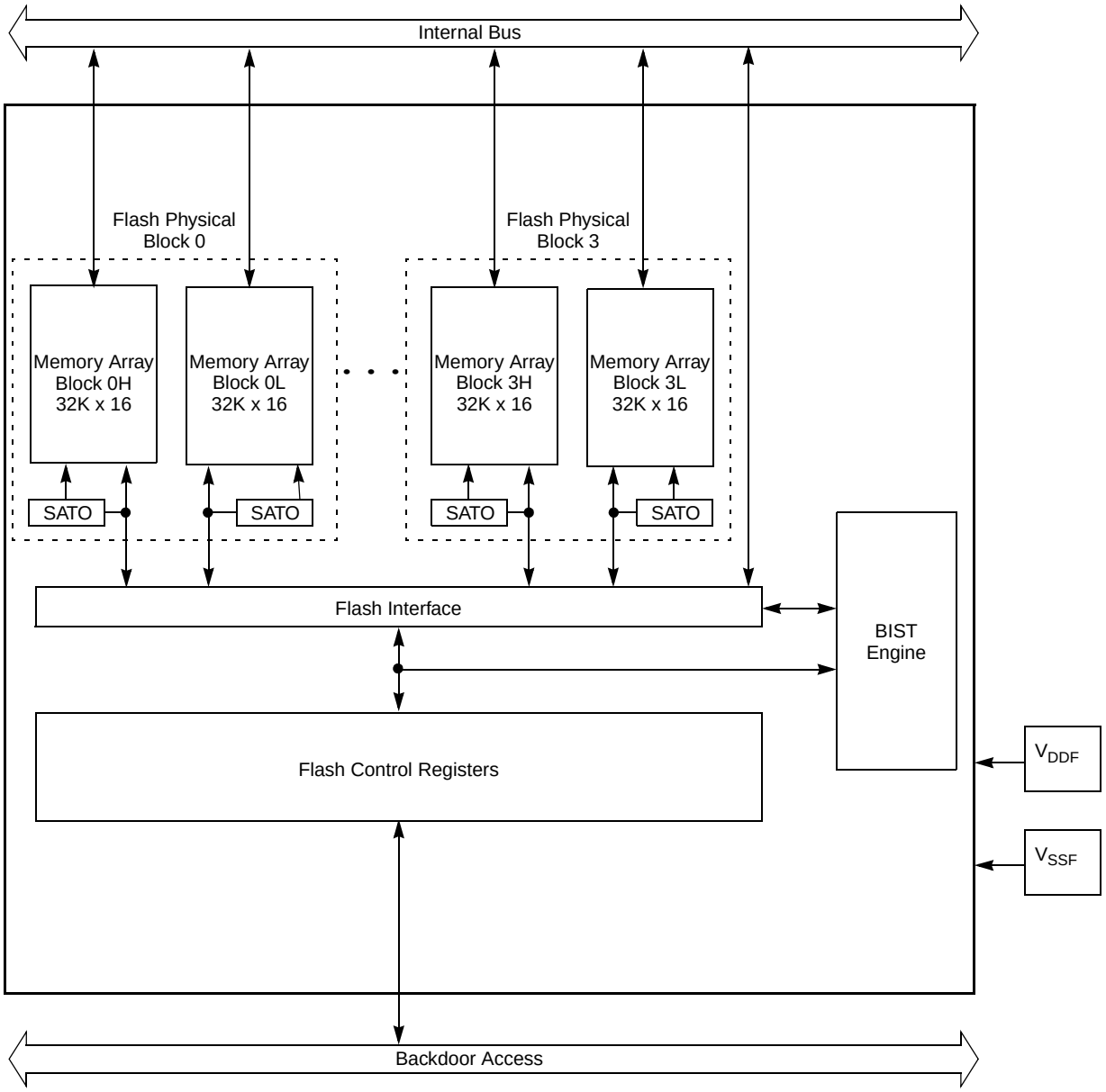
6.2 Block Diagram

The CFM module shown in Figure 6-1 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 *x*H or *x*L, 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. Therefore, it takes two mass erase operations, one on mass erase block 0 and one on mass erase block 1, to erase the full 512K CFM Flash on the MCF5282 and MCF5216.

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 (MCF5282 and MCF5216 only)

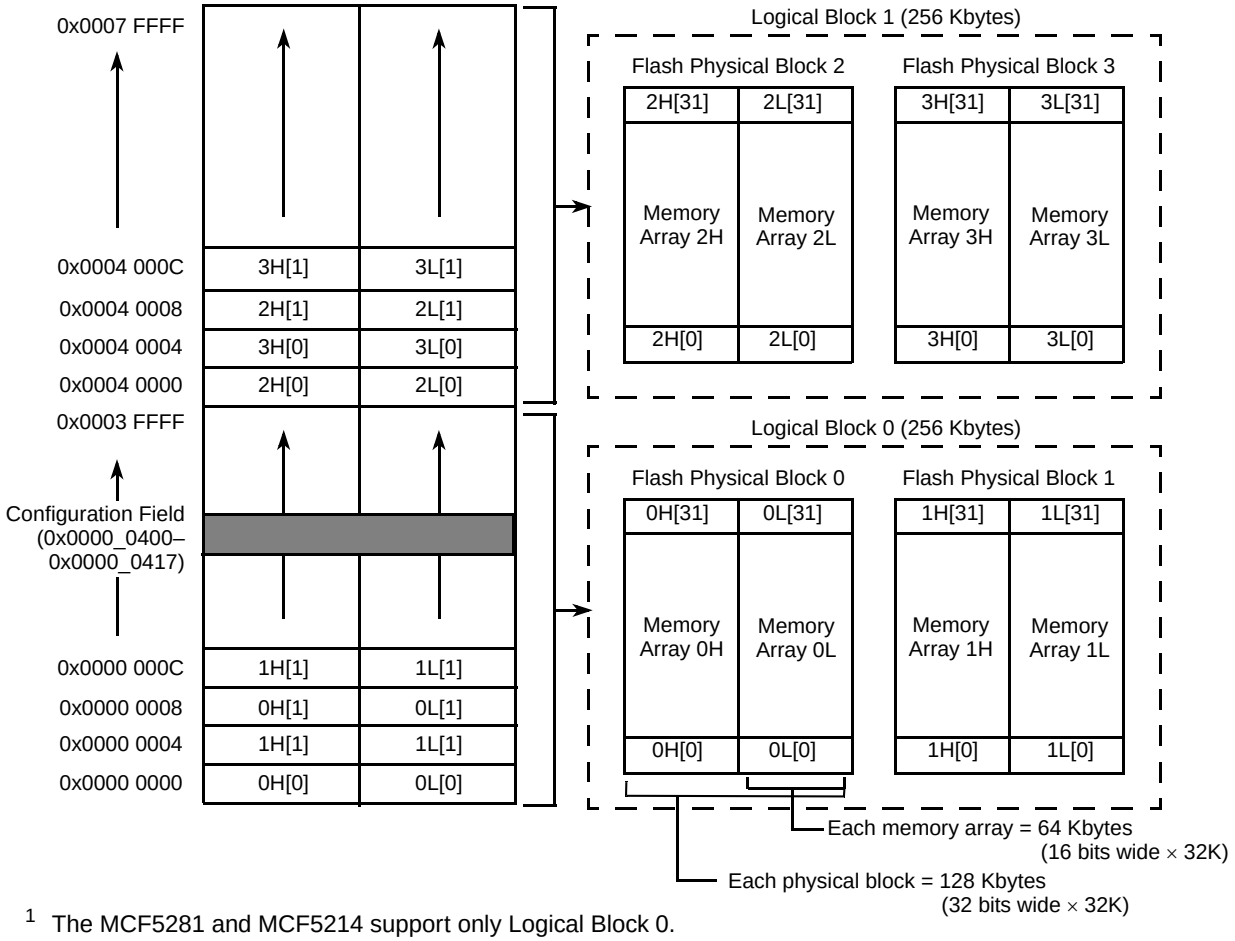
Figure 6-1. CFM Block Diagram

6.3 Memory Map

Figure 6-2 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 6-2 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 MCF5281 and MCF5214 is constructed with four banks of 32K x 16-bit Flash arrays to generate 256 Kbytes of 32-bit Flash memory.



[1] The MCF5281 and MCF5214 support only Logical Block 0.

Figure 6-2. CFM Array Memory Map

The CFM module has hardware interlocks to protect data from accidental corruption. The <<BLOCK NAME>> 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 6.3.4.4, “CFM Protection Register (CFMPROT)”). A similar mechanism is available to control supervisor/user and program/data space access to these sectors.

6.3.1 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 6-1 describes each byte used in this field.

Table 6-1. 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 6.3.4.4, “CFM Protection Register (CFMPROT)”)
0x0000_040C–0x0000_040F	4	Flash supervisor/user space restrictions Blocks 0H/0L (see Section 6.3.4.5, “CFM Supervisor Access Register (CFMSACC)”)
0x0000_0410–0x0000_0413	4	Flash program/data space restrictions Blocks 0H/0L (see Section 6.3.4.6, “CFM Data Access Register (CFMDACC)”)
0x0000_0414–0x0000_0417	4	Flash security longword (see Section 6.3.4.3, “CFM Security Register (CFMSEC)”)

6.3.2 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 6-3

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 Fast Ethernet 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 device (by default) boots 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	17	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	14	13	12	11	10	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										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 6-3. Flash Base Address Register (FLASHBAR)

Table 6-2. 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. Read only. Allows only read accesses to the Flash. This bit is always set and 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

6.3.3 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 6-3.

Table 6-3. 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 6-3. 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.

6.3.4 Register Descriptions

The Flash registers are described in this subsection.

6.3.4.1 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 6-4. CFM Module Configuration Register (CFMCR)

Bits 10 -5 in the CFMCR register are readable and writable with restrictions.

Table 6-4. 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 6-4. 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.

6.3.4.2 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.

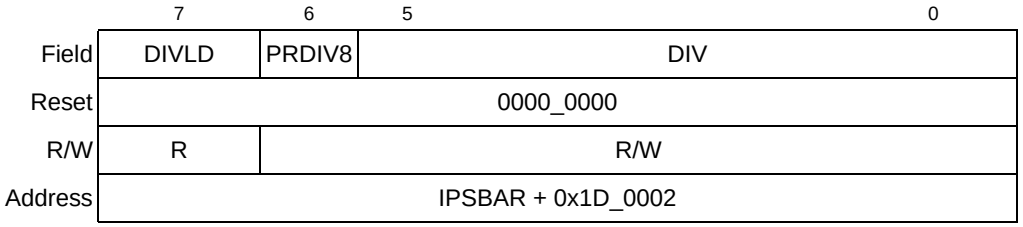


Figure 6-5. CFM Clock Divider Register (CFMCLKD)

All bits in CFMCLKD are readable. Bit 7 is a read-only status bit, while bits 6–0 can only be written once.

Table 6-5. 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 6.4.3.1, “Setting the CFMCLKD Register.”

6.3.4.3 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 6-6. CFM Security Register (CFMSEC)

Table 6-6. 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 6-6. CFMSEC Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>29–16</td><td>—</td><td>Reserved. Should be cleared.</td></tr>
<tr><td>15–0</td><td>SEC[15:0]</td><td>Security field. The SEC bits define the security state of the device; see below.
<table>
<tr><th>SEC[15:0]</th><th>Description</th></tr>
<tr><td>0x4AC8</td><td>Flash secured[1]</td></tr>
<tr><td>All other combinations</td><td>Flash unsecured</td></tr>
</table>
[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.</td></tr>
</table>

The security features of the CFM are described in Section 6.5, “Flash Security Operation.”

6.3.4.4 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 6-7. 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 6-7. 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 6-8 shows the association between each bit in the CFMPROT and its corresponding logical sector.

NOTE

Since the MCF5281 and MCF5214 devices contain a 256-Kbyte Flash, only CFMPROT[15:0] is used.

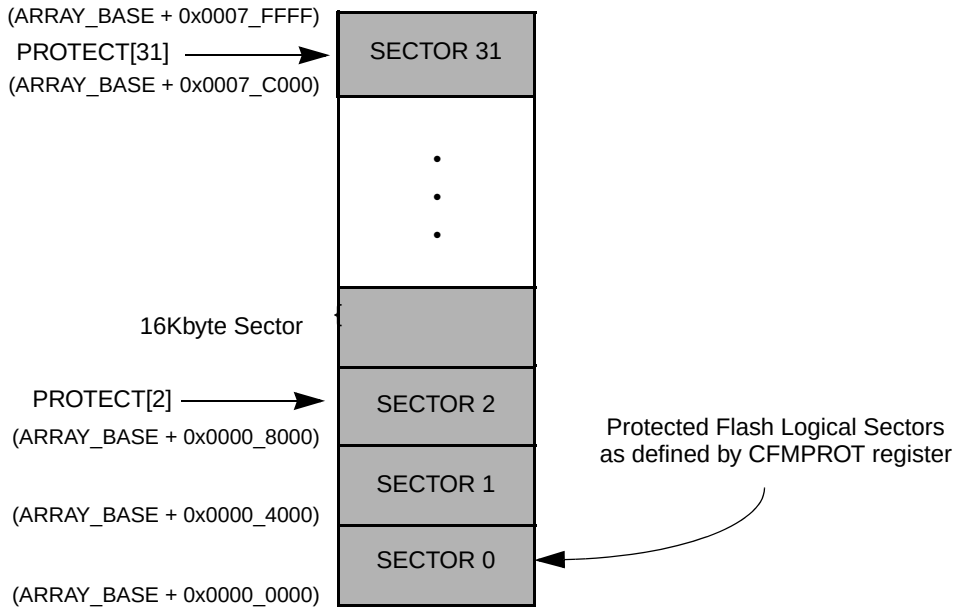


Figure 6-8. CFMPROT Protection Diagram

6.3.4.5 CFM Supervisor Access Register (CFMSACC)

The CFMSACC specifies the supervisor/user access permissions of Flash logical sectors.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	SUPV															
Reset	See Note															
R/W	R/W															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	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 6-9. CFM Supervisor Access Register (CFMSACC)

Table 6-8. 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 6-8 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.

6.3.4.6 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 6-10. CFM Data Access Register (CFMDACC)

Table 6-9. 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 6-8 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.

6.3.4.7 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 6-11. CFM User Status Register (CFMUSTAT)

NOTE

Only one CFMUSTAT bit should be cleared at a time.

Table 6-10. 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 6.4.3.4, “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.

Table 6-10. CFMUSTAT Field Descriptions

Bits	Name	Description
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.

6.3.4.8 CFM Command Register (CFMCMD)

The CFMCMD is the register to which Flash program, erase, and verify commands are written.

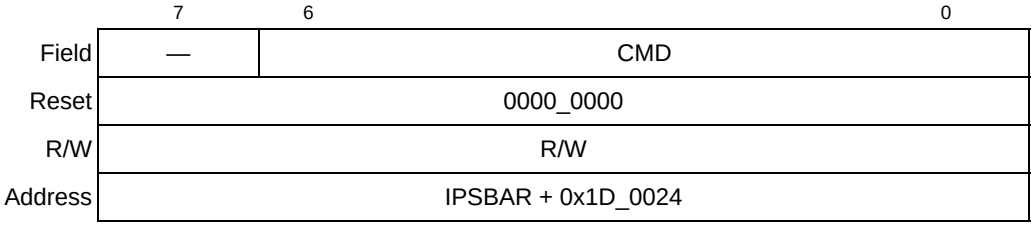


Figure 6-12. CFM Command Register (CFMCMD)

Table 6-11. 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 6-12. Writing a command in user mode other than those listed in Table 6-12 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 6-12. 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

6.4 CFM Operation

The CFM registers, subject to the restrictions previously noted, can generally be read and written (see Section 6.3.4, “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.

6.4.1 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.

6.4.2 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.

6.4.3 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.

6.4.3.1 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} ÷ 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:

$$f_{CLK} = \frac{f_{SYS}}{2 \times (DIV[5:0] + 1) \times (1 + (PRDIV8 \times 7))}$$

Consider the following example for f_{SYS} = 66 MHz:

$$DIV[5:0] = \frac{f_{SYS}}{2 \times 200kHz \times (1 + (PRDIV8 \times 7))}$$

$$= \frac{66\ MHz}{400\ kHz \times (1 + (1 \times 7))} = 20$$

$$f_{CLK} = \frac{f_{SYS}}{2 \times (DIV[5:0] + 1) \times (1 + (PRDIV8 \times 7))}$$

$$= \frac{66\ MHz}{2 \times (20 + 1) \times (1 + (1 \times 7))} = 196.43\ kHz$$

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 6.4.3.4, “Flash User Mode Illegal Operations”).

6.4.3.2 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 6.4.3.3, “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 6-13. The flow is similar for the erase and verify algorithms with the exceptions noted in step 1 above.

6.4.3.3 Flash Valid Commands

Table 6-13 summarizes the valid Flash user commands.

Table 6-13. 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.

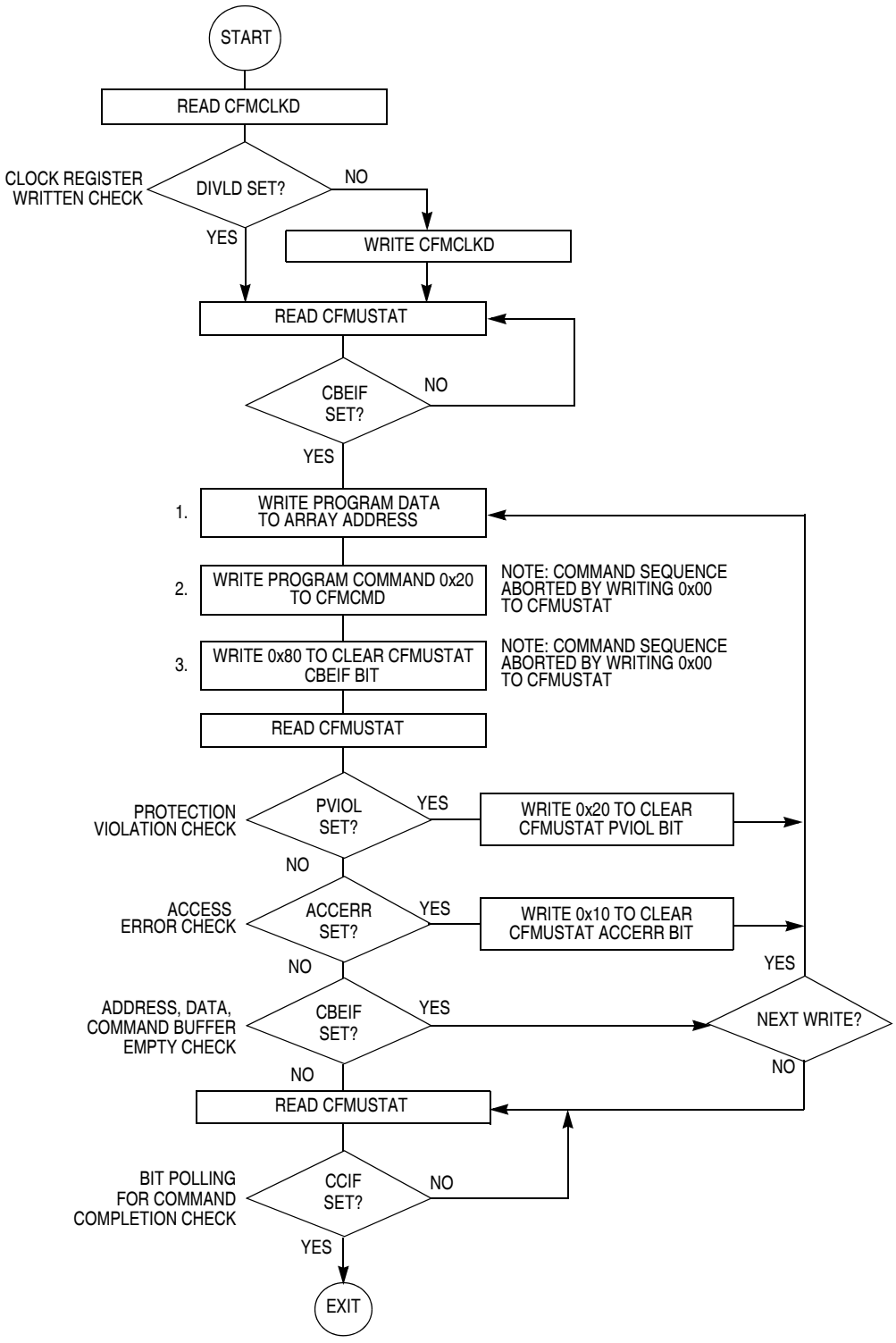


Figure 6-13. Example Program Algorithm

6.4.3.4 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 6.3.4.4, “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.

6.4.4 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.

6.4.5 Master Mode

If the MCU is booted in master mode with an external memory selected as the boot device, the CFM will not respond to the first transfer request out of reset. This will allow the external boot device to provide the reset vector and terminate the bus cycle.

6.5 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.

NOTE

When Flash security is enabled, the chip will boot in single chip mode regardless of the external reset configuration.

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.

6.5.1 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).

6.5.2 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.

6.6 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.

6.7 Interrupts

The CFM module can request an interrupt when all commands are completed or when the address, data, and command buffers are empty. Table 6-14 shows the CFM interrupt mechanism.

Table 6-14. CFM Interrupt Sources

Interrupt Source	Interrupt Flag	Local Enable
Command, data and address buffers empty	CBEIF (CFMUSTAT)	CBEIE (CFMCR)

Table 6-14. CFM Interrupt Sources (continued)

Interrupt Source	Interrupt Flag	Local Enable
All commands are completed	CCIF (CFMUSTAT)	CCIE (CFMCR)
Access error	ACCERR (CFMUSTAT)	AEIE (CFMCR)

Chapter 7
Power Management

The power management module (PMM) controls the device's low-power operation.

7.1 Features

The following features support low-power operation.

- Four modes of operation:
  - Run
  - Wait
  - Doze
  - Stop
- Ability to shut down most peripherals independently
- Ability to shut down the external CLKOUT pin

7.2 Memory Map and Registers

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

7.2.1 Programming Model

The PMM programming model consists of one register:

- The low-power control register (LPCR) specifies the low-power mode entered when the STOP instruction is issued, and controls clock activity in this low-power mode.

7.2.2 Memory Map

Table 7-1. Chip Configuration Module Memory Map

IPSBAR Offset	Bits 31–24	Bits 23–16	Bits 15–8	Bits 7–0	Access[1]
0x0000_0010	Core Reset Status Register (CRSR)[2]	Core Watchdog Control Register (CWCR)	Low-Power Interrupt Control Register (LPICR)	Core Watchdog Service Register (CWSR)	S
0x0011_0004	Chip Configuration Register (CCR)[3]		Reserved	Low-Power Control Register (LPCR)	S

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 The CRSR, CWCR, and CWSR are described in the System Integration Module. They are shown here only to warn against accidental writes to these registers when accessing the LPICR.

3 The CCR is described in the Chip Configuration Module. It is shown here only to warn against accidental writes to this register when accessing the LPCR.

7.2.3 Register Descriptions

The following subsection describes the PMM registers.

7.2.3.1 Low-Power Interrupt Control Register (LPICR)

Implementation of low-power stop mode and exit from a low-power mode via an interrupt require communication between the CPU and logic associated with the interrupt controller. The LPICR 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 (LPMD) field in the power management module's low-power control register (LPCR) 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 ENBSTOP bit in the LPICR must also be set.

Following is the sequence of operations needed to enable this functionality:

1. The LPICR is programmed, setting the ENBSTOP bit (if stop mode is the desired low-power mode) and loading the appropriate 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 LPICR[ENBSTOP] 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 LPICR[XLPM_IPL[2:0]].

NOTE

Only fixed (external) interrupt can bring a device out of stop mode. To exit from other low-power modes, such as doze or wait, either fixed or programmable interrupts may be used; however, the module generating the interrupt must be enabled in that particular low-power mode.

5. Once an appropriately-high interrupt request level arrives, the interrupt controller signals its presence, and the SIM 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.

	7	6		4	3	0
Field	ENBSTOP	XLPM_IPL			—	
Reset	1/0	0	1/0	0	Undefined	
R/W	R/W					
Address	IPSBAR + 0x012					

Figure 7-1. Low-Power Interrupt Control Register (LPICR)

Table 7-2. LPICR Field Descriptions

Bits	Name	Description
7	ENBSTOP	Enable low-power stop mode. 0 Low-power stop mode disabled 1 Low-power stop mode enabled. Once the core is stopped and the signal to enter stop mode is asserted, processor clocks can be disabled.
6–4	XLPM_IPL	Exit low-power mode interrupt priority level. This field defines the interrupt priority level needed to exit the low-power mode.Refer to Table 7-3.
3–0	—	Reserved, should be cleared.

Table 7-3. XLPM_IPL Settings

XLPM_IPL[2:0]	Interrupts 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

7.2.3.2 Low-Power Control Register (LPCR)

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

	7	6	5	4	3	2	1	0
Field	LPMD		—	STPMD		—	LVDSE	—
Reset	0000_0010							
R/W	R/W							
Address	IPSBAR + 0x0011_0007							

Figure 7-2. Low-Power Control Register (LPCR)

Table 7-4. LPCR Field Descriptions

Bits	Name	Description
7–6	LPMD	Low-power mode select. Used to select the low-power mode the chip enters once the ColdFire CPU executes the STOP instruction. These bits must be written prior to instruction execution for them to take effect. The LPMD[1:0] bits are readable and writable in all modes. Table 7-5 illustrates the four different power modes that can be configured with the LPMD bit field.
5	—	Reserved, should be cleared.
4–3	STPMD	PLL/CLKOUT stop mode. Controls PLL and CLKOUT operation in stop mode as shown in Table 7-6
2	—	Reserved, should be cleared.
1	LVDSE	LDV standby enable. Controls whether the PMM enters VREG Standby Mode (LVD disabled) or VREG Pseudo-Standby (LVD enabled) mode when the PMM receives a power down request. This bit has no effect if the RCR[LVDE] bit is a logic 0. 1 VREG Pseudo-Standby mode (LVD enabled on power down request). 0 VREG Standby mode (LVD disabled on power down request).
0	—	Reserved, should be cleared.

Table 7-5. Low-Power Modes

LPMD[1:0]	Mode
11	STOP
10	WAIT
01	DOZE
00	RUN

Table 7-6. PLL/CLKOUT Stop Mode Operation

STPMD[1:0]	Operation During Stop Mode				
	System Clocks	CLKOUT	PLL	OSC	PMM
00	Disabled	Enabled	Enabled	Enabled	Enabled
01	Disabled	Disabled	Enabled	Enabled	Enabled
10	Disabled	Disabled	Disabled	Enabled	Enabled
11	Disabled	Disabled	Disabled	Disabled	Low-Power Option

NOTE

If LPCR[LPMD] is cleared, then the device stops executing code upon issue of a STOP instruction. However, no clocks are disabled.

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 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 is programmed in 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.

For entry into stop mode, the LPICR[ENBSTOP] bit must be set before a STOP instruction is issued.

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

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

The latter method of exiting from low-power mode, by a valid and enabled interrupt request, requires several things:

- An interrupt request whose priority is higher than the value programmed in the XLPM_IPL field of the LPICR
- An interrupt request whose priority 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

7.3.1.1 Run Mode

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

7.3.1.2 Wait Mode

Wait mode is intended to be used to stop only the CPU and memory clocks until a wakeup 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.

7.3.1.3 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.

7.3.1.4 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.

NOTE

Entering stop mode will disable the SDRAMC including the refresh counter. If SDRAM is used, then code is required to insure proper entry and exit from stop mode. See Section 7.3.2.5, “SDRAM Controller (SDRAMC)” for more information.

7.3.1.5 Peripheral Shut Down

Most peripherals may be disabled by software in order to cease internal clock generation and remain in a static state. Each peripheral has its own specific disabling sequence (refer to each peripheral description for further details). A peripheral may be disabled at any time and will remain disabled during any low-power mode of operation.

7.3.2 Peripheral Behavior in Low-Power Modes

7.3.2.1 ColdFire Core

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

7.3.2.2 Static Random-Access Memory (SRAM)

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

7.3.2.3 Flash

The Flash module is in a low-power state if not being accessed. No recovery time is required after exit from any low-power mode.

The MCF5280 does not have a Flash module.

7.3.2.4 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 does not operate.

When enabled, the core Watchdog can bring the device out of low-power mode in one of two ways. If the core Watchdog reset/interrupt select (CSRI) bit is set, then a core Watchdog timeout will cause a reset of the device. If the CSRI bit is cleared, then a core Watchdog interrupt may be enabled and upon watchdog timeout, can bring the device out of low-power mode. This system setup must meet the conditions specified in Section 7.3.1, "Low-Power Modes" for the core Watchdog interrupt to bring the part out of low-power mode.

7.3.2.5 SDRAM Controller (SDRAMC)

SDRAMC operation is unaffected by either the wait or doze modes; however, the SDRAMC is disabled by stop mode. Since all clocks to the SDRAMC are disabled by stop mode, the SDRAMC will not generate refresh cycles.

To prevent loss of data the SDRAM should be placed in self-refresh mode by setting DCR[IS] before entering stop mode. The SDRAM self-refresh mode allows the SDRAM to enter a low-power state where internal refresh operations are used to maintain the integrity of the data stored in the SDRAM.

When stop mode is exited clearing the DCR[IS] bit will cause the SDRAM to exit the self-refresh mode and allow bus cycles to the SDRAM to resume.

NOTE

The SDRAM is inaccessible while in the self-refresh mode. Therefore, if stop mode is used the vector table and any interrupt handlers that could wake the processor should not be stored in or attempt to access SDRAM.

7.3.2.6 Chip Select Module

In wait and doze modes, the chip select module continues operation but does not generate interrupts; therefore it cannot bring a device out of a low-power mode. This module is stopped in stop mode.

7.3.2.7 DMA Controller (DMAC0–DMA3)

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 the DCR[INT] bit, and an interrupt is generated when the DSR[DONE] bit is set. The interrupt upon error condition is generated when the DCR[INT] bit is set, and an interrupt is generated when either the CE, BES or BED bit in the DSR becomes set.

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

7.3.2.8 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.

7.3.2.9 I^2C Module

When the I^2C Module is enabled by the setting of the I2CR[IEN] bit and when the device is not in stop mode, the I^2C 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^2C Module stops immediately and freezes operation, register values, and external pins. Upon exiting stop mode, the I^2C resumes operation unless stop mode was exited by reset.

7.3.2.10 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.

7.3.2.11 DMA Timers (DMAT0–DMAT3)

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.

7.3.2.12 Interrupt Controllers (INTC0, INTC1)

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 CPU processor 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). 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.

7.3.2.13 Fast Ethernet Controller (FEC)

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

- Clearing the ECNTRL[ETHER_EN] bit disables the FEC function.
- The FEC is unaffected by wait mode and may generate an interrupt to exit this mode.

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

The MCF5214 and MCF5216 do not contain an FEC.

7.3.2.14 I/O Ports

The I/O 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.

7.3.2.15 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 appropriate (LOCRE, LOLRE) bits in the synthesizer control register are set, then any loss-of-clock or loss-of-lock will reset the chip and exit any low-power modes.

If the 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.

7.3.2.16 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.

7.3.2.17 Clock Module

In wait and doze modes, the clocks to the CPU, Flash, 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 wakeup recovery time and stop mode power. The PLL may be disabled during stop mode. A wakeup 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 CLKOUT signal may be enabled during low-power stop (if the PLL is still enabled) to support systems using this signal as the clock source.

The system clocks may be enabled during wakeup from stop mode without waiting for the PLL to lock. This eliminates the wakeup 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.

During wakeup from stop mode, the Flash clock will always clock through 16 cycles before the system clocks are enabled. This allows the Flash module time to recover from the low-power mode. Thus, software may immediately continue to fetch instructions from the Flash memory.

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

7.3.2.18 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.

7.3.2.19 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.

7.3.2.20 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.

7.3.2.21 Queued Analog-to-Digital Converter (QADC)

Setting the queued analog-to-digital converter (QADC) stop bit (QSTOP) will disable the QADC.

The QADC is unaffected by either wait or doze mode and may generate an interrupt to exit these modes.

Low-power stop mode (or setting the QSTOP bit), immediately freezes operation, register values, state machines, and external pins. This stops the clock signals to the digital electronics of the module and eliminates the quiescent current draw of the analog electronics. Any conversion sequences in progress are stopped. Exit from low-power stop mode (or clearing the QSTOP bit), returns the QADC to operation from the state prior to stop mode entry, but any conversions in progress are undefined and the QADC requires recovery time to stabilize the analog circuits before new conversions can be performed.

7.3.2.22 General Purpose Timers (GPTA and GPTB)

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

Clearing the timer enable bit (TE) in the GPT system control register 1 (GPTSCR1) or the pulse accumulator enable bit (PAE) in the GPT pulse accumulator control register (GPTPACTL) disables timer functions. Timer and pulse accumulator registers are still accessible by the CPU and BDM interface, but the remaining functions of the timer are disabled.

The timer is unaffected by either the wait or doze modes and may generate an interrupt to exit these modes.

In stop mode, the General Purpose Timers stop immediately and freeze their operation, register values, state machines, and external pins. Upon exiting stop mode, the timer will resume operation unless stop mode was exited by reset.

7.3.2.23 FlexCAN

When enabled, the FlexCAN module is capable of generating interrupts and bringing the device out of a low-power mode. The module has 35 interrupt sources (32 sources due to message buffers and 3 sources due to Bus-off, Error and Wake-up).

When in stop mode, a recessive to dominant transition on the CAN bus causes the WAKE-INT bit in the error & status register to be set. This event can cause a CPU interrupt if the WAKE-MASK bit in module configuration register (MCR) is set.

When setting stop mode in the FlexCAN (by setting the MCR[STOP] bit), the FlexCAN checks for the CAN bus to be either idle or waits for the third bit of intermission and checks to see if it is recessive. When this condition exists, the FlexCAN waits for all internal activity other than in the CAN bus interface to complete and then the following occurs:

- The FlexCAN shuts down its clocks, stopping most of the internal circuits, to achieve maximum possible power saving.
- The internal bus interface logic continues operation, enabling CPU to access the MCR register.
- The FlexCAN ignores its Rx input pin, and drives its Tx pins as recessive.
- FlexCAN loses synchronization with the CAN bus, and STOP_ACK and NOT_RDY bits in MCR register are set.

Exiting stop mode is done in one of the following ways:

- Reset the FlexCAN (either by hard reset or by asserting the SOFT_RST bit in MCR).
- Clearing the STOP bit in the MCR.

- Self-wake mechanism. If the SELF-WAKE bit in the MCR is set at the time the FlexCAN enters stop mode, then upon detection of recessive to dominant transition on the CAN bus, the FlexCAN resets the STOP bit in the MCR and resumes its clocks.

Recommendations for, and features of, FlexCAN's stop mode operation are as follows:

- Upon stop/self-wake mode entry, the FlexCAN tries to receive the frame that caused it to wake; that is, it assumes that the dominant bit detected is a start-of-frame bit. It does not arbitrate for the CAN bus then.
- Before asserting stop Mode, the CPU should disable all interrupts in the FlexCAN, otherwise it may be interrupted while in stop mode upon a non-wake-up condition. If desired, the WAKE-MASK bit should be set to enable the WAKE-INT.
- If stop mode is asserted while the FlexCAN is BUSOFF (see error and status register), then the FlexCAN enters stop mode and stops counting the synchronization sequence; it continues this count once stop mode is exited.
- The correct flow to enter stop mode with SELF-WAKE:
  - assert SELF-WAKE at the same time as STOP.
  - wait for STOP_ACK bit to be set.
- The correct flow to negate STOP with SELF-WAKE:
  - negate SELF-WAKE at the same time as STOP.
  - wait for STOP_ACK negation.
- SELF-WAKE should be set only when the MCR[STOP] bit is negated and the FlexCAN is ready; that is, the NOT_RDY bit in the MCR is negated.
- If STOP and SELF_WAKE are set and if a recessive to dominant edge immediately follows on the CAN bus, the STOP_ACK bit in the MCR may never be set, and the STOP bit in the MCR is reset.
- If the user does not want to have old frames sent when the FlexCAN is awakened (STOP with Self-Wake), the user should disable all Tx sources, including remote-response, before stop mode entry.
- If halt mode is active at the time the STOP bit is set, then the FlexCAN assumes that halt mode should be exited; hence it tries to synchronize to the CAN bus (11 consecutive recessive bits), and only then does it search for the correct conditions to stop.
- Trying to stop the FlexCAN immediately after reset is allowed only after basic initialization has been performed.

If stop with self-wake is activated, and the FlexCAN operates with single system clock per time-quanta, then there are extreme cases in which FlexCAN's wake-up upon recessive to dominant edge may not conform to the standard CAN protocol, in the sense that the FlexCAN synchronization is shifted one time quanta from the required timing. This shift lasts until the next recessive to dominant edge, which re-synchronizes the FlexCAN back to conform to the protocol. The same holds for auto-power save mode upon wake-up by recessive to dominant edge.

The auto-power save mode in the FlexCAN is intended to enable NORMAL operation with optimized power saving. Upon setting the AUTO POWER SAVE bit in the MCR register, the FlexCAN looks for a set of conditions in which there is no need for clocks to run. If all these conditions are met, then the FlexCAN stops its clocks, thus saving power. While its clocks are stopped, if any of the conditions below is not met, the FlexCAN resumes its clocks. It then continues to monitor the conditions and stops/resumes its clocks appropriately.

The following are conditions for the automatic shut-off of FlexCAN clocks:

- No Rx/Tx frame in progress.
- No moving of Rx/Tx frames between SMB and MB and no Tx frame is pending for transmission in any MB.

- No host access to the FlexCAN module.
- The FlexCAN is neither in halt mode (MCR bit 8), in stop mode (MCT bit 15), nor in BUSOFF.

7.3.2.24 ColdFire Flash Module

The ColdFire Flash Control Module is capable of generating interrupts by the setting of the CBEIF or CCIF bits in the CFMUSTAT. These interrupt sources, however, should not occur when the device is in a low-power mode as long as no Flash operation was in progress when the low-power mode was entered.

When performing a program or erase operation on the Flash, if a command is active (CCIF = 0) when the MCU enters a low-power mode, the command sequence monitor will perform the following:

1. The command in progress will be aborted.
2. The Flash high voltage circuitry will be switched off and any pending command (CBEIF = 0) will not be executed when the MCU exits low-power mode.
3. The CCIF and ACCERR flags will be set if a command is active when the MCU enters low-power 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. Active commands are immediately aborted when the MCU enters stop mode. Do not execute the STOP instruction during program and erase operations.

7.3.2.25 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.

7.3.2.26 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.

7.3.3 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 7-7. 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 wakeup capability field refers to the ability of an interrupt or reset by that peripheral to force the CPU into run mode.

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

Module	Peripheral Status[1] / Wakeup Capability					
	Wait Mode		Doze Mode		Stop Mode	
CPU	Stopped	No	Stopped	No	Stopped	No
SRAM	Stopped	No	Stopped	No	Stopped	No
Flash	Stopped	No	Stopped	No	Stopped	No
Flash Control Module	Enabled	Yes[2]	Enabled	Yes[2]	Stopped	No
System Integration Module	Enabled	Yes [3]	Enabled	Yes [3]	Stopped	No
SDRAM Controller	Enabled	No	Enabled	No	Stopped	No
Chip Select Module	Enabled	No	Enabled	No	Stopped	No
DMA Controller	Enabled	Yes	Enabled	Yes	Stopped	No
UART0, UART1 and UART2	Enabled	Yes[2]	Enabled	Yes[2]	Stopped	No
I^2C Module	Enabled	Yes[2]	Enabled	Yes[2]	Stopped	No
QSPI	Enabled	Yes[2]	Enabled	Yes[2]	Stopped	No
DMA Timers	Enabled	Yes[2]	Enabled	Yes[2]	Stopped	No
Interrupt controller	Enabled	Yes[2]	Enabled	Yes[2]	Enabled	Yes[2]
Fast Ethernet Controller	Enabled	Yes[2]	Enabled	Yes[2]	Stopped	No
I/O Ports	Enabled	No	Enabled	No	Enabled	No
Reset Controller	Enabled	Yes[3]	Enabled	Yes[3]	Enabled	Yes[3]
Chip Configuration Module	Enabled	No	Enabled	No	Stopped	No
Power Management	Enabled	No	Enabled	No	Stopped	No
Clock Module	Enabled	Yes[2]	Enabled	Yes[2]	Program	Yes[2]
Edge port	Enabled	Yes[2]	Enabled	Yes[2]	Stopped	Yes[2]
Watchdog timer	Program	Yes [3]	Program	Yes [3]	Stopped	No
Programmable Interrupt Timers	Enabled	Yes[2]	Program	Yes[2]	Stopped	No
QADC	Enabled	Yes[2]	Program	Yes[2]	Stopped	No
General Purpose Timers	Enabled	Yes[2]	Enabled	Yes[2]	Stopped	No
FlexCAN	Enabled	Yes[2]	Enabled	Yes[2]	Stopped	No
BDM	Enabled	Yes[4]	Enabled	Yes[4]	Enabled	Yes[4]
JTAG	Enabled	No	Enabled	No	Enabled	No

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 interrupt which will exit a low-power mode. The CPU will begin to service the interrupt exception after wakeup.

3 These modules can generate a reset which will exit any low-power mode.

[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
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.

8.1 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 (interrupt) to allow recovery or corrective action to be taken.

8.2 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)

8.3 Memory Map and Register Definition

The memory map for the SCM registers is shown in Table 8-1. 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 8-1. SCM Register Map

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

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

8.4 Register Descriptions

8.4.1 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.

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. Cache
4. SDRAM

5. Chip Selects

NOTE

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

See Figure 8-1 and Table 8-2 for descriptions of the bits in IPSBAR.

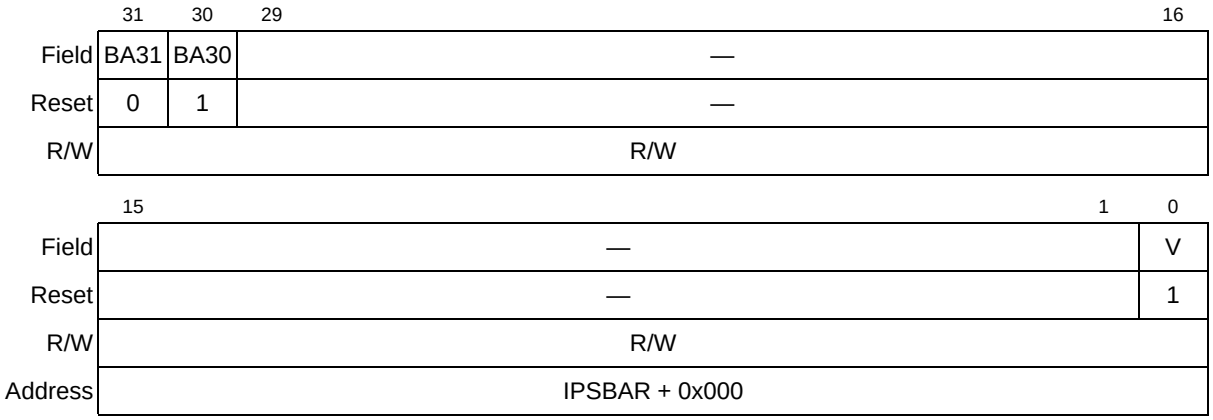


Figure 8-1. IPS Base Address Register (IPSBAR)

Table 8-2. 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.

8.4.2 Memory Base Address Register (RAMBAR)

The processor 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 processor 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.

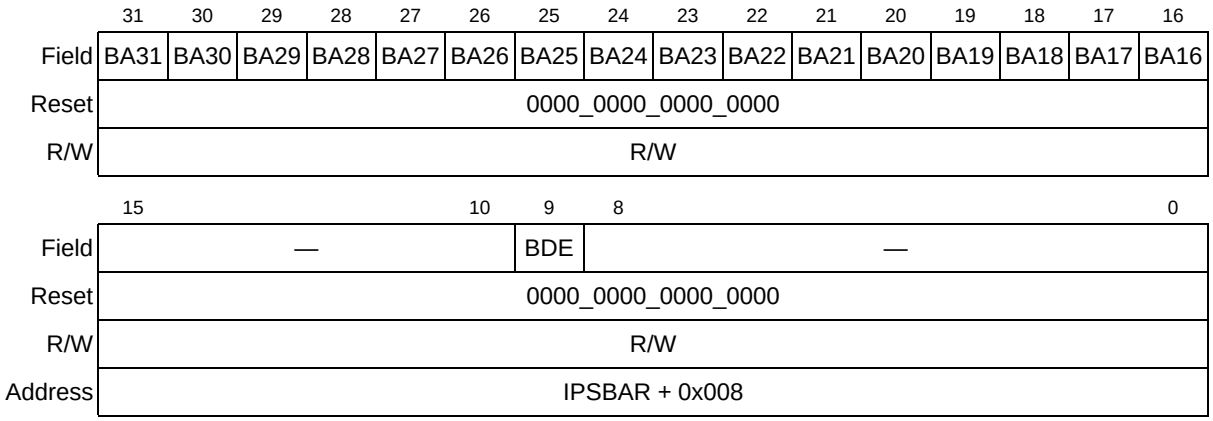


Figure 8-2. Memory Base Address Register (RAMBAR)

Table 8-3. 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 memory. 1 Enables module accesses to the memory. 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 5.3.1, "SRAM Base Address Register (RAMBAR)."
8–0	—	Reserved, should be cleared.

The SRAM modules are configured through the RAMBAR shown in Figure 8-2.

- 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 5, "Static RAM (SRAM)" for more information.

For details on the processor's view of the local SRAM memories, see Section 5.3.1, "SRAM Base Address Register (RAMBAR)."

8.4.3 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	—			
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 8-3. Core Reset Status Register (CRSR)

Table 8-4. CRSR Field Descriptions

Bits	Name	Description
7	EXT	External reset. 1 An external device driving $\overline{\text{RSTI}}$ 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.
6-0	—	Reserved.

8.4.4 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. 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.

NOTE

The core watchdog timer is available to provide compatibility with the watchdog timer implemented on previous ColdFire devices. However, there is a second watchdog timer available that has new features. See Chapter 18, "Watchdog Timer Module" for more information.

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.

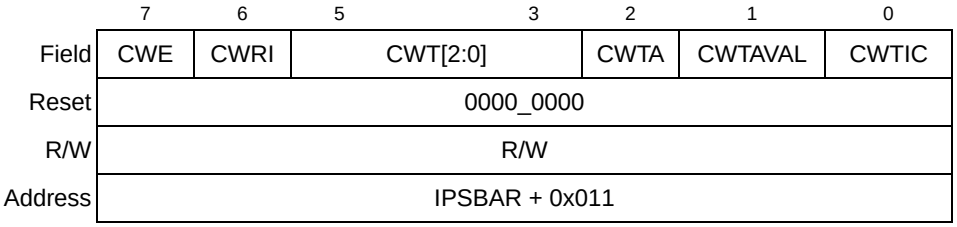


Figure 8-4. Core Watchdog Control Register (CWCR)

Table 8-5. 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 8 (ICR8) of INTC0. 1 Reserved; do not use.

Table 8-5. CWCR Field Description (continued)

<table>
<tr><td>5–3</td><td>CWT[2:0]</td><td>Core watchdog timing delay. These bits select the timeout period for the CWT. At system reset, the CWT field is cleared signaling the minimum time-out period but the watchdog is disabled (CWCR[CWE] = 0).
<table>
<tr><th>CWT</th><th>CWT Time-Out Period</th></tr>
<tr><td>000</td><td>2^9 Bus clock frequency</td></tr>
<tr><td>001</td><td>2^{11} Bus clock frequency</td></tr>
<tr><td>010</td><td>2^{13} Bus clock frequency</td></tr>
<tr><td>011</td><td>2^{15} Bus clock frequency</td></tr>
</table>
<table>
<tr><th>CWT</th><th>CWT Time-Out Period</th></tr>
<tr><td>100</td><td>2^{19} Bus clock frequency</td></tr>
<tr><td>101</td><td>2^{23} Bus clock frequency</td></tr>
<tr><td>110</td><td>2^{27} Bus clock frequency</td></tr>
<tr><td>111</td><td>2^{31} Bus clock frequency</td></tr>
</table>
</td></tr>
<tr><td>2</td><td>CWTA</td><td>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.</td></tr>
<tr><td>1</td><td>CWTAVAL</td><td>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.</td></tr>
<tr><td>0</td><td>CWTIF</td><td>Core watchdog timer interrupt flag.
0 CWT interrupt has not occurred
1 CWT interrupt has occurred. Write a 1 to clear the interrupt request.</td></tr>
</table>

8.4.5 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 8-5 illustrates the CWSR. At system reset, the contents of CWSR are uninitialized.

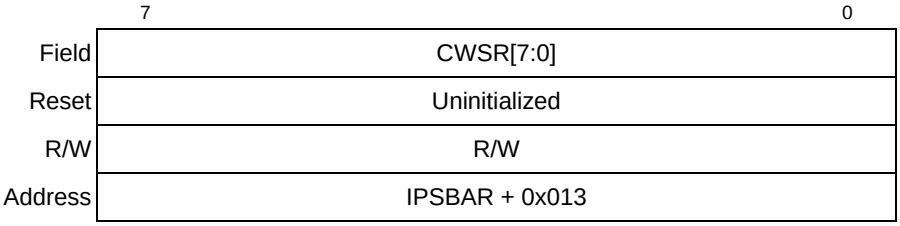


Figure 8-5. Core Watchdog Service Register (CWSR)

8.5 Internal Bus Arbitration

The internal bus arbitration is performed by the on-chip bus arbiter, which containing the arbitration logic that controls which of up to four MBus masters (M0–M3 in Figure 8-6) has access to the external buses. The function of the arbitration logic is described in this section.

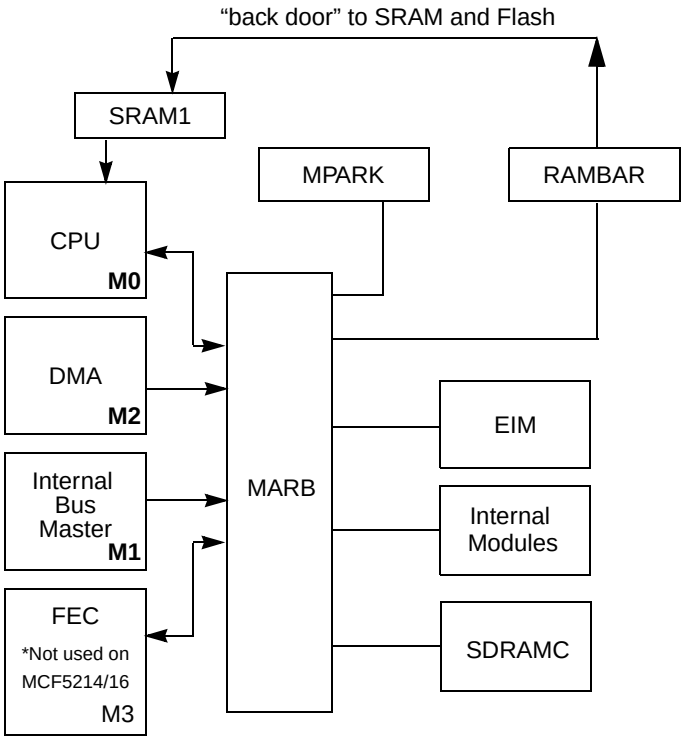


Figure 8-6. Arbiter Module Functions

8.5.1 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 FEC (M3) > DMA (M2) > internal master (M1) > CPU (M0), where M3 is the highest and M0 the lowest priority. M3 is not used for the MCF5216 and MCF5214.
- 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.

8.5.2 Arbitration Algorithms

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

8.5.2.1 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.

	M3 = 11 M2 =01 M1 = 10 M0 = 00
next +1	M3 = 00 M2 =10 M1 = 11 M0 = 01
next +2	M3 = 01 M2 =11 M1 = 00 M0 = 10
next +3	M3 = 10 M2 =00 M1 = 01 M0 = 11

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.

8.5.2.2 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.

8.5.3 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 3 (M3): Fast Ethernet Controller (Not used for the MCF5216 and MCF5214)
- Master 2 (M2): 4-channel DMA
- Master 1 (M1): Internal Bus Master (not used in normal user operation)
- Master 0 (M0): V2 ColdFire Core

	31								26	25	24	23	22	21	20	19	18	17	16
Field	—									M2_P_EN	BCR24BIT	M3_PRTY		M2_PRTY		M0_PRTY		M1_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 8-7. Default Bus Master Park Register (MPARK)

Table 8-6. 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	M3_PRTY	Master priority level for master 3 (Fast Ethernet Controller) 00 fourth (lowest) priority 01 third priority 10 second priority 11 first (highest) priority **Note:** Reserved on the MCF5214 and MCF5216
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	M1_PRTY	Master priority level for master 1 (Not used in user mode) 00 fourth (lowest) priority 01 third priority 10 second priority 11 first (highest) priority
15	—	Reserved, should be cleared.

Table 8-6. MPARK Field Description (continued)

Bits	Name	Description
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^ LCKOUT_TIME[3:0].
7–0	—	Reserved, should be cleared.

The initial state of the master priorities is M3 > M2 > M1 > M0. System software should guarantee that the programmed M*n*_PRTY fields are unique, otherwise the hardware defaults to the initial-state priorities.

NOTE

The M1_PRTY field should not be set for a priority higher than third (default).

8.6 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.

8.6.1 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.

8.6.2 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

8.6.3 Memory Map/Register Definition

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

Table 8-7. SACU Register Memory Map

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

8.6.3.1 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, not used on MCF5216 and MCF5214), 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 8-8. Master Privilege Register (MPR)

Table 8-8. 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.

8.6.3.2 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 8-9. For a list of PACRs and the modules that they control, refer to Table 8-11.

	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 8-9. Peripheral Access Control Register (PACR*n*)

Table 8-9. 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 8-10.
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 8-10.

Table 8-10. 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
110	Read/Write	Read/Write
111	No Access	No Access

Table 8-11. Peripheral Access Control Registers (PACRs)

IPSBAR Offset	Name	Modules Controlled	
		ACCESS_CTRL1	ACCESS_CTRL0
0x024	PACR0	SCM	SDRAMC
0x025	PACR1	EIM	DMA
0x026	PACR2	UART0	UART1

Table 8-11. Peripheral Access Control Registers (PACRs) (continued)

IPSBAR Offset	Name	Modules Controlled	
		ACCESS_CTRL1	ACCESS_CTRL0
0x027	PACR3	UART2	—
0x028	PACR4	I^2C	QSPI
0x029	—	—	—
0x02a	PACR5	DTIM0	DTIM1
0x02b	PACR6	DTIM2	DTIM3
0x02c	PACR7	INTC0	INTC1
0x02d	—	—	—
0x02e	PACR8	FEC0	—

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.

8.6.3.3 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 8-10. Grouped Peripheral Access Control Register (GPACR)

Table 8-12. 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 8-13.

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 8-13. Table 8-14 shows the memory space protected by the GPACRs and the modules mapped to these spaces.

Table 8-13. 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 8-14. 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) **Note:** Reserved for the MCF5280

Chapter 9
Clock Module

The clock module configures the device 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:

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

9.1 Features

Features of the clock module include:

- 2- to 10-MHz reference crystal oscillator
- Support for low-power modes
- Separate clock out signal

9.2 Modes of Operation

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

9.2.1 Normal PLL Mode

In normal PLL mode, the PLL is fully programmable. It can synthesize frequencies ranging from 2x to 9x 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.

9.2.2 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.

9.2.3 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.

9.3 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 9-1 shows the clock module operation in low-power modes.

Table 9-1. 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

During wakeup from a low-power mode, the Flash clock always clocks through at least 16 cycles before the CPU clocks are enabled. This allows the Flash module time to recover from the low-power mode, and software can immediately resume fetching instructions from memory.

In wait and doze modes, the system clocks to the peripherals are enabled, and the clocks to the CPU, Flash, 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.

9.4 Block Diagram

Figure shows a block diagram of the entire clock module. The PLL block in this diagram is expanded in detail in Figure 9-2.

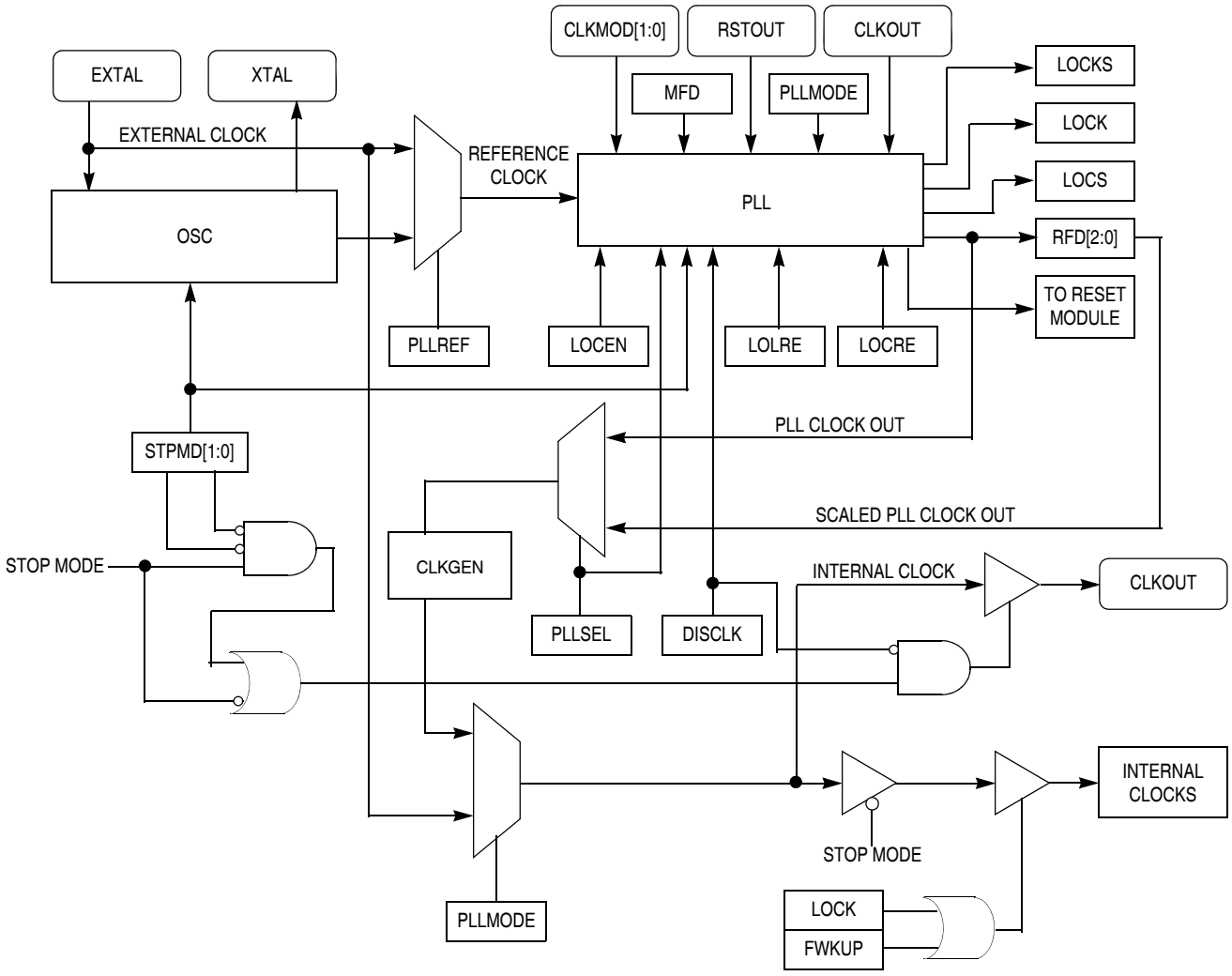


Figure 9-1. Clock Module Block Diagram

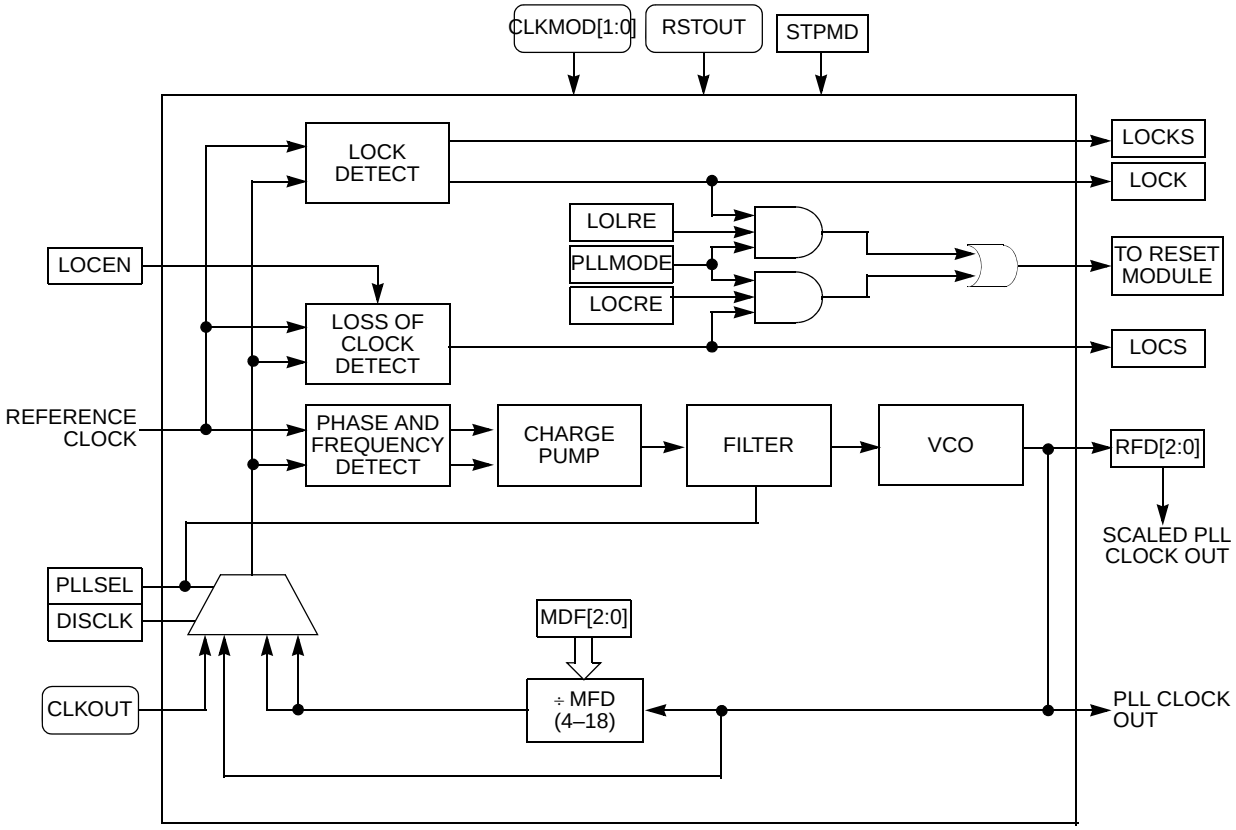


Figure 9-2. PLL Block Diagram

9.5 Signal Descriptions

The clock module signals are summarized in Table 9-2 and a brief description follows. For more detailed information, refer to Chapter 14, "Signal Descriptions."

Table 9-2. Signal Properties

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

9.5.1 EXTAL

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

9.5.2 XTAL

This output is an internal oscillator connection to the external crystal.

9.5.3 CLKOUT

This output reflects the internal system clock.

9.5.4 CLKMOD[1:0]

These inputs are used to select the clock mode during chip configuration.

9.5.5 $\overline{\text{RSTOUT}}$

The $\overline{\text{RSTOUT}}$ 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).”

9.6 Memory Map and Registers

The clock module programming model consists of these registers:

- Synthesizer control register (SYNCR), which defines clock operation
- Synthesizer status register (SYNSR), which reflects clock status

9.6.1 Module Memory Map

Table 9-3. Clock Module Memory Map

IPSBAR Offset	Register Name	Access[1]
0x0012_0000	Synthesizer Control Register (SYNCR)	S
0x0012_0002	Synthesizer Status Register (SYNSR)	S

1 S = CPU supervisor mode access only.

9.6.2 Register Descriptions

This subsection provides a description of the clock module registers.

9.6.2.1 Synthesizer Control Register (SYNCR)

	15	14	13	12	11	10	9	8
Field	LOLRE	MFD2	MFD1	MFD0	LOCRE	RFD2	RFD1	RFD0
Reset	0010_0001							
R/W	R/W							

	7	6	5	4	3	2	1	0
Field	LOCEN	DISCLK	FWKUP	—	STPMD1	STPMD0	—	—
Reset	0000_0000							
R/W	R/W			R	R/W		R	
Address	IPSBAR + 0x0012_0000							

Figure 9-3. Synthesizer Control Register (SYNCR)

Table 9-4. 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 9-4. SYNCR Field Descriptions (continued)

<table>
<tr><th>Bit(s)</th><th>Name</th><th>Description</th></tr>
<tr><td>14–12</td><td>MFD</td><td>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.
<table>
<tr><th></th><th></th><th colspan="8">MFD[2:0]</th></tr>
<tr><th></th><th></th><th>000[2] (4x)</th><th>001 (6x)</th><th>010 (8x)[(3)]</th><th>011 (10x)</th><th>100 (12x)</th><th>101 (14x)</th><th>110 (16x)</th><th>111 (18x)</th></tr>
<tr><td rowspan="8">RFD[2:0]</td><td>000 (÷ 1)</td><td>4</td><td>6</td><td>8</td><td>10</td><td>12</td><td>14</td><td>16</td><td>18</td></tr>
<tr><td>001 (÷ 2)[3]</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr>
<tr><td>010 (÷ 4)</td><td>1</td><td>3/2</td><td>2</td><td>5/2</td><td>3</td><td>7/2</td><td>4</td><td>9/2</td></tr>
<tr><td>011 (÷ 8)</td><td>1/2</td><td>3/4</td><td>1</td><td>5/4</td><td>3/2</td><td>7/4</td><td>2</td><td>9/4</td></tr>
<tr><td>100 (÷ 16)</td><td>1/4</td><td>3/8</td><td>1/2</td><td>5/8</td><td>3/4</td><td>7/8</td><td>1</td><td>9/8</td></tr>
<tr><td>101 (÷ 32)</td><td>1/8</td><td>3/16</td><td>1/4</td><td>5/16</td><td>3/8</td><td>7/16</td><td>1/2</td><td>9/16</td></tr>
<tr><td>110 (÷ 64)</td><td>1/16</td><td>3/32</td><td>1/8</td><td>5/32</td><td>3/16</td><td>7/32</td><td>1/4</td><td>9/32</td></tr>
<tr><td>111 (÷ 128)</td><td>1/32</td><td>3/64</td><td>1/16</td><td>5/64</td><td>3/32</td><td>7/64</td><td>1/8</td><td>9/64</td></tr>
</table>
1 $f_{sys} = \frac{f_{ref} \times 2(MFD + 2)}{2^{RFD}}$; $f_{ref} \times 2(MFD + 2) \leq f_{sys(max)}$; $f_{sys} \leq f_{sys(max)}$,
where $f_{sys(max)}$ is the maximum system frequency for the particular MCF5282 device (66 MHz or 80 MHz).
2 MFD = 000 not valid for f_{ref} < 3 MHz
3 Default value out of reset</td></tr>
<tr><td>11</td><td>LOCRE</td><td>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.</td></tr>
<tr><td>10–8</td><td>RFD</td><td>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.</td></tr>
</table>

Table 9-4. SYNCR Field Descriptions (continued)

<table>
<tr><th>Bit(s)</th><th>Name</th><th>Description</th></tr>
<tr><td>7</td><td>LOCEN</td><td>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.</td></tr>
<tr><td>6</td><td>DISCLK</td><td>Disable CLKOUT determines whether CLKOUT is driven. Setting the DISCLK bit holds CLKOUT low.
1 CLKOUT disabled
0 CLKOUT enabled</td></tr>
<tr><td>5</td><td>FWKUP</td><td>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.</td></tr>
<tr><td>4</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>3–2</td><td>STPMD</td><td>Control PLL and CLKOUT operation in stop mode. The following table illustrates STPMD operation in stop mode.
<table>
<tr><th rowspan="2">STPMD[1:0]</th><th colspan="4">Operation During Stop Mode</th></tr>
<tr><th>System Clocks</th><th>PLL</th><th>OSC</th><th>CLKOUT</th></tr>
<tr><td>00</td><td>Disabled</td><td>Enabled</td><td>Enabled</td><td>Enabled</td></tr>
<tr><td>01</td><td>Disabled</td><td>Enabled</td><td>Enabled</td><td>Disabled</td></tr>
<tr><td>10</td><td>Disabled</td><td>Disabled</td><td>Enabled</td><td>Disabled</td></tr>
<tr><td>11</td><td>Disabled</td><td>Disabled</td><td>Disabled</td><td>Disabled</td></tr>
</table></td></tr>
<tr><td>1–0</td><td>—</td><td>Reserved, should be cleared.</td></tr>
</table>

9.6.2.2 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.

	7	6	5	4	3	2	1	0
Field	PLLMODE	PLLSEL	PLLREF	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.

Figure 9-4. Synthesizer Status Register (SYNSR)

Table 9-5. SYNSR Field Descriptions

Bit(s)	Name	Description
7	PLLMODE	Clock mode bit. The PLLMODE bit is configured at reset and reflects the clock mode as shown in Table 9-6. 1 PLL clock mode 0 External clock mode
6	PLLSEL	PLL select. Configured at reset and reflects the PLL mode as shown in Table 9-6. 1 Normal PLL mode 0 1:1 PLL mode
5	PLLREF	PLL reference. Configured at reset and reflects the PLL reference source in normal PLL mode as shown in Table 9-6. 1 Crystal clock reference 0 External 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

Table 9-5. SYNSR Field Descriptions (continued)

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.

Table 9-6. System Clock Modes

PLLMODE:PLLSEL:PLLREF	Clock Mode
000	External clock mode
100	1:1 PLL mode
110	Normal PLL mode with external clock reference
111	Normal PLL mode with crystal reference

9.7 Functional Description

This subsection provides a functional description of the clock module.

9.7.1 System Clock Modes

The system clock source is determined during reset (see Table 27-8). The values of CLKMOD[1:0] are latched during reset and are of no importance after reset is negated. If CLKMOD1 or CLKMOD0 is changed 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 9-7 shows the clockout frequency to clockin frequency relationships for the possible system clock modes.

Table 9-7. 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.

CAUTION

XTAL must be tied low in external clock mode when reset is asserted. If it is not, clocks could be suspended indefinitely.

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

9.7.2 Clock Operation During Reset

In external clock mode, the system is static and does not recognize reset until a clock is applied to EXTAL.

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.

9.7.3 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.

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.

9.7.4 PLL Operation

In PLL mode, the PLL synthesizes the system clocks. The PLL can multiply the reference clock frequency by 2x to 9x, 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 4 MHz to 18 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 9-5 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 to see how these functional sub-blocks interact.

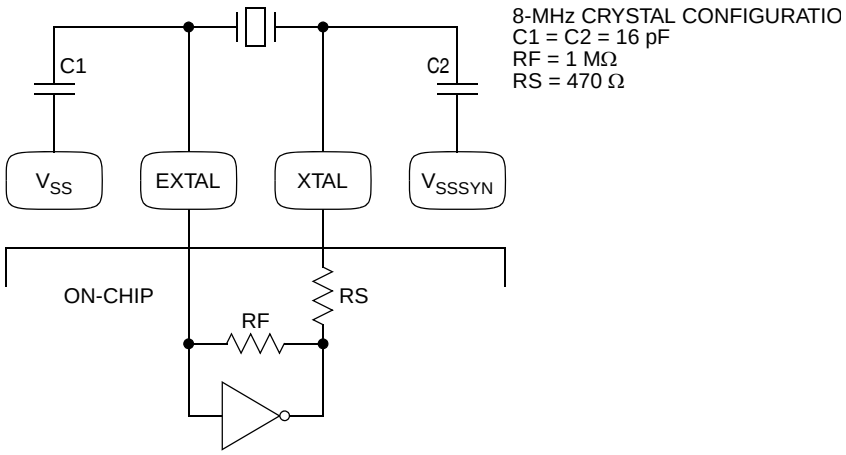


Figure 9-5. Crystal Oscillator Example

9.7.4.1 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.

9.7.4.2 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 9-8.

Table 9-8. Charge Pump Current and MFD in Normal Mode Operation

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

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.

9.7.4.3 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.

9.7.4.4 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.

9.7.4.5 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 9-6 shows the sequence for detecting locked and non-locked conditions.

In external clock mode, the PLL is disabled and cannot lock.

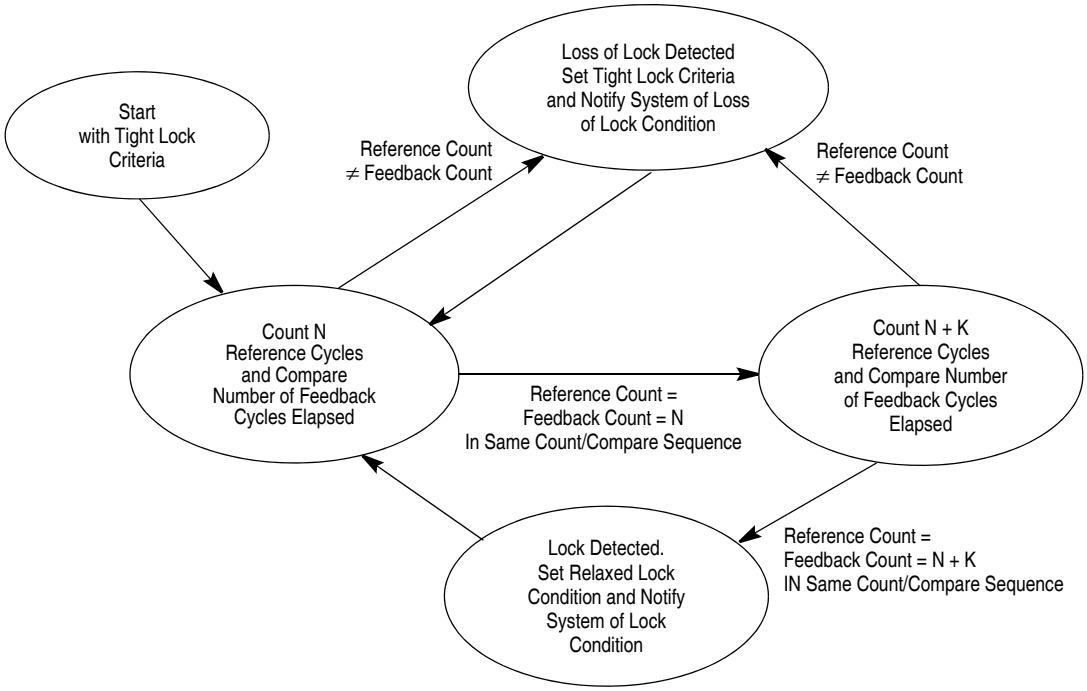


Figure 9-6. Lock Detect Sequence

9.7.4.6 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 is 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.

9.7.4.7 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.

9.7.4.8 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.

9.7.4.9 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 9.6, "Memory Map and Registers."

9.7.4.10 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 9-9 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 9-9. 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 9-5.

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.

9.7.4.11 Loss of Clock in Stop Mode

Table 9-10 shows the resulting actions for a loss of clock in Stop Mode when the device is being clocked by the various clocking methods.

Table 9-10. 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	—	—	—	

Table 9-10. Stop Mode Operation (Sheet 2 of 5)

MODE In	LOCEN	LOCRE	LOLRE	PLL	OSC	EWKUP	Expected PLL Action at Stop	PLL Action During Stop	MODE Out	LOCKSS	LOCK	LOCS	Comments
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
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

Table 9-10. Stop Mode Operation (Sheet 3 of 5)

MODE In	LOCEN	LOCRE	LOLRE	PLL	OSC	EWKUP	Expected PLL Action at Stop	PLL Action During Stop	MODE Out	LOCKSS	LOCK	LOCS	Comments
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
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	

Table 9-10. Stop Mode Operation (Sheet 4 of 5)

MODE In	LOCEN	LOCRE	LOLRE	PLL	OSC	EWKUP	Expected PLL Action at Stop	PLL Action During Stop	MODE Out	LOCKSS	LOCK	LOCS	Comments
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	
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 9-10. Stop Mode Operation (Sheet 5 of 5)

MODE In	LOCEN	LOCRE	LOLRE	PLL	OSC	EWKUP	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 10
Interrupt Controller Modules

This section details the functionality for the interrupt controllers (INTC0, INTC1). The general features of each interrupt controller include:

- 63 interrupt sources, organized as:
  - 56 fully-programmable interrupt sources
  - 7 fixed-level interrupt sources
- Each of the 63 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 56 fully-programmable and seven fixed-level interrupt sources for each of the two interrupt controllers 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.

10.1 68K/ColdFire Interrupt Architecture Overview

Before continuing with the specifics of the interrupt controllers, 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 requires 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 2.3.3.1, “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 10.1.1.3, “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.

10.1.1 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 interrupt priority structure shown in Table 10-1, which orders the interrupt levels/priorities from highest to lowest.

Table 10-1. Interrupt Priority Scheme

Interrupt Level ICR[IL]	Priority ICR[IP]	Supported Interrupt Sources
7	7	#8–63
	6	
	5	
	4	
	— (Mid-point)	#7 (IRQ7)
	3	#8–63
	2	
	1	
	0	

Table 10-1. Interrupt Priority Scheme (continued)

Interrupt Level ICR[IL]	Priority ICR[IP]	Supported Interrupt Sources
6	7–4	#8–63
	— (Mid-point)	#6 (IRQ6)
	3–0	#8–63
5	7–4	#8–63
	— (Mid-point)	#5 (IRQ5)
	3–0	#8–63
4	7–4	#8–63
	— (Mid-point)	#4 (IRQ4)
	3–0	#8–63
3	7–4	#8–63
	— (Mid-point)	#3 (IRQ3)
	3–0	#8–63
2	7–4	#8–63
	— (Mid-point)	#2 (IRQ2)
	3–0	#8–63
1	7–4	#8–63
	— (Mid-point)	#1 (IRQ1)
	3–0	#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

10.1.1.1 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.

10.1.1.2 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. The decoded priority levels from all the interrupt controllers are logically summed together and the highest enabled interrupt request is then encoded into a 3-bit priority level that is sent to the processor core during this prioritization phase.

10.1.1.3 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

For INTC1,                vector_number = 128 + 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.

Vector numbers 64-71, and 91-255 are unused.

10.2 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 10-3. The offsets listed start from the base address for each interrupt controller. The base addresses for the interrupt controllers are listed below:

Table 10-2. Interrupt Controller Base Addresses

Interrupt Controller Number	Base Address
INTC0	IPSBAR + 0xC00
INTC1	IPSBAR + 0xD00
Global IACK Registers Space[1]	IPSBAR + 0xF00

[1] This address space only contains the L1ACK-L7IACK registers. See Section 10.3.7, “Software and Level n IACK Registers (SWIACKR, L1IACK–L7IACK)" for more information

Table 10-3. Interrupt Controller Memory Map

Module Offset	Bits[31:24]	Bits[23:16]	Bits[15:8]	Bits[7:0]
0x00	Interrupt Pending Register High (IPRH), [63:32]			
0x04	Interrupt Pending Register Low (IPRL), [31:1]			
0x08	Interrupt Mask Register High (IMRH), [63:32]			
0x0c	Interrupt Mask Register Low (IMRL), [31:0]			
0x10	Interrupt Force Register High (INTFRCH), [63:32]			
0x14	Interrupt Force Register Low (INTFRCL), [31:1]			
0x18	IRLR[7:1]	IACKLPR[7:0]	Reserved	
0x1C–0x3C	Reserved			
0x40	Reserved	ICR01	ICR02	ICR03
0x44	ICR04	ICR05	ICR06	ICR07
0x48	ICR08	ICR09	ICR10	ICR11
0x4C	ICR12	ICR13	ICR14	ICR15
0x50	ICR16	ICR17	ICR18	ICR19
0x54	ICR20	ICR21	ICR22	ICR23
0x58	ICR24	ICR25	ICR26	ICR27
0x5C	ICR28	ICR29	ICR30	ICR31
0x60	ICR32	ICR33	ICR34	ICR35
0x64	ICR36	ICR37	ICR38	ICR39
0x68	ICR40	ICR41	ICR42	ICR43
0x6C	ICR44	ICR45	ICR46	ICR47
0x70	ICR48	ICR49	ICR50	ICR51
0x74	ICR52	ICR53	ICR54	ICR55
0x78	ICR56	ICR57	ICR58	ICR59
0x7C	ICR60	ICR61	ICR62	ICR63

Table 10-3. Interrupt Controller Memory Map (continued)

Module Offset	Bits[31:24]	Bits[23:16]	Bits[15:8]	Bits[7:0]
0x80–0xDC	Reserved			
0xE0	SWIACK	Reserved		
0xE4	L1IACK	Reserved		
0xE8	L2IACK	Reserved		
0xEC	L3IACK	Reserved		
0xF0	L4IACK	Reserved		
0xF4	L5IACK	Reserved		
0xF8	L6IACK	Reserved		
0xFC	L7IACK	Reserved		

10.3 Register Descriptions

10.3.1 Interrupt Pending Registers (IPRH*n*, IPRL*n*)

The IPRH*n* and IPRL*n* registers, Figure 10-1 and Figure 10-2, 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, 0xD00

Figure 10-1. Interrupt Pending Register High (IPRH*n*)

Table 10-4. 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, 0xD04		

Figure 10-2. Interrupt Pending Register Low (IPRL*n*)

Table 10-5. 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.

10.3.2 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, 0xD08	

Figure 10-3. Interrupt Mask Register High (IMRH*n*)

Table 10-6. 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, 0xD0C		

Figure 10-4. Interrupt Mask Register Low (IMRL*n*)

Table 10-7. 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.

10.3.3 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	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	INTFRCH[63:48]															
Reset	0000_0000_0000_0000															
R/W	R/W															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	INTFRCH[47:32]															
Reset	0000_0000_0000_0000															
R/W	R/W															

IPSBAR + 0xC10, 0xD10

Figure 10-5. Interrupt Force Register High (INTFRCH*n*)

Table 10-8. 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	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	INTFRCL[31:16]															
Reset	0000_0000_0000_0000															
R/W	R/W															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	INTFRCL[16:1]															—
Reset	0000_0000_0000_0000															
R/W	R/W															

IPSBAR + 0xC14, 0xD14

Figure 10-6. Interrupt Force Register Low (INTFRCL*n*)

Table 10-9. 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.

10.3.4 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. This register output is combined with similar outputs from INTC1 and eventually encoded into the 3-bit priority interrupt level driven to the processor core.

	7	2	1	0
Field	IRQ[7:1]			—
Reset	0000_0000			
R/W	R			
Address	IPSBAR + 0xC18, 0xD18			

Figure 10-7. Interrupt Request Level Register (IRLR*n*)

Table 10-10. 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

10.3.5 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 10-8 and Table 10-11.

	7	6	4	3	0
Field	—	LEVEL		PRI	
Reset	0000_0000				
R/W	R				
Address	IPSBAR + 0xC19, 0xD19				

Figure 10-8. IACK Level and Priority Register (IACKLPR*n*)

Table 10-11. IACKLPR*n* Field Descriptions

Bits	Name	Description
7	—	Reserved

Table 10-11. 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

10.3.6 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 the responsibility of the software 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 10-2 and Table 10-3 for register offsets					

Note: It is the responsibility of the software to program the ICR*n*x registers with unique and non-overlapping level and priority definitions. Failure to program the ICR*n*x registers in this manner can result in undefined behavior. If a specific interrupt request is completely unused, the ICR*n*x value can remain in its reset (and disabled) state.

Figure 10-9. Interrupt Control Register (ICR*nx*)

Table 10-12. 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.

10.3.6.1 Interrupt Sources

Table 10-13 and Table 10-14 list the interrupt sources for each interrupt request line for INTC0 and INTC1.

Table 10-13. Interrupt Source Assignment for INTC0

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	SWT1	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	Multiple	UART0 interrupt	Cleared when service complete
14	UART1	Multiple	UART1 interrupt	Cleared when service complete
15	UART2	Multiple	UART2 interrupt	Cleared when service complete
16	Not used			
17	I^2C	IIF	I^2C interrupt	Write IIF = 0
18	QSPI	Multiple	QSPI interrupt	See QIR description
19	DTIM0	CAP/REF	DTIM0 capture/reference event	Write CAP = 1 or REF = 1
20	DTIM1	CAP/REF	DTIM1 capture/reference event	Write CAP = 1 or REF = 1
21	DTIM2	CAP/REF	DTIM2 capture/reference event	Write CAP = 1 or REF = 1
22	DTIM3	CAP/REF	DTIM3 capture/reference event	Write CAP = 1 or REF = 1

Table 10-13. Interrupt Source Assignment for INTC0 (continued)

Source	Module	Flag	Source Description	Flag Clearing Mechanism
23	FEC **Note:** Not used on MCF5214 & MCF5216	X_INTF	Transmit frame interrupt	Write X_INTF = 1
24		X_INTB	Transmit buffer interrupt	Write X_INTB = 1
25		UN	Transmit FIFO underrun	Write UN = 1
26		RL	Collision retry limit	Write RL = 1
27		R_INTF	Receive frame interrupt	Write R_INTF = 1
28		R_INTB	Receive buffer interrupt	Write R_INTB = 1
29		MII	MII interrupt	Write MII = 1
30		LC	Late collision	Write LC = 1
31		HBERR	Heartbeat error	Write HBERR = 1
32		GRA	Graceful stop complete	Write GRA = 1
33		EBERR	Ethernet bus error	Write EBERR = 1
34		BABT	Babbling transmit error	Write BABT = 1
35		BABR	Babbling receive error	Write BABR = 1
36	PMM	LVDF	LVD	Write LVDF = 1
37	QADC	CF1	Queue 1 conversion complete	Write CF1 = 0 after reading CF1 = 1
38		CF2	Queue 2 conversion complete	Write CF2 = 0 after reading CF2 = 1
39		PF1	Queue 1 conversion pause	Write PF1 = 0 after reading PF1 = 1
40		PF2	Queue 2 conversion pause	Write PF2 = 0 after reading PF2 = 1
41	GPTA	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 C1F = 1 or access IC/OC if TFFCA = 1
46		C2F	Timer channel 2	Write C2F = 1 or access IC/OC if TFFCA = 1
47		C3F	Timer channel 3	Write C3F = 1 or access IC/OC if TFFCA = 1
48	GPTB	TOF	Timer overflow	Write TOF = 1 or access TIMCNTH/L if TFFCA = 1
49		PAIF	Pulse accumulator input	Write PAIF = 1 or access PAC if TFFCA = 1
50		PAOVF	Pulse accumulator overflow	Write PAOVF = 1 or access PAC if TFFCA = 1
51		C0F	Timer channel 0	Write C0F = 1 or access IC/OC if TFFCA = 1
52		C1F	Timer channel 1	Write C1F = 1 or access IC/OC if TFFCA = 1
53		C2F	Timer channel 2	Write C2F = 1 or access IC/OC if TFFCA = 1
54		C3F	Timer channel 3	Write C3F = 1 or access IC/OC if TFFCA = 1
55	PIT0	PIF	PIT interrupt flag	Write PIF = 1 of write PMR

Table 10-13. Interrupt Source Assignment for INTC0 (continued)

Source	Module	Flag	Source Description	Flag Clearing Mechanism
56	PIT1	PIF	PIT interrupt flag	Write PIF = 1 of write PMR
57	PIT2	PIF	PIT interrupt flag	Write PIF = 1 of write PMR
58	PIT3	PIF	PIT interrupt flag	Write PIF = 1 of write PMR
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	Not Used			

Table 10-14. Interrupt Source Assignment for INTC1

Source	Module	Flag	Source Description	Flag Clearing Mechanism
1-7	Not Used			
8	FLEX CAN	BUF0I	Message buffer 0 interrupt	Write BUF0I = 1 after reading BUF0I = 1
9		BUF1I	Message buffer 1 interrupt	Write BUF1I = 1 after reading BUF1I = 1
10		BUF2I	Message buffer 2 interrupt	Write BUF2I = 1 after reading BUF2I = 1
11		BUF3I	Message buffer 3 interrupt	Write BUF3I = 1 after reading BUF3I = 1
12		BUF4I	Message buffer 4 interrupt	Write BUF4I = 1 after reading BUF4I = 1
13		BUF5I	Message buffer 5 interrupt	Write BUF5I = 1 after reading BUF5I = 1
14		BUF6I	Message buffer 6 interrupt	Write BUF6I = 1 after reading BUF6I = 1
15		BUF7I	Message buffer 7 interrupt	Write BUF7I = 1 after reading BUF7I = 1
16		BUF8I	Message buffer 8 interrupt	Write BUF8I = 1 after reading BUF8I = 1
17		BUF9I	Message buffer 9 interrupt	Write BUF9I = 1 after reading BUF9I = 1
18		BUF10I	Message buffer 10 interrupt	Write BUF10I = 1 after reading BUF10I = 1
19		BUF11I	Message buffer 11 interrupt	Write BUF11I = 1 after reading BUF11I = 1
20		BUF12I	Message buffer 12 interrupt	Write BUF12I = 1 after reading BUF12I = 1
21		BUF13I	Message buffer 13 interrupt	Write BUF13I = 1 after reading BUF13I = 1
22		BUF14I	Message buffer 14 interrupt	Write BUF14I = 1 after reading BUF14I = 1
23		BUF15I	Message buffer 15 interrupt	Write BUF15I = 1 after reading BUF15I = 1
24		ERR_INT	Error interrupt	Write ERR_INT = 1 after reading ERR_INT = 1
25		BOFF_INT	Bus-off interrupt	Write BOFF_INT = 1 after reading BOFF_INT = 1
26		WAKE_INT	Wake-up interrupt	Write WAKE_INT = 1 after reading WAKE_INT = 1
27-63	Not used			

10.3.7 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 IACK registers within each interrupt controller, there are global L*n*IACK registers. A read from one of the global L*n*IACK registers returns the vector for the highest priority unmasked interrupt within a level for all interrupt controllers. There is no global SWIACK register. However, reading the SWIACK register from each interrupt controller returns the vector number of the highest priority unmasked request within that controller.

	7 6 4 3 0
Field	VECTOR
Reset	0000_0000
R/W	R
Address	See Table 10-2 and Table 10-3 for register offsets

Figure 10-10. Software and Level *n* IACK Registers (SWIACKR, L1IACK–L7IACK)

Table 10-15. 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.

10.4 Prioritization Between Interrupt Controllers

The interrupt controllers have a fixed priority, where INTC0 has the highest priority, and INTC1 has the lowest priority. If both interrupt controllers have active interrupts at the same level and priority, then the INTC0 interrupt will be serviced first. If INTC1 has an active interrupt that has a higher level or priority than the highest INTC0 interrupt, then the INTC1 interrupt will be serviced first.

10.5 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.

Each 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 each interrupt controller asserts the wake-up output signal, which is routed to the SCM where it is combined with the wakeup signals from the other interrupt controller and then to the PLL module to re-enable the device's clock trees and resume processing.

Chapter 11
Edge Port Module (EPORT)

11.1 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 11-1.

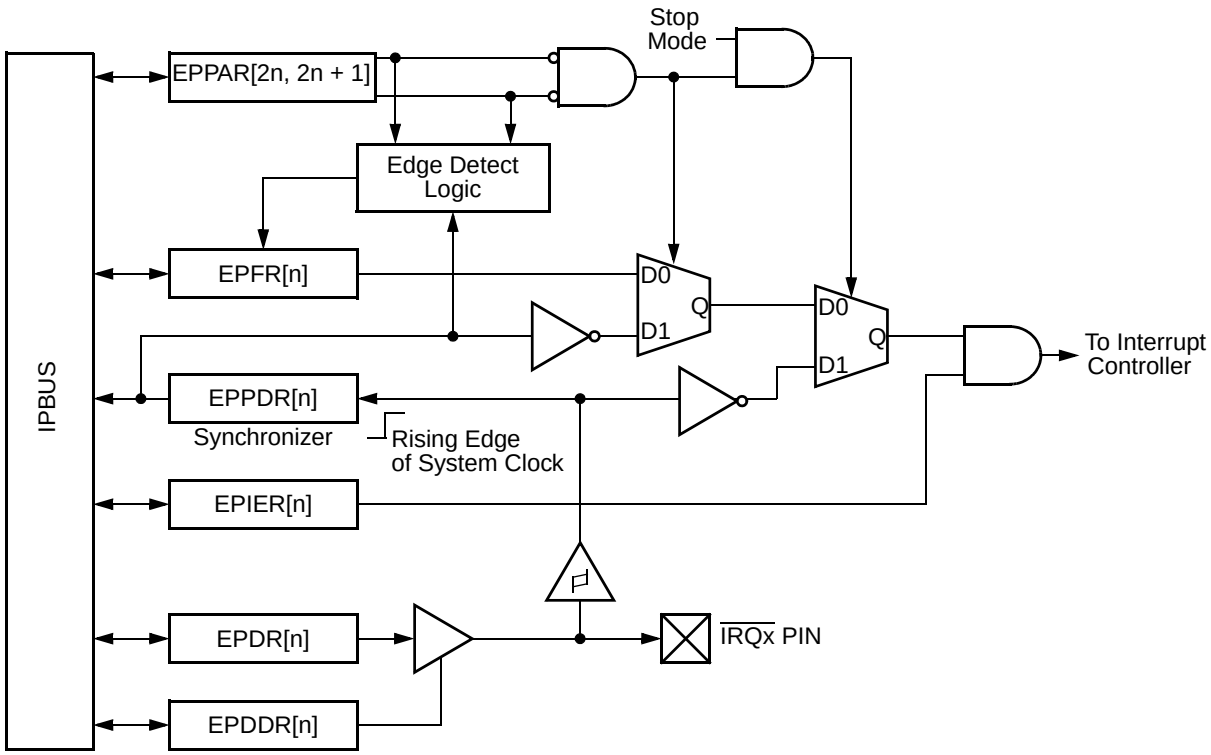


Figure 11-1. EPORT Block Diagram

11.2 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 11-1 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 11-1. 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.

11.3 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.

11.4 Memory Map and Registers

This subsection describes the memory map and register structure.

11.4.1 Memory Map

Refer to Table 11-2 for a description of the EPORT memory map. The EPORT has an IPSBAR offset for base address of 0x0013_0000.

Table 11-2. 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.

11.4.2 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.

11.4.2.1 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 11-2. EPORT Pin Assignment Register (EPPAR)

Table 11-3. 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.

11.4.2.2 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 11-3. EPORT Data Direction Register (EPDDR)

Table 11-4. 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.

11.4.2.3 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 11-4. EPORT Port Interrupt Enable Register (EPIER)

Table 11-5. 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.

11.4.2.4 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 11-5. EPORT Port Data Register (EPDR)

Table 11-6. 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.

11.4.2.5 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 11-6. EPORT Port Pin Data Register (EPPDR)

Table 11-7. 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 $\overline{\text{IRQ7}}$–$\overline{\text{IRQ1}}$. Writing to EPPDR has no effect, and the write cycle terminates normally. Reset does not affect EPPDR.
0	—	Reserved, should be cleared.

11.4.2.6 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 11-7. EPORT Port Flag Register (EPFR)

Table 11-8. 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 $\overline{\text{IRQx}}$ pin has been detected. 0 Selected edge for $\overline{\text{IRQx}}$ pin has not been detected.
0	—	Reserved, should be cleared.

Chapter 12
Chip Select Module

This chapter describes the chip select module, including the operation and programming model of the chip select registers, which include the chip select address, mask, and control registers.

NOTE

Unless otherwise noted, in this chapter, “clock” refers to the CLKOUT used for the bus.

12.1 Overview

The following list summarizes the key chip select features:

- Up to seven independent, user-programmable chip select signals ($\overline{CS}$[6:0]) that can interface with external SRAM, PROM, EPROM, EEPROM, Flash, and peripherals
- Address masking for 64-Kbyte to 4-Gbyte memory block sizes

12.2 Chip Select Module Signals

Table 12-1 lists signals used by the chip select module.

Table 12-1. Chip Select Module Signals

Signal	Description
Chip Selects ($\overline{CS}$[6:0])	Each $\overline{CS}n$ can be independently programmed for an address location as well as for masking, port size, read/write burst capability, wait-state generation, and internal/external termination. Only $\overline{CS0}$ is initialized at reset and may act as an external boot chip select to allow boot ROM to be at an external address space. Port size for $\overline{CS0}$ is configured by the logic levels of D[19:18] when $\overline{RSTO}$ negates and $\overline{RCON}$ is asserted.
Output Enable ($\overline{OE}$)	Interfaces to memory or to peripheral devices and enables a read transfer. It is asserted and negated on the falling edge of the clock. $\overline{OE}$ is asserted only when one of the chip selects matches for the current address decode.
Byte Strobes $\overline{BS}$[3:0]	These signals are individually programmed through the byte-enable mode bit, CSCR*n*[BEM], described in Section 12.4.1.3, “Chip Select Control Registers (CSCR0–CSCR6)”. These generated signals provide byte data select signals, which are decoded from the transfer size, A1, and A0 signals in addition to the programmed port size and burstability of the memory accessed, as Table 12-2 shows.

Table 12-2 shows the interaction of the byte-enable/byte-write enables with related signals.

Table 12-2. Byte Enables/Byte Write Enable Signal Settings

Transfer Size	Port Size	A1	A0	$\overline{BS3}$	$\overline{BS2}$	$\overline{BS1}$	$\overline{BS0}$
				D[31:24]	D[23:16]	D[15:8]	D[7:0]
Byte	8-bit	0	0	0	1	1	1
		0	1	0	1	1	1
		1	0	0	1	1	1
		1	1	0	1	1	1
	16-bit	0	0	0	1	1	1
		0	1	1	0	1	1
		1	0	0	1	1	1
		1	1	1	0	1	1
	32-bit	0	0	0	1	1	1
		0	1	1	0	1	1
		1	0	1	1	0	1
		1	1	1	1	1	0
Word	8-bit	0	0	0	1	1	1
		0	1	0	1	1	1
		1	0	0	1	1	1
		1	1	0	1	1	1
	16-bit	0	0	0	0	1	1
		1	0	0	0	1	1
	32-bit	0	0	0	0	1	1
		1	0	1	1	0	0
Longword	8-bit	0	0	0	1	1	1
		0	1	0	1	1	1
		1	0	0	1	1	1
		1	1	0	1	1	1
	16-bit	0	0	0	0	1	1
		1	0	0	0	1	1
	32-bit	0	0	0	0	0	0
Line	8-bit	0	0	0	1	1	1
		0	1	0	1	1	1
		1	0	0	1	1	1
		1	1	0	1	1	1
	16-bit	0	0	0	0	1	1
		1	0	0	0	1	1
	32-bit	0	0	0	0	0	0

12.3 Chip Select Operation

Each chip select has a dedicated set of registers for configuration and control.

- Chip select address registers (CSAR*n*) control the base address of the chip select. See Section 12.4.1.1, “Chip Select Address Registers (CSAR0–CSAR6)”.
- Chip select mask registers (CSMR*n*) provide 16-bit address masking and access control. See Section 12.4.1.2, “Chip Select Mask Registers (CSMR0–CSMR6)”.
- Chip select control registers (CSCR*n*) provide port size and burst capability indication, wait-state generation, and automatic acknowledge generation features. See Section 12.4.1.3, “Chip Select Control Registers (CSCR0–CSCR6)”.

$\overline{\text{CS0}}$ is a global chip select after reset and provides relocatable boot ROM capability.

12.3.1 General Chip Select Operation

When a bus cycle is initiated, the device first compares its address with the base address and mask configurations programmed for chip selects 0–6 (configured in CSCR0–CSCR6) and DRAM blocks 0 and 1 (configured in DACR0 and DACR1). If the driven address matches a programmed chip select or DRAM block, the appropriate chip select is asserted or the DRAM block is selected using the specifications programmed in the respective configuration register. Otherwise, the following occurs:

- If the address and attributes do not match in CSAR or DACR, the device runs an external burst-inhibited bus cycle with a default of external termination on a 32-bit port.
- Should an address and attribute match in multiple CSCRs, the matching chip select signals are driven; however, the chip select signals are driven during an external burst-inhibited bus cycle with external termination on a 32-bit port.
- If the address and attribute match both DACRs or a DACR and a CSAR, the operation is undefined.

Table 12-3 shows the type of access as a function of match in the CSARs and DACRs.

Table 12-3. Accesses by Matches in CSARs and DACRs

Number of CSCR Matches	Number of DACR Matches	Type of Access
0	0	External
1	0	Defined by CSAR
Multiple	0	External, burst-inhibited, 32-bit
0	1	Defined by DACRs
1	1	Undefined
Multiple	1	Undefined
0	Multiple	Undefined
1	Multiple	Undefined
Multiple	Multiple	Undefined

12.3.1.1 8-, 16-, and 32-Bit Port Sizing

Static bus sizing is programmable through the port size bits, CSCR[PS]. See Section 12.4.1.3, “Chip Select Control Registers (CSCR0–CSCR6)” for more information. Figure 12-1 shows the correspondence

between the data bus and the external byte strobe control lines ($\overline{BS}$[3:0]). Note that all byte lanes are driven, although the state of unused byte lanes is undefined.

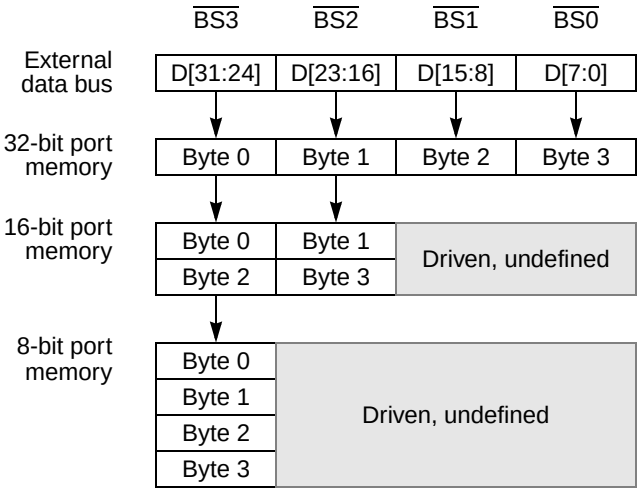


Figure 12-1. Connections for External Memory Port Sizes

12.3.1.2 External Boot Chip Select Operation

$\overline{CS0}$, the external boot chip select, allows address decoding for boot ROM before system initialization. Its operation differs from other external chip select outputs after system reset.

After system reset, $\overline{CS0}$ is asserted for every external access. No other chip select can be used until the valid bit, CSMR0[V], is set, at which point $\overline{CS0}$ functions as configured and $\overline{CS}$[6:1] can be used. At reset, the port size function of the external boot chip select is determined by the logic levels of the inputs on D[19:18]. Table 12-4 and Table 12-4 list the various reset encodings for the configuration signals multiplexed with D[19:18].

Table 12-4. D[19:18] External Boot Chip Select Configuration

D[19:18]	Boot Device/Data Port Size
00	Internal (32-bit)
01	External (16-bit)
10	External (8-bit)
11	External (32-bit)

Provided the required address range is in the chip select address register (CSAR0), $\overline{CS0}$ can be programmed to continue decoding for a range of addresses after the CSMR0[V] is set, after which the external boot chip select can be restored only by a system reset.

12.4 Chip Select Registers

Table 12-5 shows the chip select register memory map. Reading reserved locations returns zeros.

Table 12-5. Chip Select Registers

IPSBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x00_0080	Chip select address register—bank 0 (CSAR0) [p. 12-6]		Reserved[1]	
0x00_0084	Chip select mask register—bank 0 (CSMR0) [p. 12-6]			
0x00_0088	Reserved[1]		Chip select control register—bank 0 (CSCR0) [p. 12-7]	
0x00_008C	Chip select address register—bank 1 (CSAR1) [p. 12-6]		Reserved[1]	
0x00_0090	Chip select mask register—bank 1 (CSMR1) [p. 12-6]			
0x00_0094	Reserved[1]		Chip select control register—bank 1 (CSCR1) [p. 12-7]	
0x00_0098	Chip select address register—bank 2 (CSAR2) [p. 12-6]		Reserved[1]	
0x00_009C	Chip select mask register—bank 2 (CSMR2) [p. 12-6]			
0x00_00A0	Reserved[1]		Chip select control register—bank 2 (CSCR2) [p. 12-7]	
0x00_00A4	Chip select address register—bank 3 (CSAR3) [p. 12-6]		Reserved[1]	
0x00_00A8	Chip select mask register—bank 3 (CSMR3) [p. 12-6]			
0x00_00AC	Reserved[1]		Chip select control register—bank 3 (CSCR3) [p. 12-7]	
0x00_00B0	Chip select address register—bank 4 (CSAR4) [p. 12-6]		Reserved[1]	
0x00_00B4	Chip select mask register—bank 4 (CSMR4) [p. 12-6]			
0x00_00B8	Reserved[1]		Chip select control register—bank 4 (CSCR4) [p. 12-7]	
0x00_00BC	Chip select address register—bank 5 (CSAR5) [p. 12-6]		Reserved[1]	
0x00_00C0	Chip select mask register—bank 5 (CSMR5) [p. 12-6]			
0x00_00C4	Reserved[1]		Chip select control register—bank 5 (CSCR5) [p. 12-7]	
0x00_00C8	Chip select address register—bank 6 (CSAR6) [p. 12-6]		Reserved[1]	
0x00_00CC	Chip select mask register—bank 6 (CSMR6) [p. 12-6]			
0x00_00D0	Reserved[1]		Chip select control register—bank 6 (CSCR6) [p. 12-7]	

[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.

12.4.1 Chip Select Module Registers

The chip select module is programmed through the chip select address registers (CSAR0–CSAR6), chip select mask registers (CSMR0–CSMR6), and the chip select control registers (CSCR0–CSCR6).

12.4.1.1 Chip Select Address Registers (CSAR0–CSAR6)

The CSARs, Figure 12-2, specify the chip select base addresses.

	15 … 0
Field	BA
Reset	Uninitialized
R/W	R/W
Address	0x080 (CSAR0); 0x08C (CSAR1); 0x098 (CSAR2); 0x0A4 (CSAR3); 0x0B0 (CSAR4); 0x0BC (CSAR5); 0x0C8 (CSAR6)

Figure 12-2. Chip Select Address Registers (CSAR*n*)

Table 12-6 describes CSAR[BA].

Table 12-6. CSAR*n* Field Description

Bits	Name	Description
15–0	BA	Base address. Defines the base address for memory dedicated to chip select $\overline{CS}$[6:0]. BA is compared to bits 31–16 on the internal address bus to determine if chip select memory is being accessed.

12.4.1.2 Chip Select Mask Registers (CSMR0–CSMR6)

The CSMRs, Figure 12-3, are used to specify the address mask and allowable access types for the respective chip selects.

	31 … 16	15 … 9	8	7	6	5	4	3	2	1	0
Field	BAM	—	WP	—	AM	C/I	SC	SD	UC	UD	V
Reset	Unitialized										0
R/W	R/W										
Addr	0x084 (CSMR0); 0x090 (CSMR1); 0x09C (CSMR2); 0x0A8 (CSMR3); 0x0B4 (CSMR4); 0x0C0 (CSMR5); 0x0CC (CSMR6)										

Figure 12-3. Chip Select Mask Registers (CSMR*n*)

Table 12-7 describes CSMR fields.

Table 12-7. CSMR*n* Field Descriptions

Bits	Name	Description
31–16	BAM	Base address mask. Defines the chip select block by masking address bits. Setting a BAM bit causes the corresponding CSAR bit to be ignored in the decode. 0 Corresponding address bit is used in chip select decode. 1 Corresponding address bit is a don't care in chip select decode. The block size for $\overline{CS}$[6:0] is 2^n where n = (number of bits set in respective CSMR[BAM]) + 16. So, if CSAR0 = 0x0000 and CSMR0[BAM] = 0x0001, $\overline{CS0}$ addresses a 128-Kbyte (2^{17} byte) range from 0x0000–0x1_FFFF. Likewise, for $\overline{CS0}$ to access 32 Mbytes (2^{25} bytes) of address space starting at location 0x0000, and for $\overline{CS1}$ to access 16 Mbytes (2^{24} bytes) of address space starting after the $\overline{CS0}$ space, then CSAR0 = 0x0000, CSMR0[BAM] = 0x01FF, CSAR1 = 0x0200, and CSMR1[BAM] = 0x00FF.
8	WP	Write protect. Controls write accesses to the address range in the corresponding CSAR. Attempting to write to the range of addresses for which CSAR*n*[WP] = 1 results in the appropriate chip select not being selected. No exception occurs. 0 Both read and write accesses are allowed. 1 Only read accesses are allowed.
7	—	Reserved, should be cleared.
6	AM	Alternate master. When AM = 0 during a DMA access, SC, SD, UC, and UD are don't cares in the chip select decode.
5–1	C/I, SC, SD, UC, UD	Address space mask bits. These bits determine whether the specified accesses can occur to the address space defined by the BAM for this chip select. C/I CPU space and 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 0 The address space assigned to this chip select is available to the specified access type. 1 The address space assigned to this chip select is not available (masked) to the specified access type. If this address space is accessed, chip select is not activated and a regular external bus cycle occurs. Note that if AM = 0, SC, SD, UC, and UD are ignored in the chip select decode on DMA access.
0	V	Valid bit. Indicates whether the corresponding CSAR, CSMR, and CSCR contents are valid. Programmed chip selects do not assert until V is set (except for $\overline{CS0}$, which acts as the global chip select). Reset clears each CSMR*n*[V]. 0 Chip select invalid 1 Chip select valid

12.4.1.3 Chip Select Control Registers (CSCR0–CSCR6)

Each CSCR, shown in Figure 12-4, controls the auto-acknowledge, port size, burst capability, and activation of each chip select. Note that to support the external boot chip select, $\overline{CS0}$, the CSCR0 reset values differ from the other CSCRs. $\overline{CS0}$ allows address decoding for boot ROM before system initialization.

	15	14	13	10	9	8	7	6	5	4	3	2	0
Field	—		WS		—	AA	PS1	PS0	BEM	BSTR	BSTW	—	
Reset: CSCR0	—		11_11		—	1	D19	D18	—	—			
Reset: Other CSCRs	Uninitialized												
R/W	R/W												
Address	0x08A (CSCR0); 0x096 (CSCR1); 0x0A2 (CSCR2); 0x0AE (CSCR3); 0x0BA (CSCR4); 0x0C6 (CSCR5); 0x0D2 (CSCR6)												

Figure 12-4. Chip Select Control Registers (CSCR*n*)

Table 12-8 describes CSCR*n* fields.

Table 12-8. CSCR*n* Field Descriptions

Bits	Name	Description
15–14	—	Reserved, should be cleared.
13–10	WS	Wait states. The number of wait states inserted before an internal transfer acknowledge is generated (WS = 0 inserts zero wait states, WS = 0xF inserts 15 wait states). If AA = 0, $\overline{TA}$ must be asserted by the external system regardless of the number of wait states generated. In that case, the external transfer acknowledge ends the cycle. An external $\overline{TA}$ supercedes the generation of an internal $\overline{TA}$.
9	—	Reserved, should be cleared.
8	AA	Auto-acknowledge enable. Determines the assertion of the internal transfer acknowledge for accesses specified by the chip select address. 0 No internal $\overline{TA}$ is asserted. Cycle is terminated externally. 1 Internal $\overline{TA}$ is asserted as specified by WS. Note that if AA = 1 for a corresponding $\overline{CS}n$ and the external system asserts an external $\overline{TA}$ before the wait-state countdown asserts the internal $\overline{TA}$, the cycle is terminated. Burst cycles increment the address bus between each internal termination.
7–6	PS	Port size. Specifies the width of the data associated with each chip select. It determines where data is driven during write cycles and where data is sampled during read cycles. See Section 12.3.1.1, “8-, 16-, and 32-Bit Port Sizing”. 00 32-bit port size. Valid data sampled and driven on D[31:0] 01 8-bit port size. Valid data sampled and driven on D[31:24] 1x 16-bit port size. Valid data sampled and driven on D[31:16]
5	BEM	Byte enable mode. Specifies the byte enable operation. Certain SRAMs have byte enables that must be asserted during reads as well as writes. BEM can be set in the relevant CSCR to provide the appropriate mode of byte enable in support of these SRAMs. 0 $\overline{BS}$ is not asserted for read. $\overline{BS}$ is asserted for data write only. 1 $\overline{BS}$ is asserted for read and write accesses.
4	BSTR	Burst read enable. Specifies whether burst reads are used for memory associated with each $\overline{CS}n$. 0 Data exceeding the specified port size is broken into individual, port-sized non-burst reads. For example, a longword read from an 8-bit port is broken into four 8-bit reads. 1 Enables data burst reads larger than the specified port size, including longword reads from 8- and 16-bit ports, word reads from 8-bit ports, and line reads from 8-, 16-, and 32-bit ports.

Table 12-8. CSCR*n* Field Descriptions (continued)

Bits	Name	Description
3	BSTW	Burst write enable. Specifies whether burst writes are used for memory associated with each $\overline{CS}n$. 0 Break data larger than the specified port size into individual port-sized, non-burst writes. For example, a longword write to an 8-bit port takes four byte writes. 1 Enables burst write of data larger than the specified port size, including longword writes to 8 and 16-bit ports, word writes to 8-bit ports and line writes to 8-, 16-, and 32-bit ports.
2–0	—	Reserved, should be cleared.

Chapter 13
External Interface Module (EIM)

This chapter describes data-transfer operations, error conditions, and reset operations. Chapter 15, "Synchronous DRAM Controller Module," describes DRAM cycles.

NOTE

Unless otherwise noted, in this chapter, "clock" refers to the CLKOUT used for the bus.

13.1 Features

The following list summarizes bus operation features:

- Up to 24 bits of address and 32 bits of data
- Access 8-, 16-, and 32-bit data port sizes
- Generates byte, word, longword, and line-size transfers
- Burst and burst-inhibited transfer support
- Optional internal termination for external bus cycles

13.2 Bus and Control Signals

Table 13-1 summarizes the bus signals described in Chapter 14, "Signal Descriptions".

Table 13-1. ColdFire Bus Signal Summary

Signal Name	Description	I/O	CLKOUT Edge
A[23:0]	Address bus	O	Rising
$\overline{BS}$ [1]	Byte selects	O	Falling
$\overline{CS}$[6:0] [1]	Chip selects	O	Falling
D[31:0]	Data bus	I/O	Rising
$\overline{OE}$ [1]	Output enable	O	Falling
R/$\overline{W}$	Read/write	O	Rising
SIZ[1:0]	Transfer size	O	Rising
$\overline{TA}$	Transfer acknowledge	I	Rising
$\overline{TIP}$	Transfer in progress	O	Rising
$\overline{TS}$	Transfer start	O	Rising

[1] These signals change after the falling edge. In the *Electrical Specifications*, these signals are specified off of the rising edge because CLKIN is squared up internally.

13.3 Bus Characteristics

The device uses its system clock to generate CLKOUT. Therefore, the external bus operates at the same speed as the bus clock rate, where all bus operations are synchronous to the rising edge of CLKOUT, and some of the bus control signals ($\overline{BS}$, $\overline{OE}$, and $\overline{CSn}$,) are synchronous to the falling edge, shown in Figure 13-1. Bus characteristics may differ somewhat for interfacing with external DRAM.

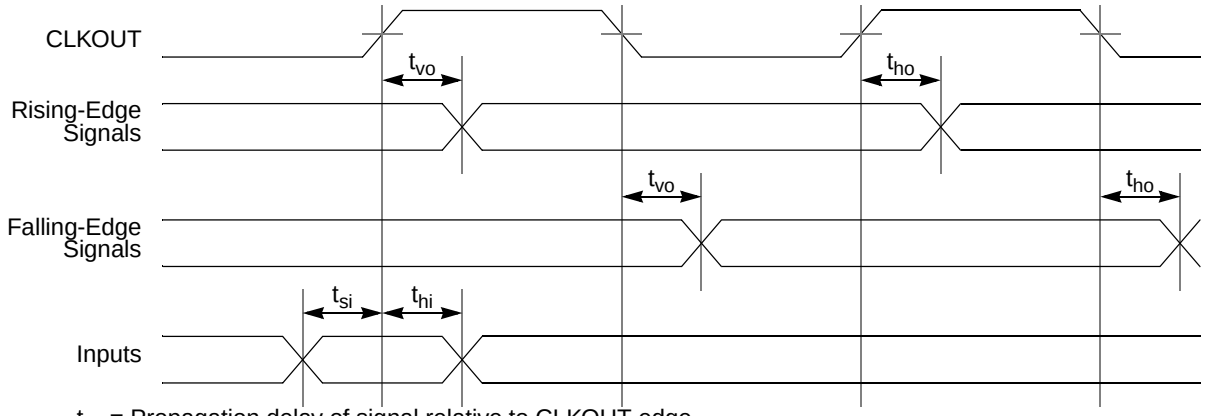


Figure 13-1. Signal Relationship to CLKOUT for Non-DRAM Access

13.4 Data Transfer Operation

Data transfers between the processor and other devices involve the following signals:

- Address bus (A[23:0])
- Data bus (D[31:0])
- Control signals ($\overline{TS}$ and $\overline{TA}$)
- $\overline{CSn}$, $\overline{OE}$, $\overline{BS}$
- Attribute signals (R/$\overline{W}$, SIZ, and $\overline{TIP}$)

The address bus, write data, $\overline{TS}$, and all attribute signals change on the rising edge of CLKOUT. Read data is latched into the processor on the rising edge of CLKOUT.

The bus supports byte, word, and longword operand transfers and allows accesses to 8-, 16-, and 32-bit data ports. Aspects of the transfer, such as the port size, the number of wait states for the external slave being accessed, and whether internal transfer termination is enabled, can be programmed in the chip-select control registers (CSCRs) and the DRAM control registers (DACRs).

Figure 13-2 shows the byte lanes that external memory should be connected to and the sequential transfers if a longword is transferred for three port sizes. For example, an 8-bit memory should be connected to D[31:24] ($\overline{BS3}$). A longword transfer takes four transfers on D[31:24], starting with the MSB and going to the LSB.

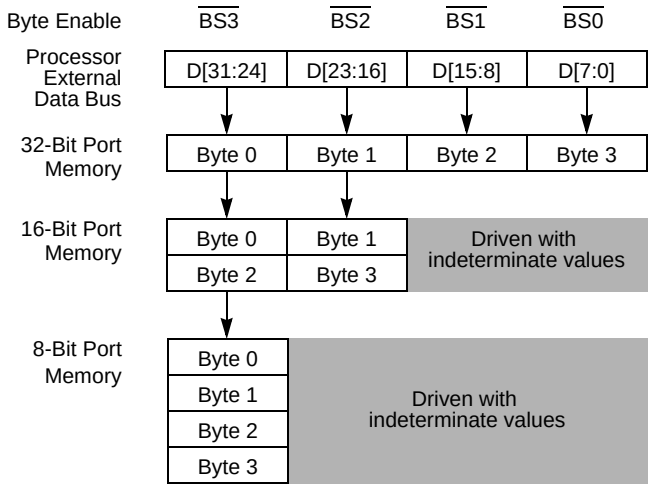


Figure 13-2. Connections for External Memory Port Sizes

The timing relationship of chip selects ($\overline{CS}$[7:0]), byte selects ($\overline{BS}$[3:0]), and output enable ($\overline{OE}$) with respect to CLKOUT is similar in that all transitions occur during the low phase of CLKOUT. However, due to differences in on-chip signal routing, signals may not assert simultaneously.

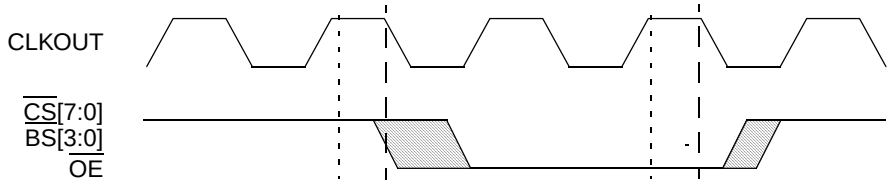


Figure 13-3. Chip-Select Module Output Timing Diagram

13.4.1 Bus Cycle Execution

When a bus cycle is initiated, the device first compares the address of that bus cycle with the base address and mask configurations programmed for chip selects 0–7 (configured in CSCR0–CSCR7) and DRAM block 0 and 1 address and control registers (configured in DACR0 and DACR1). If the driven address compares with one of the programmed chip selects or DRAM blocks, the appropriate chip select is asserted or the DRAM block is selected using the specifications programmed by the user in the respective configuration register. Otherwise, the following occurs:

- If the address and attributes do not match in CSCR or DACR, the processor runs an external burst-inhibited bus cycle with a default of external termination on a 32-bit port.
- Should an address and attribute match in multiple CSCRs, the matching chip-select signals are driven; however, the processor runs an external burst-inhibited bus cycle with external termination on a 32-bit port.
- Should an address and attribute match both DACRs or a DACR and a CSCR, the operation is undefined.

Table 13-2 shows the type of access as a function of match in the CSCRs and DACRs.

Table 13-2. Accesses by Matches in CSCRs and DACRs

Number of CSCR Matches	Number of DACR Matches	Type of Access
0	0	External
1	0	Defined by CSCR
Multiple	0	External, burst-inhibited, 32-bit
0	1	Defined by DACRs
1	1	Undefined
Multiple	1	Undefined
0	Multiple	Undefined
1	Multiple	Undefined
Multiple	Multiple	Undefined

Basic operation of the bus is a three-clock bus cycle:

1. During the first clock, the address, attributes, and $\overline{TS}$ are driven.
2. Data and $\overline{TA}$ are sampled during the second clock of a bus-read cycle. During a read, the external device provides data and is sampled at the rising edge at the end of the second bus clock. This data is concurrent with $\overline{TA}$, which is also sampled at the rising edge of the clock.

   During a write, the ColdFire device drives data from the rising clock edge at the end of the first clock to the rising clock edge at the end of the bus cycle. Wait states can be added between the first and second clocks by delaying the assertion of $\overline{TA}$. $\overline{TA}$ can be configured to be generated internally through the CSCRs. If $\overline{TA}$ is not generated internally, the system must provide it externally.
3. The last clock of the bus cycle uses what would be an idle clock between cycles to provide hold time for address, attributes and write data. Figure 13-6 and Figure 13-8 show the basic read and write operations.

13.4.2 Data Transfer Cycle States

The data transfer operation is controlled by an on-chip state machine. Each bus clock cycle is divided into two states. Even states occur when CLKOUT is high and odd states occur when CLKOUT is low. The state transition diagram for basic and fast termination read and write cycles are shown in Figure 13-4.

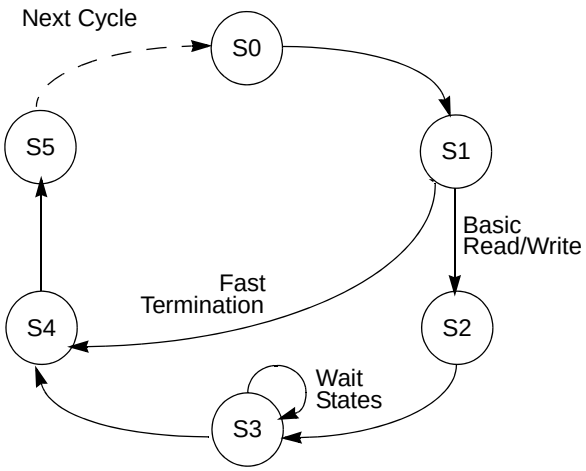


Figure 13-4. Data Transfer State Transition Diagram

Table 13-3 describes the states as they appear in subsequent timing diagrams.

Table 13-3. Bus Cycle States

State	Cycle	CLKOUT	Description
S0	All	High	The read or write cycle is initiated in S0. On the rising edge of CLKOUT, the device places a valid address on the address bus and drives R/$\overline{W}$ high for a read and low for a write, if it is not already in the appropriate state. The processor asserts $\overline{TIP}$, SIZ[1:0], and $\overline{TS}$ on the rising edge of CLKOUT.
S1	All	Low	The appropriate $\overline{CS}n$, $\overline{BS}$, and $\overline{OE}$ signals assert on the CLKOUT falling edge.
	Fast Termination		$\overline{TA}$ must be asserted during S1. Data is made available by the external device and is sampled on the rising edge of CLKOUT with $\overline{TA}$ asserted.
S2	Read/write (skipped fast termination)	High	$\overline{TS}$ is negated on the rising edge of CLKOUT in S2.
	Write		The data bus is driven out of high impedance as data is placed on the bus on the rising edge of CLKOUT.
S3	Read/write (skipped for fast termination)	Low	The processor waits for $\overline{TA}$ assertion. If $\overline{TA}$ is not sampled as asserted before the rising edge of CLKOUT at the end of the first clock cycle, the processor inserts wait states (full clock cycles) until $\overline{TA}$ is sampled as asserted.
	Read		Data is made available by the external device on the falling edge of CLKOUT and is sampled on the rising edge of CLKOUT with $\overline{TA}$ asserted.
S4	All	High	The external device should negate $\overline{TA}$.
	Read (including fast-termination)		The external device can stop driving data after the rising edge of CLKOUT. However data could be driven through the end of S5.

Table 13-3. Bus Cycle States (continued)

State	Cycle	CLKOUT	Description
S5	S5	Low	$\overline{CS}$, $\overline{BS}$, and $\overline{OE}$ are negated on the CLKOUT falling edge of S5. The processor stops driving address lines and R/$\overline{W}$ on the rising edge of CLKOUT, terminating the read or write cycle. At the same time, the processor negates $\overline{TIP}$, and SIZ[1:0] on the rising edge of CLKOUT. Note that the rising edge of CLKOUT may be the start of S0 for the next access cycle.
	Read		The external device stops driving data between S4 and S5.
	Write		The data bus returns to high impedance on the rising edge of CLKOUT. The rising edge of CLKOUT may be the start of S0 for the next access.

NOTE

An external device has at most two CLKOUT cycles after the start of S4 to three-state the data bus. This applies to basic read cycles, fast termination cycles, and the last transfer of a burst.

13.4.3 Read Cycle

During a read cycle, the device receives data from memory or from a peripheral device. Figure 13-5 is a read cycle flowchart.

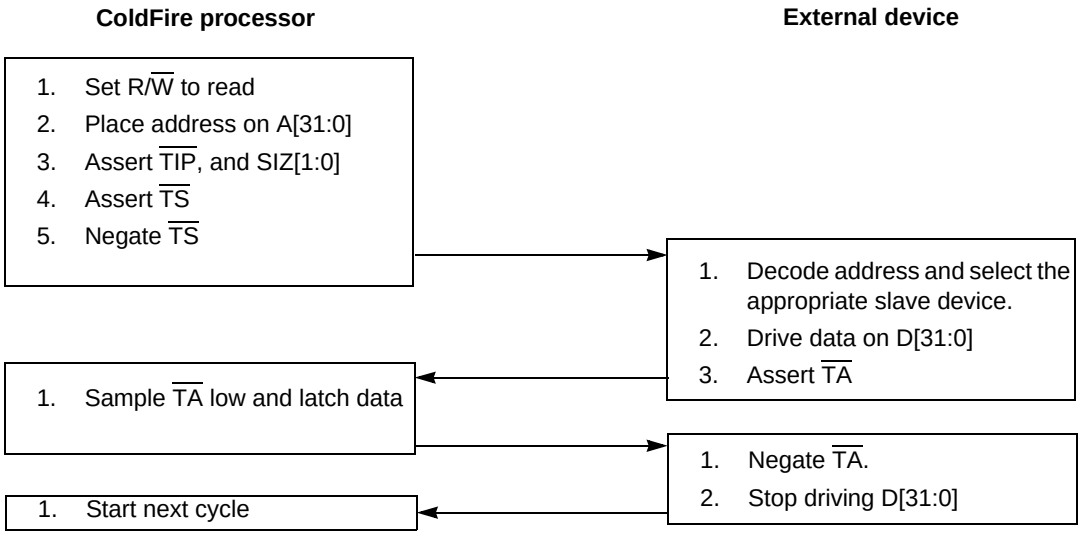


Figure 13-5. Read Cycle Flowchart

The read cycle timing diagram is shown in Figure 13-6.

NOTE

In the following timing diagrams, $\overline{TA}$ waveforms apply for chip selects programmed to enable either internal or external termination. $\overline{TA}$ assertion should look the same in either case.

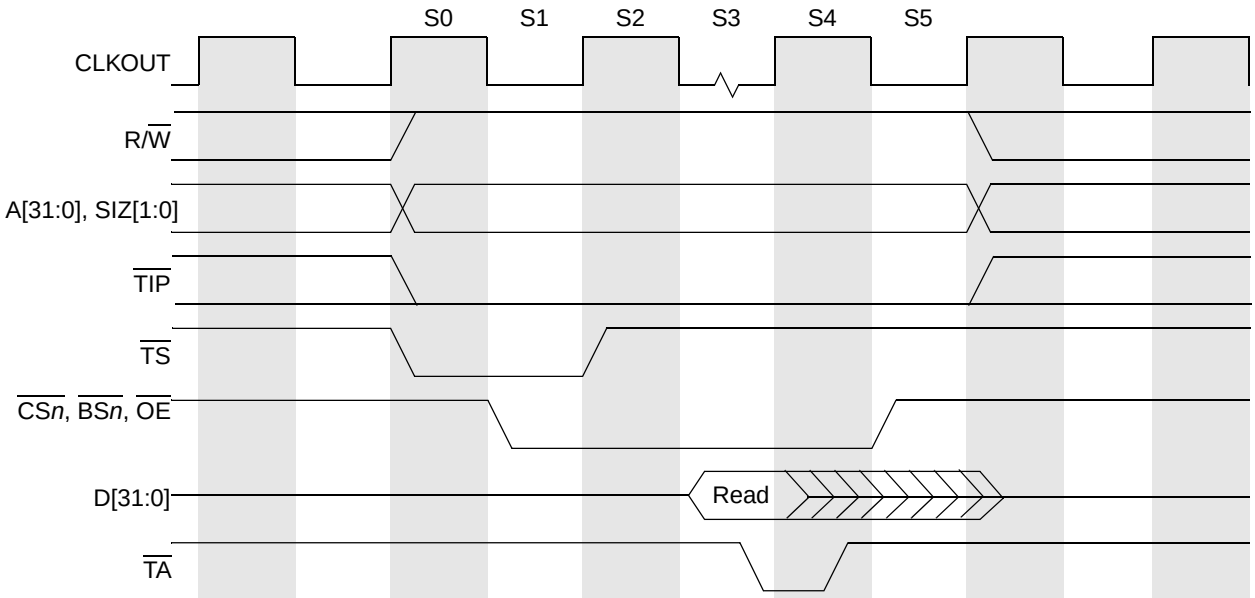


Figure 13-6. Basic Read Bus Cycle

Note the following characteristics of a basic read:

- In S3, data is made available by the external device on the falling edge of CLKOUT and is sampled on the rising edge of CLKOUT with TA asserted.
- In S4, the external device can stop driving data after the rising edge of CLKOUT. However data could be driven up to S5.
- For a read cycle, the external device stops driving data between S4 and S5.

States are described in Table 13-3.

13.4.4 Write Cycle

During a write cycle, the processor sends data to the memory or to a peripheral device. The write cycle flowchart is shown in Figure 13-7.

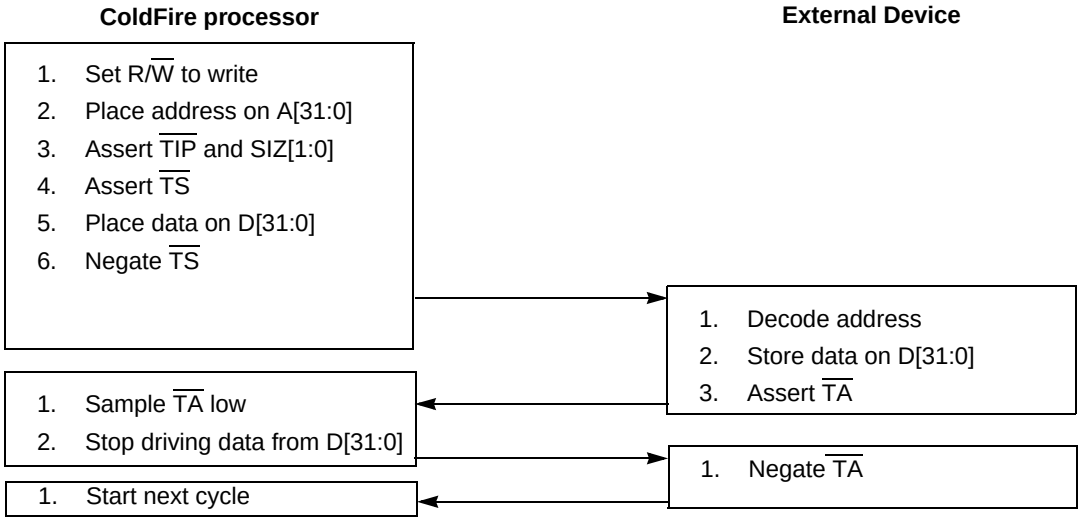


Figure 13-7. Write Cycle Flowchart

The write cycle timing diagram is shown in Figure 13-8.

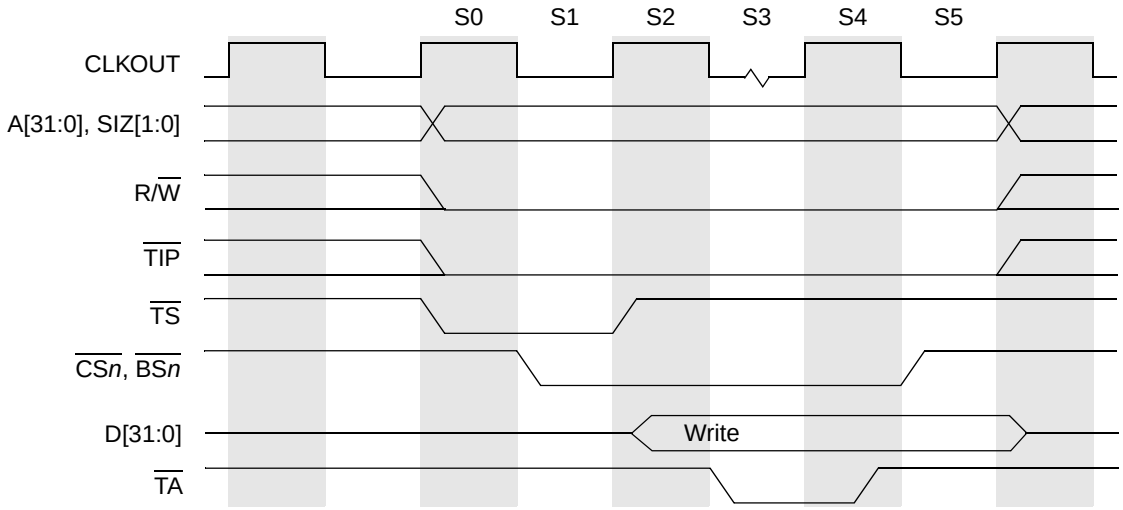


Figure 13-8. Basic Write Bus Cycle

Table 13-3 describes the six states of a basic write cycle.

13.4.5 Fast Termination Cycles

Two clock cycle transfers are supported on the external bus. In most cases, this is impractical to use in a system because the termination must take place in the same half-clock during which TS is asserted. As this is atypical, it is not referred to as the zero-wait-state case but is called the fast-termination case. Fast termination cycles occur when the external device or memory asserts TA less than one clock after TS is asserted. This means that the processor samples TA on the rising edge of the second cycle of the bus transfer. Figure 13-9 shows a read cycle with fast termination. Note that fast termination cannot be used with internal termination.

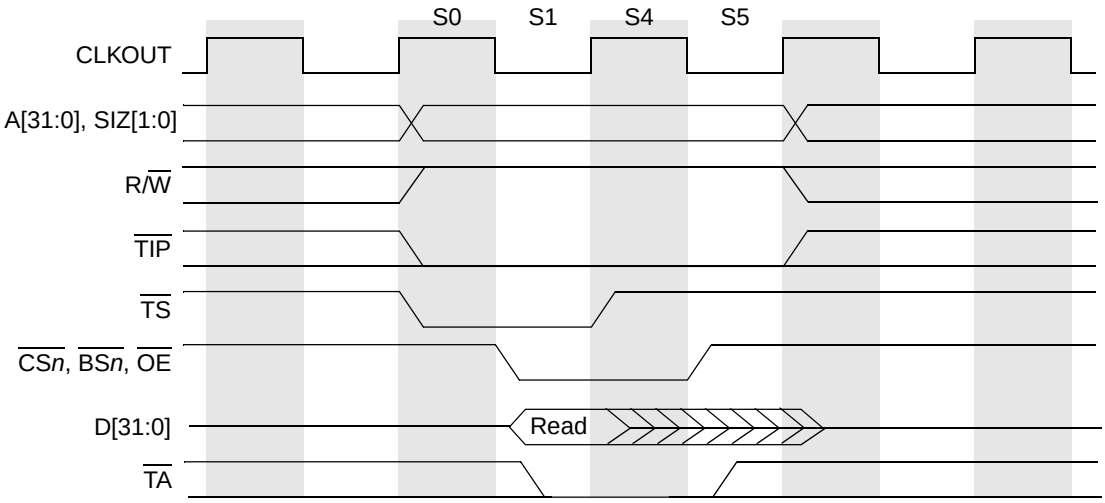


Figure 13-9. Read Cycle with Fast Termination

Figure 13-10 shows a write cycle with fast termination.

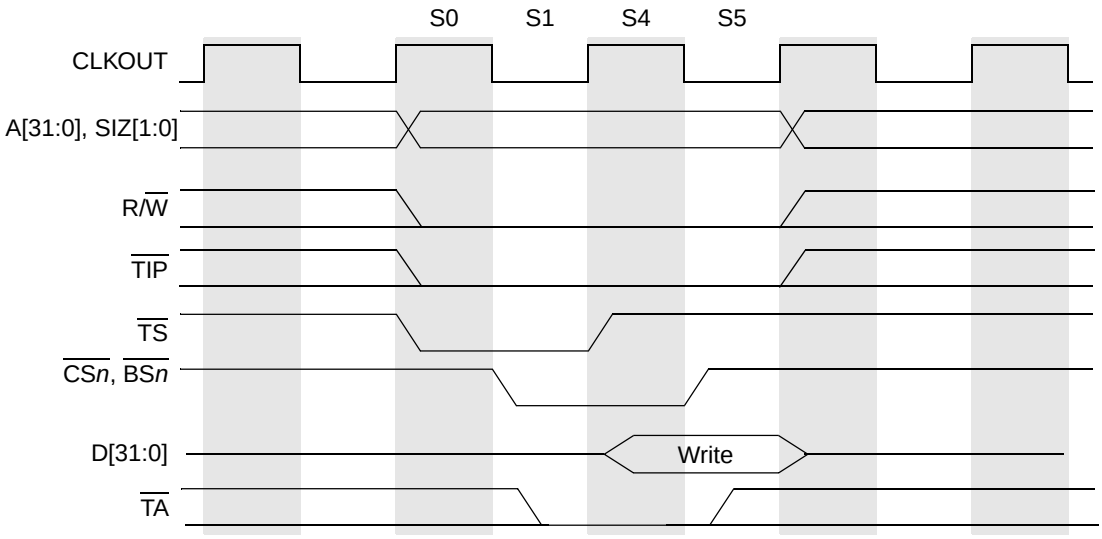


Figure 13-10. Write Cycle with Fast Termination

13.4.6 Back-to-Back Bus Cycles

The processor runs back-to-back bus cycles whenever possible. For example, when a longword read is started on a word-size bus, the processor performs two back-to-back word read accesses. Back-to-back accesses are distinguished by the continuous assertion of TIP throughout the cycle. Figure 13-11 shows a read back-to-back with a write.

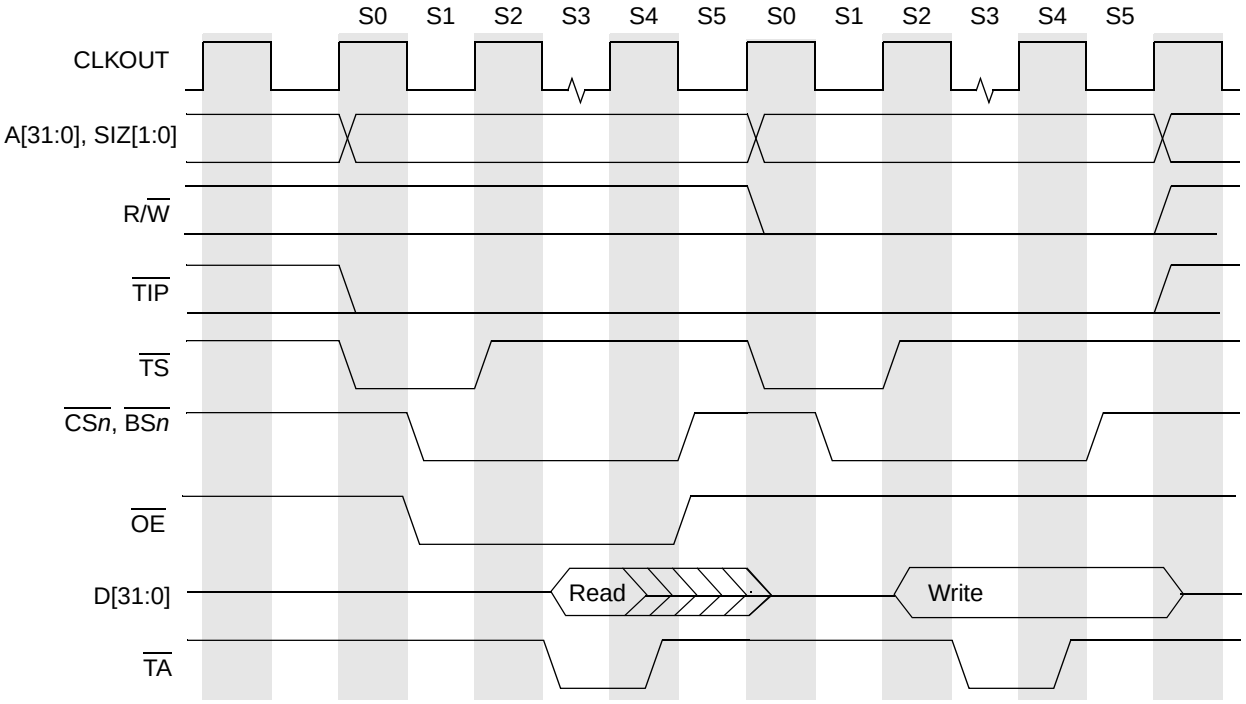


Figure 13-11. Back-to-Back Bus Cycles

Basic read and write cycles are used to show a back-to-back cycle, but there is no restriction as to the type of operations to be placed back to back. The initiation of a back-to-back cycle is not user definable.

13.4.7 Burst Cycles

The processor can be programmed to initiate burst cycles if its transfer size exceeds the size of the port it is transferring to. For example, a word transfer to an 8-bit port would take a 2-byte burst cycle. A line transfer to a 32-bit port would take a 4-longword burst cycle.

The external bus can support 2-1-1-1 burst cycles to maximize cache performance and optimize DMA transfers. A user can add wait states by delaying termination of the cycle. The initiation of a burst cycle is encoded on the size pins. For burst transfers to smaller port sizes, SIZ[1:0] indicates the size of the entire transfer. For example, if the processor writes a longword to an 8-bit port, SIZ[1:0] = 00 for the first byte transfer and does not change.

The CSCRs can be used to enable bursting for reads, writes, or both. Processor memory space can be declared burst-inhibited for reads and writes by clearing the appropriate CSCR*n*[BSTR,BSTW]. A line access to a burst-inhibited region first accesses the processor bus encoded as a line access. The SIZ[1:0] encoding does not exceed the programmed port size. The address changes if internal termination is used but does not change if external termination is used, as shown in Figure 13-12 and Figure 13-13.

13.4.7.1 Line Transfers

A line is a 16-byte-aligned, 16-byte value. Despite the alignment, a line access may not begin on the aligned address; therefore, the bus interface supports line transfers on multiple address boundaries. Table 13-4 shows allowable patterns for line accesses.

Table 13-4. Allowable Line Access Patterns

A[3:2]	Longword Accesses
00	0–4–8–C
01	4–8–C–0
10	8–C–0–4
11	C–0–4–8

13.4.7.2 Line Read Bus Cycles

Figure 13-12 and Figure 13-13 show a line access read with zero wait states. The access starts like a basic read bus cycle with the first data transfer sampled on the rising edge of S4, but the next pipelined burst data is sampled a cycle later on the rising edge of S6. Each subsequent pipelined data burst is single cycle until the last one, which can be held for up to two CLKOUT cycles after $\overline{TA}$ is asserted. Note that $\overline{CS}n$ are asserted throughout the burst transfer. This example shows the timing for external termination, which differs from the internal termination example in Figure 13-13 only in that the address lines change only at the beginning (assertion of $\overline{TS}$ and $\overline{TIP}$) and end (negation of $\overline{TIP}$) of the transfer.

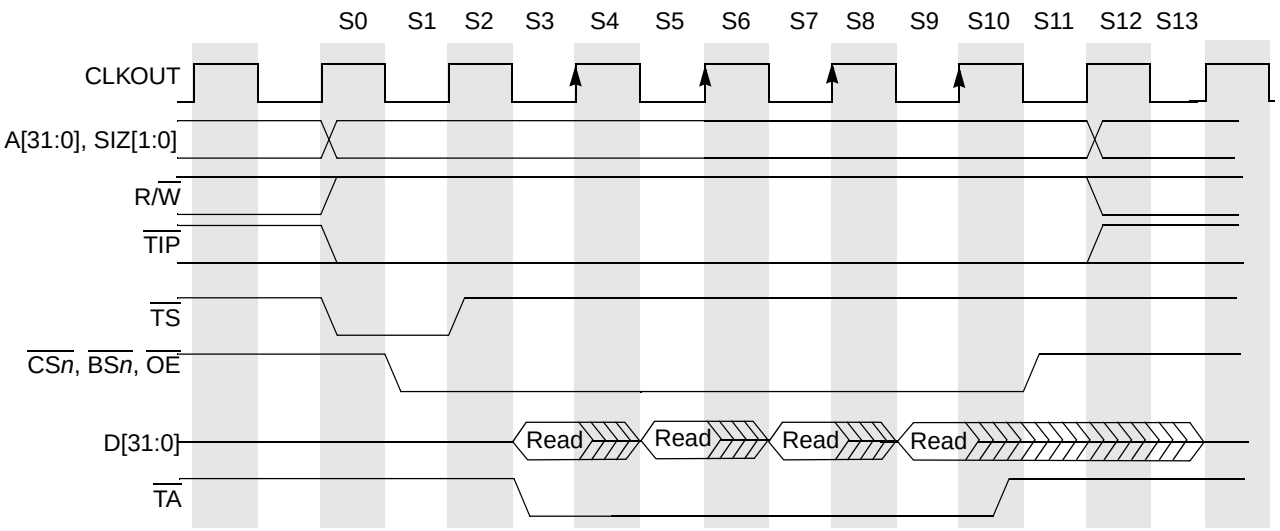


Figure 13-12. Line Read Burst (2-1-1-1), External Termination

Figure 13-13 shows timing when internal termination is used.

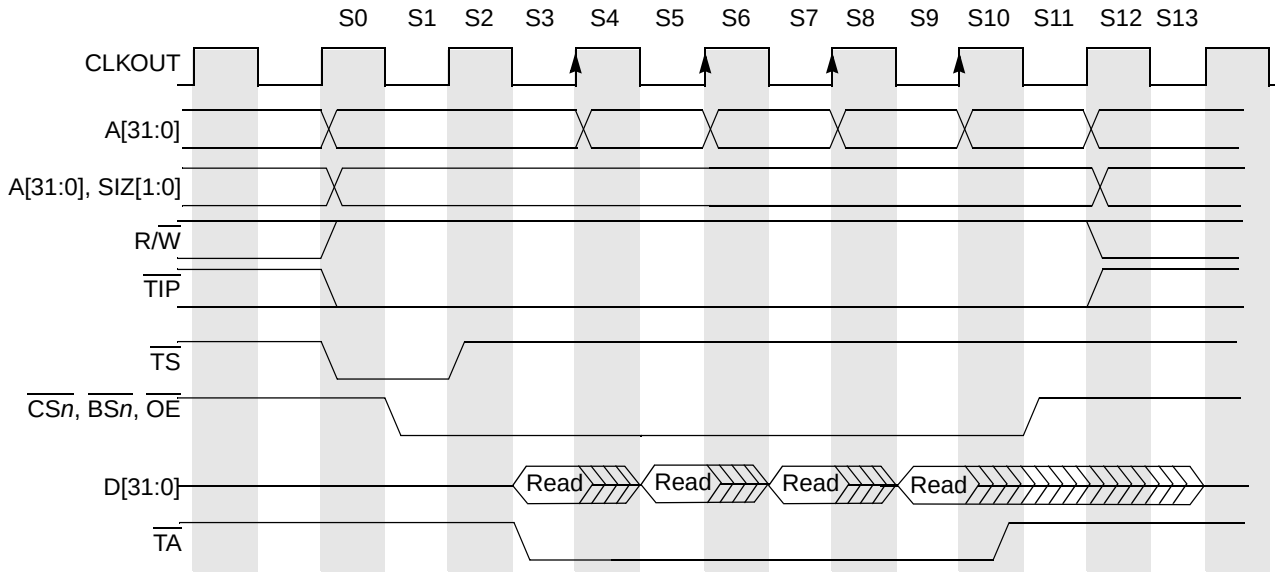


Figure 13-13. Line Read Burst (2-1-1-1), Internal Termination

Figure 13-14 shows a line access read with one wait state programmed in CSCR*n* to give the peripheral or memory more time to return read data. This figure follows the same execution as a zero-wait state read burst with the exception of an added wait state.

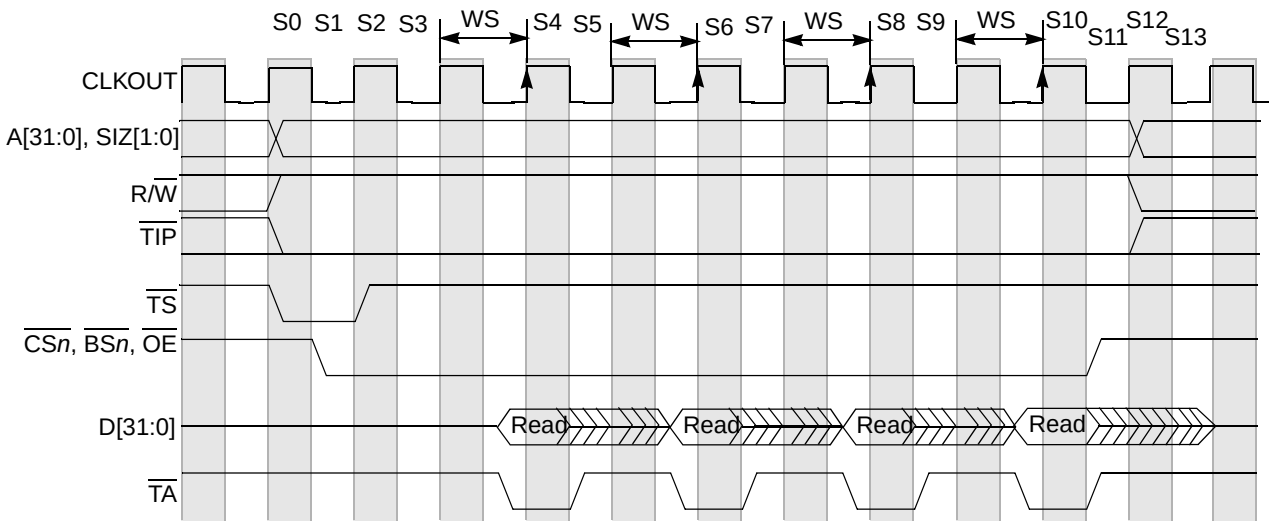


Figure 13-14. Line Read Burst (3-2-2-2), External Termination

Figure 13-15 shows a burst-inhibited line read access with fast termination. The external device executes a basic read cycle while determining that a line is being transferred. The external device uses fast termination for subsequent transfers.

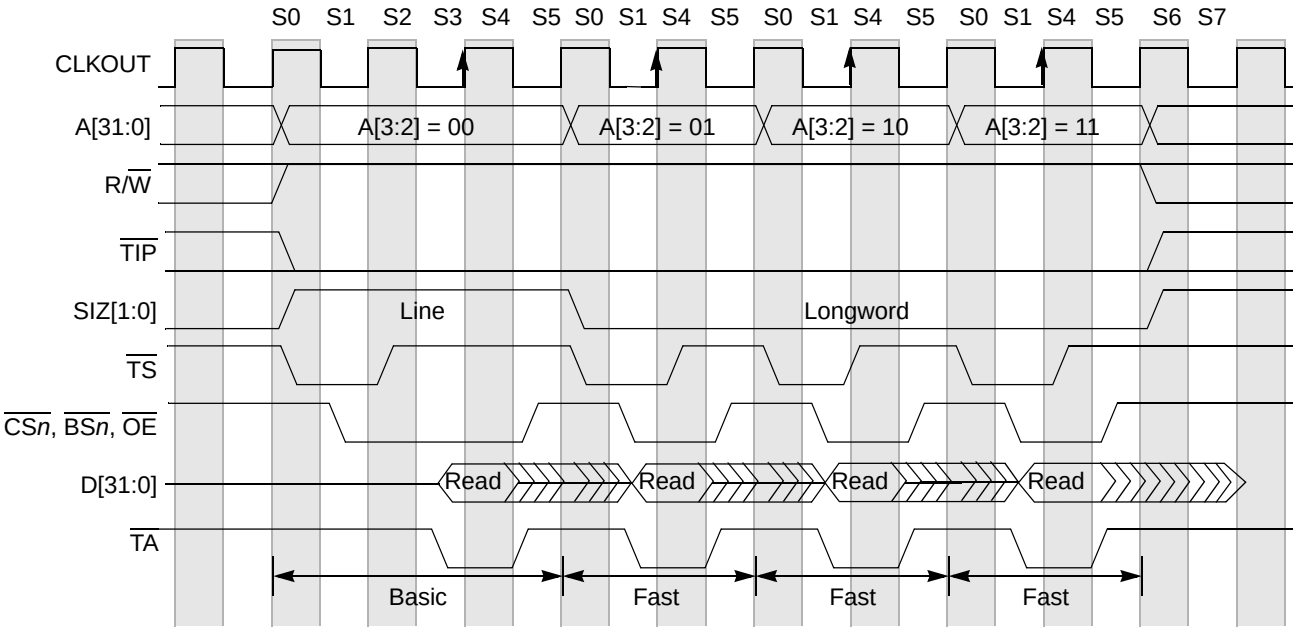


Figure 13-15. Line Read Burst-Inhibited, Fast Termination, External Termination

13.4.7.3 Line Write Bus Cycles

Figure 13-16 shows a line access write with zero wait states. It begins like a basic write bus cycle with data driven one clock after TS. The next pipelined burst data is driven a cycle after the write data is registered (on the rising edge of S6). Each subsequent burst takes a single cycle. Note that as with the line read example in Figure 13-12, CS*n* remain asserted throughout the burst transfer. This example shows the behavior of the address lines for both internal and external termination. Note that when external termination is used, the address lines change with SIZ[1:0].

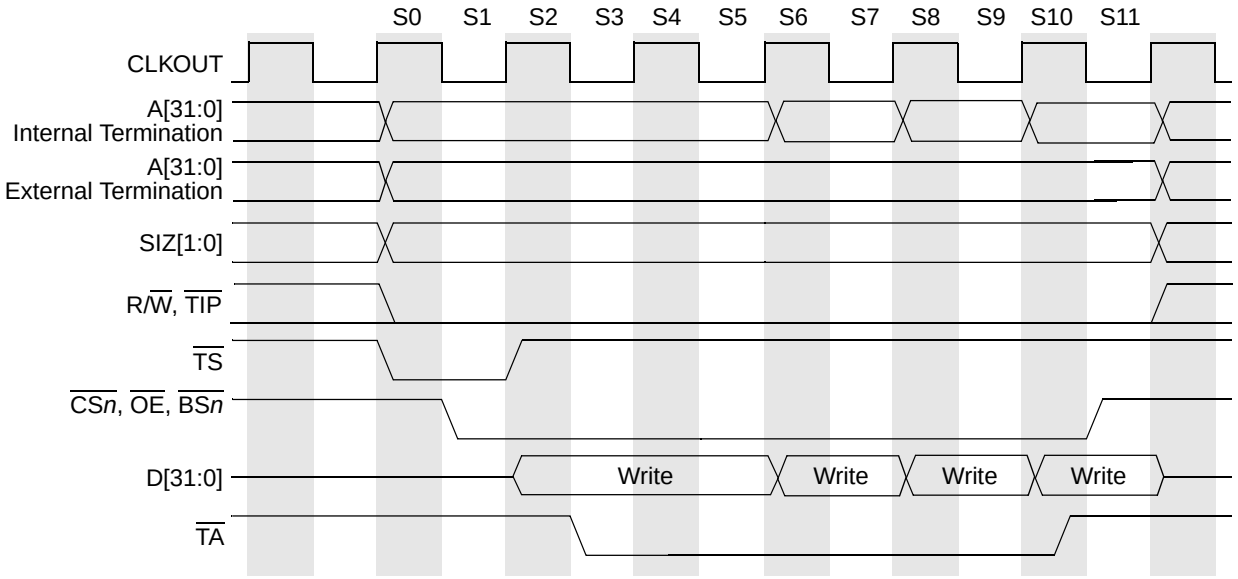


Figure 13-16. Line Write Burst (2-1-1-1), Internal/External Termination

Figure 13-17 shows a line burst write with one wait-state insertion.

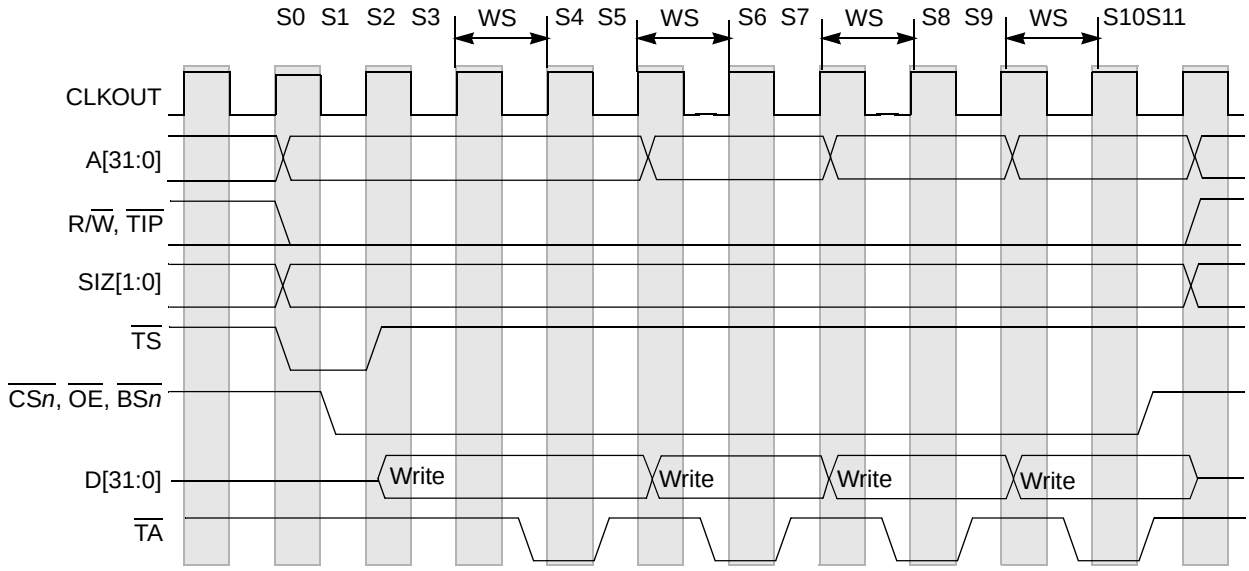


Figure 13-17. Line Write Burst (3-2-2-2) with One Wait State

Figure 13-18 shows a burst-inhibited line write. The external device executes a basic write cycle while determining that a line is being transferred. The external device uses fast termination to end each subsequent transfer.

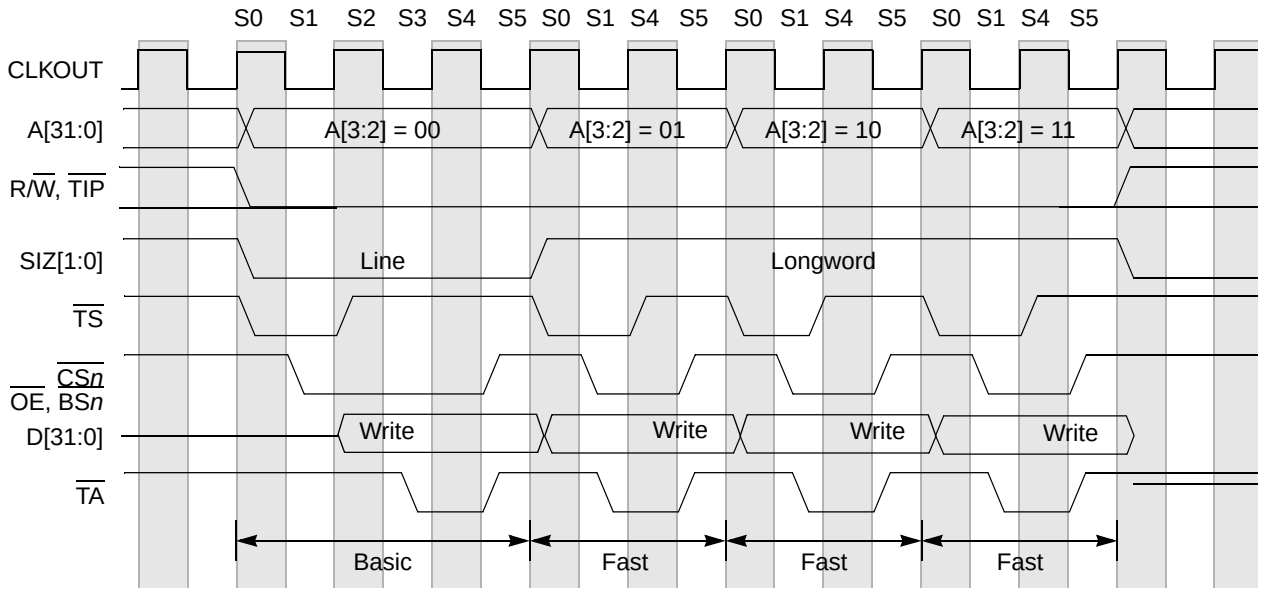


Figure 13-18. Line Write Burst-Inhibited

13.5 Misaligned Operands

Because operands can reside at any byte boundary, unlike opcodes, they are allowed to be misaligned. A byte operand is properly aligned at any address, a word operand is misaligned at an odd address, and a longword is misaligned at an address not a multiple of four. Although the processor enforces no alignment restrictions for data operands (including program counter (PC) relative data addressing), additional bus cycles are required for misaligned operands.

Instruction words and extension words (opcodes) must reside on word boundaries. Attempting to prefetch a misaligned instruction word causes an address error exception.

The processor converts misaligned, cache-inhibited operand accesses to multiple aligned accesses. Figure 13-19 shows the transfer of a longword operand from a byte address to a 32-bit port. In this example, SIZ[1:0] specify a byte transfer and a byte offset of 0x1. The slave device supplies the byte and acknowledges the data transfer. When the processor starts the second cycle, SIZ[1:0] specify a word transfer with a byte offset of 0x2. The next two bytes are transferred in this cycle. In the third cycle, byte 3 is transferred. The byte offset is now 0x0, the port supplies the final byte, and the operation is complete.

	31 — 24	23 — 16	15 — 8	7 — 0	A[2:0]
Transfer 1	—	Byte 0	—	—	001
Transfer 2	—	—	Byte 1	Byte 2	010
Transfer 3	Byte 3	—	—	—	100

Figure 13-19. Example of a Misaligned Longword Transfer (32-Bit Port)

If an operand is cacheable and is misaligned across a cache-line boundary, both lines are loaded into the cache. The example in Figure 13-20 differs from that in Figure 13-19 in that the operand is word-sized and the transfer takes only two bus cycles.

	31 24	23 16	15 8	7 0	A[2:0]
Transfer 1	—	—	—	Byte 0	001
Transfer 2	Byte 1	—	—	—	100

Figure 13-20. Example of a Misaligned Word Transfer (32-Bit Port)

Chapter 14
Signal Descriptions

This chapter describes the processor's external signals. It 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. Chapter 13, "External Interface Module (EIM)," describes how these signals interact.

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.

14.1 Overview

Figure 14-1 shows the block diagram with the signal interface.

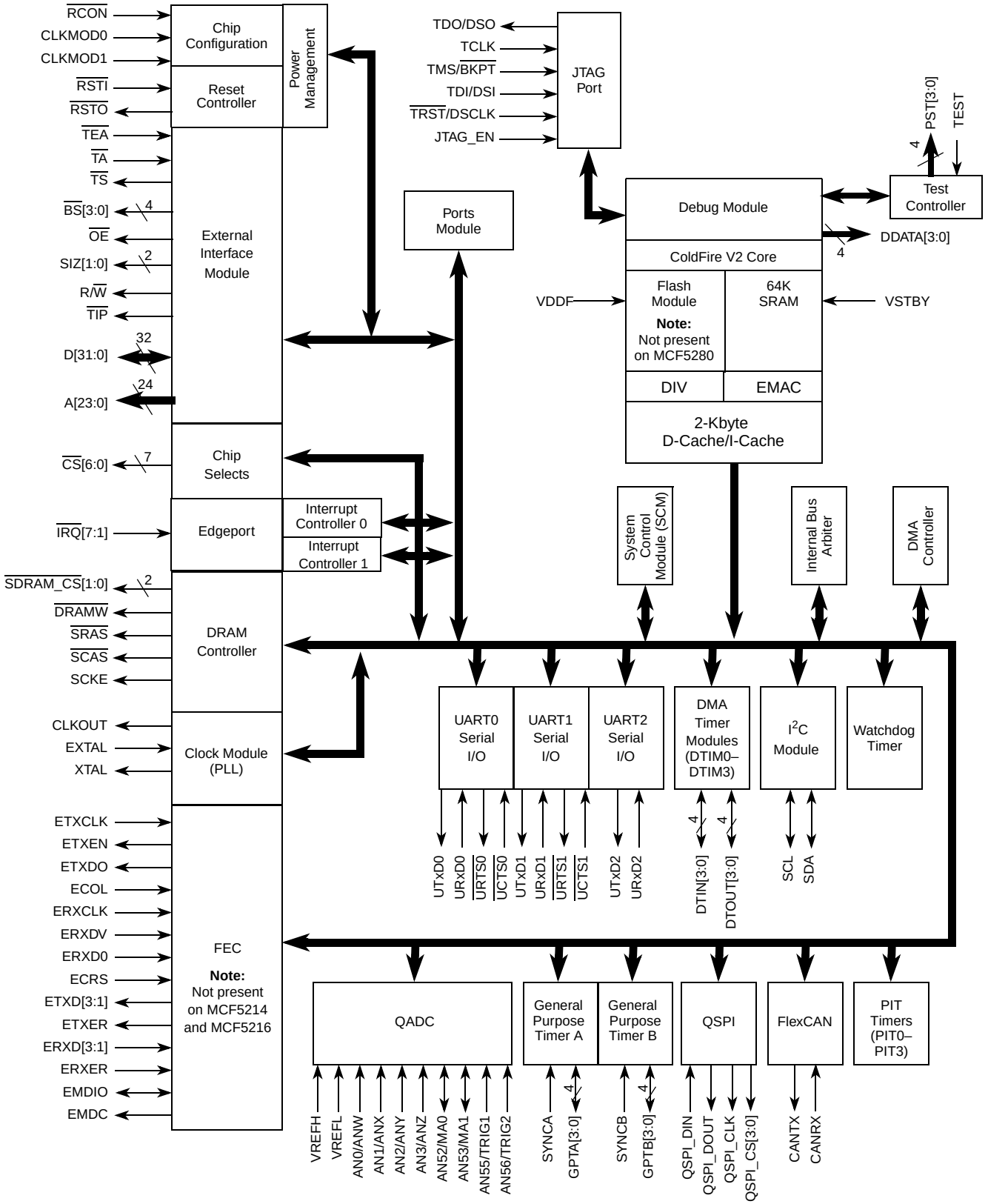


Figure 14-1. MCF5282 Block Diagram with Signal Interfaces

Table 14-1 lists the external signals grouped by functionality.

NOTE

The primary functionality of a pin is not necessarily its default functionality. Pins that are muxed with GPIO will default to their GPIO functionality.

Table 14-1. MCF5282 Signal Description

Signal Name	Abbreviation	Function	I/O	Page
External Memory Interface				
Address	A[23:0]	Define the address of external byte, word, longword, and 16-byte burst accesses.	I/O	14-19
Data	D[31:0]	Data bus. Provide the general purpose data path between the MCU and all other devices.	I/O	14-19
Byte strobes	$\overline{BS}$[3:0]	Define the byte lane of data on the data bus.	I/O	14-19
Output enable	$\overline{OE}$	Indicates when an external device can drive data on the bus.	O	14-19
Transfer acknowledge	$\overline{TA}$	Indicates that the external data transfer is complete and should be asserted for one clock.	I	14-19
Transfer error acknowledge	$\overline{TEA}$	Indicates that an error condition exists for the bus transfer.	I	14-20
Read/Write	R/$\overline{W}$	Indicates the direction of the data transfer on the bus.	I/O	14-20
Transfer size	SIZ[1:0]	Specify the data access size of the current external bus reference.	O	14-20
Transfer start	$\overline{TS}$	Asserted during the first CLKOUT cycle of a transfer when address and attributes are valid.	O	14-20
Transfer in progress	$\overline{TIP}$	Asserted to indicate that a bus transfer is in progress. Negated during idle bus cycles.	O	14-21
Chip selects	$\overline{CS}$[6:0]	Programmed for a base address location and for masking addresses, port size and burst capability indication, wait state generation, and internal/external termination.	O	14-21
SDRAM Controller Signals				
SDRAM row address strobe	$\overline{SRAS}$	SDRAM synchronous row address strobe.	O	14-21
SDRAM column address strobe	$\overline{SCAS}$	SDRAM synchronous column address strobe.	O	14-21

Table 14-1. MCF5282 Signal Description (continued)

Signal Name	Abbreviation	Function	I/O	Page
SDRAM write enable	$\overline{\text{DRAMW}}$	Asserted to signify that a DRAM write cycle is underway. Negated to indicate a read cycle.	O	14-21
SDRAM bank selects	$\overline{\text{SDRAM\_CS}}$[1:0]	Interface to the chip-select lines of the SDRAMs within a memory block.	O	14-21
SDRAM clock enable	SCKE	SDRAM clock enable.	O	14-22
Clock and Reset Signals				
Reset in	$\overline{\text{RSTI}}$	Asserted to enter reset exception processing.	I	14-22
Reset out	$\overline{\text{RSTO}}$	Automatically asserted with $\overline{\text{RSTI}}$. Negation indicates that the PLL has regained its lock.	O	14-22
EXTAL	EXTAL	Driven by an external clock except when used as a connection to the external crystal.	I	14-22
XTAL	XTAL	Internal oscillator connection to the external crystal.	O	14-22
Clock output	CLKOUT	Reflects the system clock.	O	14-22
Chip Configuration Module				
Clock mode	CLKMOD[1:0]	Clock mode select	I	14-22
Reset configuration	RCON	Reset configuration select	I	14-22
External Interrupt Signals				
External interrupts	$\overline{\text{IRQ}}$[7:1]	External interrupt sources.	I	14-23
Ethernet Module Signals (not available on MCF5214 and MCF5216)				
Management data	EMDIO	Transfers control information between the external PHY and the media access controller.	I/O	14-23
Management data clock	EMDC	Provides a timing reference to the PHY for data transfers on the EMDIO signal.	O	14-23
Transmit clock	ETXCLK	Provides a timing reference for ETXEN, ETXD[3:0], and ETXER.	I	14-23
Transmit enable	ETXEN	Indicates when valid nibbles are present on the MII.	O	14-23
Transmit data 0	ETXD0	Serial output Ethernet data.	O	14-23
Collision	ECOL	Asserted to indicate a collision.	I	14-24
Receive clock	ERXCLK	Provides a timing reference for ERXDV, ERXD[3:0], and ERXER.	I	14-24

Table 14-1. MCF5282 Signal Description (continued)

Signal Name	Abbreviation	Function	I/O	Page
Receive data valid	ERXDV	Asserted to indicate that the PHY has valid nibbles present on the MII.	I	14-24
Receive data 0	ERXD0	Ethernet input data transferred from the PHY to the media access controller (when ERXDV is asserted).	I	14-24
Carrier receive sense	ECRS	Asserted to indicate that the transmit or receive medium is not idle.	I	14-24
Transmit data	ETXD[3:1]	Contain the serial output Ethernet data.	O	14-24
Transmit error	ETXER	Asserted (for one or more E_TXCLKs while ETXEN is also asserted) to cause the PHY to send one or more illegal symbols.	O	14-24
Receive data	ERXD[3:1]	Contain the Ethernet input data transferred from the PHY to the media access controller (when ERXDV is asserted in MII mode).	I	14-24
Receive error	ERXER	Indicates (when asserted with ERXDV) that the PHY has detected an error in the current frame.	I	14-25
Queued Serial Peripheral Interface (QSPI) Signals				
QSPI synchronous serial data output	QSPI_DOUT	Provides serial data from the QSPI.	O	14-25
QSPI synchronous serial data input	QSPI_DIN	Provides serial data to the QSPI.	I	14-25
QSPI serial clock	QSPI_CLK	Provides the serial clock from the QSPI.	O	14-25
QSPI chip selects	QSPI_CS[3:0]	Provide QSPI peripheral chip selects.	O	14-25
FlexCAN Signals				
FlexCAN transmit	CANTX	Controller area network transmit data.	O	14-25
FlexCAN transmit	CANRX	Controller area network transmit data.	I	14-25
I^2C Signals				
Serial clock	SCL	Clock signal for the I^2C interface.	I/O	14-26
Serial data	SDA	Data input/output for the I^2C interface.	I/O	14-26
UART Signals				
Transmit serial data output	UTXD[2:0]	Transmitter serial data outputs.	O	14-26
Receive serial data input	URXD[2:0]	Receiver serial data inputs.	I	14-26

Table 14-1. MCF5282 Signal Description (continued)

Signal Name	Abbreviation	Function	I/O	Page
Clear-to-send	$\overline{\text{UCTS}}$[1:0]	Signals UART that it can begin data transmission.	I	14-26
Request to send	$\overline{\text{URTS}}$[1:0]	Automatic UART request to send outputs.	O	14-27
General Purpose Timer Signals				
GPTA	GPTA[3:0]	Provide the external interface to the timer A functions.	I/O	14-27
GPTB	GPTB[3:0]	Provide the external interface to the timer B functions.	I/O	14-27
External clock input	SYNCA/SYNCB	Clear the timer's clock, providing a means of synchronization to externally clocked or timed events.	I	14-27
DMA Timer Signals				
DMA timer input	DTIN[3:0]	Clock the event counter or provide a trigger to timer value capture logic.	I/O	14-27
DMA timer output	DTOUT[3:0]	Pulse or toggle on timer events.	I/O	14-27
Analog-to-Digital Converter (QADC) Signals				
QADC analog input	AN[0:3]/AN[W:Z]	Direct analog input AN*n*, or multiplexed input AN*x*.	I	14-28
QADC analog input	AN[52:53]/MA[0:1]	Direct analog input AN*n*, or multiplexed output MA*n*. MA*n* selects the output of the external multiplexer.	I/O	14-29
QADC analog input	AN[55:56]/ TRIG[1:2]	Direct analog input AN*n*, or input TRIG*n*. TRIG*n* causes one of the two queues to execute.	I	14-29
Debug Support Signals				
JTAG_EN	JTAG_EN	Selects between multiplexed debug module and JTAG signals at reset.	I	14-29
Development serial clock/Test reset	DSCLK/$\overline{\text{TRST}}$	Development serial clock for the serial interface to debug module (DSCLK). Asynchronously resets the internal JTAG controller to the test logic reset state ($\overline{\text{TRST}}$).	I	14-30
Breakpoint/ Test mode select	$\overline{\text{BKPT}}$/TMS	Signals a hardware breakpoint in debug mode ($\overline{\text{BKPT}}$). Provides information that determines JTAG test operation mode (TMS).	I	14-30

Table 14-1. MCF5282 Signal Description (continued)

Signal Name	Abbreviation	Function	I/O	Page
Development serial input/Test data	DSI/TDI	Provides single-bit communication for debug module commands (DSI). Provides serial data port for loading JTAG boundary scan, bypass, and instruction registers (TDI).	I	14-30
Development serial output/Test data	DSO/TDO	Provides single-bit communication for debug module responses (DSO). Provides serial data port for outputting JTAG logic data (TDO).	O	14-30
Test clock	TCLK	JTAG test logic clock.	I	14-30
Debug data	DDATA[3:0]	Display captured processor addresses, data, and breakpoint status.	O	14-31
Processor status outputs	PST[3:0]	Indicate core status.	O	14-31
Test Signals				
Test	TEST	Reserved, should be connected to VSS.	I	14-31
Power and Reference Signals				
QADC analog reference	VRH, VRL	High (VRH) and low (VRL) reference potentials for the analog converter.	Ground	14-32
QADC analog supply	VDDA, VSSA	Isolate the QADC analog circuitry from digital power supply noise.	I	14-32
PLL analog supply	VDDPLL, VSSPLL	Isolate the PLL analog circuitry from digital power supply noise.	I	14-32
QADC positive supply	VDDH	Supplies positive power to the ESD structures in the QADC pads.	I	14-32
Flash erase/program power	VPP	Used for Flash stress testing.	I	14-32
Flash array power and ground	VDDF, VSSF	Supply power and ground to Flash array.	I	14-32
Standby power	VSTBY	Provides standby voltage to RAM array if VDD is lost.	I	14-32
Positive supply	VDD	Supplies positive power to the core logic and I/O pads.	I	14-32
Ground	VSS	Negative supply.		14-32

Table 14-2 lists signals in alphabetical order by abbreviated name.

Table 14-2. MCF5282 Alphabetical Signal Index

Abbreviation	Function	I/O
A[23:0]	Define the address of external byte, word, longword, and 16-byte burst accesses.	I/O
AN[0:3]/AN[W:Z]	Direct analog input AN*n*, or multiplexed input AN*x*.	I
AN[52:53]/MA[0:1]	Direct analog input AN*n*, or multiplexed output MA*n*. MA*n* selects the output of the external multiplexer.	I/O
AN[55:56]/ TRIG[1:2]	Direct analog input AN*n*, or input TRIG*n*. TRIG*n* causes one of the two queues to execute.	I
Breakpoint/ Test mode select	Signals a hardware breakpoint in debug mode ($\overline{\text{BKPT}}$). Provides information that determines JTAG test operation mode (TMS).	I
$\overline{\text{BS}}$[3:0]	Define the byte lane of data on the data bus.	I/O
CANRX	Controller area network transmit data.	I
CANTX	Controller area network transmit data.	O
CLKMOD[1:0]	Clock mode select	I
CLKOUT	Reflects the system clock.	O
$\overline{\text{CS}}$[6:0]	Programmed for a base address location and for masking addresses, port size and burst capability indication, wait state generation, and internal/external termination.	O
D[31:0]	Data bus. Provide the general purpose data path between the MCU and all other devices.	I/O
DDATA[3:0]	Display captured processor addresses, data, and breakpoint status.	O
DSO/TDO	Provides single-bit communication for debug module responses (DSO). Provides serial data port for outputting JTAG logic data (TDO).	O
DSI/TDI	Development serial clock for the serial interface to debug module (DSCLK). Asynchronously resets the internal JTAG controller to the test logic reset state ($\overline{\text{TRST}}$).	I
DSCLK/$\overline{\text{TRST}}$	Provides single-bit communication for debug module commands (DSI). Provides serial data port for loading JTAG boundary scan, bypass, and instruction registers (TDI).	I
$\overline{\text{DRAMW}}$	Asserted to signify that a DRAM write cycle is underway. Negated to indicate a read cycle.	O
DTIN[3:0]	Clock the event counter or provide a trigger to timer value capture logic.	I/O
DTOUT[3:0]	Pulse or toggle on timer events.	I/O
ECOL	Asserted to indicate a collision. **Note:** Not available on MCF5214 and MCF5216	I
ECRS	Asserted to indicate that the transmit or receive medium is not idle. **Note:** Not available on MCF5214 and MCF5216	I

Table 14-2. MCF5282 Alphabetical Signal Index (continued)

Abbreviation	Function	I/O
EMDC	Provides a timing reference to the PHY for data transfers on the EMDIO signal. **Note:** Not available on MCF5214 and MCF5216	O
EMDIO	Transfers control information between the external PHY and the media access controller. **Note:** Not available on MCF5214 and MCF5216	I/O
ERXCLK	Provides a timing reference for ERXDV, ERXD[3:0], and ERXER. **Note:** Not available on MCF5214 and MCF5216	I
ERXD[3:1]	Contain the Ethernet input data transferred from the PHY to the media access controller (when ERXDV is asserted in MII mode). **Note:** Not available on MCF5214 and MCF5216	I
ERXD0	Ethernet input data transferred from the PHY to the media access controller (when ERXDV is asserted). **Note:** Not available on MCF5214 and MCF5216	I
ERXDV	Asserted to indicate that the PHY has valid nibbles present on the MII. **Note:** Not available on MCF5214 and MCF5216	I
ERXER	Indicates (when asserted with ERXDV) that the PHY has detected an error in the current frame. **Note:** Not available on MCF5214 and MCF5216	I
ETXCLK	Provides a timing reference for ETXEN, ETXD[3:0], and ETXER. **Note:** Not available on MCF5214 and MCF5216	I
ETXD[3:1]	Contain the serial output Ethernet data. **Note:** Not available on MCF5214 and MCF5216	O
ETXD0	Serial output Ethernet data. **Note:** Not available on MCF5214 and MCF5216	O
ETXEN	Indicates when valid nibbles are present on the MII. **Note:** Not available on MCF5214 and MCF5216	O
ETXER	Asserted (for one or more E_TXCLKs while ETXEN is also asserted) to cause the PHY to send one or more illegal symbols. **Note:** Not available on MCF5214 and MCF5216	O
EXTAL	Driven by an external clock except when used as a connection to the external crystal.	I
VDDF, VSSF	Supply power and ground to Flash array.	I
VPP	Used for Flash stress testing.	I
GPTA[3:0]	Provide the external interface to the timer A functions.	I/O
GPTB[3:0]	Provide the external interface to the timer B functions.	I/O
Ground	Negative supply.	
$\overline{\text{IRQ}}$[7:1]	External interrupt sources.	I

Table 14-2. MCF5282 Alphabetical Signal Index (continued)

Abbreviation	Function	I/O
JTAG_EN	Selects between multiplexed debug module and JTAG signals at reset.	I
$\overline{OE}$	Indicates when an external device can drive data on the bus.	O
VDDPLL	Isolate the PLL analog circuitry from digital power supply noise.	I
VDD	Supplies positive power to the core logic and I/O pads.	I
PST[3:0]	Indicate core status.	O
VRH, VRL	High (VRH) and low (VRL) reference potentials for the analog converter.	I
VDDA, VSSA	Isolate the QADC analog circuitry from digital power supply noise.	I
QADC analog supply	Supplies positive power to the ESD structures in the QADC pads.	I
QSPI_CLK	Provides the serial clock from the QSPI.	O
QSPI_CS[3:0]	Provide QSPI peripheral chip selects.	O
QSPI_DIN	Provides serial data to the QSPI.	I
QSPI_DOUT	Provides serial data from the QSPI.	O
R/$\overline{W}$	Indicates the direction of the data transfer on the bus.	I/O
$\overline{RCON}$	Reset configuration select.	I
$\overline{RSTI}$	Asserted to enter reset exception processing.	I
$\overline{RSTO}$	Automatically asserted with $\overline{RSTI}$. Negation indicates that the PLL has regained its lock.	O
$\overline{SCAS}$	SDRAM synchronous column address strobe.	O
SCKE	SDRAM clock enable.	O
SCL	Clock signal for the I^2C interface.	I/O
SDA	Data input/output for the I^2C interface.	I/O
$\overline{SDRAM\_CS}$[1:0]	Interface to the chip-select lines of the SDRAMs within a memory block.	O
SIZ[1:0]	Specify the data access size of the current external bus reference.	O
$\overline{SRAS}$	SDRAM synchronous row address strobe.	O
VSTBY	Provides standby voltage to RAM array if VDD is lost.	I
SYNCA/SYNCB	Clear the timer's clock, providing a means of synchronization to externally clocked or timed events.	I
$\overline{TA}$	Indicates that the external data transfer is complete and should be asserted for one CLKOUT cycle.	I
$\overline{TEA}$	Indicates that an error condition exists for the bus transfer.	I

Table 14-2. MCF5282 Alphabetical Signal Index (continued)

Abbreviation	Function	I/O
TEST	Reserved, should be connected to VSS.	I
TCK	JTAG test logic clock.	I
$\overline{\text{TIP}}$	Asserted to indicate that a bus transfer is in progress. Negated during idle bus cycles.	O
$\overline{\text{TS}}$	Asserted during the first CLKOUT cycle of a transfer when address and attributes are valid.	O
$\overline{\text{UCTS}}$[1:0]	Signals UART that it can begin data transmission.	I
$\overline{\text{URTS}}$[1:0]	Automatic UART request to send outputs.	O
URXD[2:0]	Receiver serial data inputs.	I
UTXD[2:0]	Transmitter serial data outputs.	O
XTAL	Internal oscillator connection to the external crystal.	O

Table 14-3. MCF5282 Signals and Pin Numbers Sorted by Function

MAPBGA Pin	Pin Functions			Description	Primary I/O	Internal Pull-up[1]
	Primary[2]	Secondary	Tertiary			
Reset						
R11	$\overline{\text{RSTI}}$	—	—	Reset in	I	Yes
P11	$\overline{\text{RSTO}}$	—	—	Reset out	O	—
Clock						
T8	EXTAL	—	—	External clock/crystal in	I	—
R8	XTAL	—	—	Crystal drive	O	—
N7	CLKOUT	—	—	Clock out	O	—
Chip Configuration/Mode Selection						
R14	CLKMOD0	—	—	Clock mode select	I	Yes
T14	CLKMOD1	—	—	Clock mode select	I	Yes
T11	$\overline{\text{RCON}}$	—	—	Reset configuration enable	I	Yes
H1	D26	PA2	—	Chip mode	I/O	—
K2	D17	PB1	—	Chip mode	I/O	—
K3	D16	PB0	—	Chip mode	I/O	—
J4	D19	PB3	—	Boot device/data port size	I/O	—
K1	D18	PB2	—	Boot device/data port size	I/O	—
J2	D21	PB5	—	Output pad drive strength	I/O	—

Table 14-3. MCF5282 Signals and Pin Numbers Sorted by Function (continued)

MAPBGA Pin	Pin Functions			Description	Primary I/O	Internal Pull-up[1]
	Primary[2]	Secondary	Tertiary			
External Memory Interface and Ports						
C6:B6:A5	A[23:21]	PF[7:5]	$\overline{CS}$[6:4]	Address bus	O	Yes
C4:B4:A4:B3:A3	A[20:16]	PF[4:0]	—	Address bus	O	Yes
A2:B1:B2:C1: C2:C3:D1:D2	A[15:8]	PG[7:0]	—	Address bus	O	Yes
D3:D4:E1:E2: E3:E4:F1:F2	A[7:0]	PH[7:0]	—	Address bus	O	Yes
F3:G1:G2:G3: G4:H1:H2:H3	D[31:24]	PA[7:0]	—	Data bus	I/O	—
H4:J1:J2:J3: J4:K1:K2:K3	D[23:16]	PB[7:0]	—	Data bus	I/O	—
L1:L2:L3:L4: M1:M2:M3:M4	D[15:8]	PC[7:0]	—	Data bus	I/O	—
N1:N2:N3:P1: N5:T6:R6:P6	D[7:0]	PD[7:0]	—	Data bus	I/O	—
P14:T15:R15:R16	$\overline{BS}$[3:0]	PJ[7:4]	—	Byte strobe	I/O	Yes
N16	$\overline{OE}$	PE7	—	Output enable	I/O	—
P16	$\overline{TA}$	PE6	—	Transfer acknowledge	I/O	Yes
P15	$\overline{TEA}$	PE5	—	Transfer error acknowledge	I/O	Yes
N15	R/$\overline{W}$	PE4	—	Read/write	I/O	Yes
N14	SIZ1	PE3	SYNCA	Transfer size	I/O	Yes[3]
M16	SIZ0	PE2	SYNCB	Transfer size	I/O	Yes[4]
M15	$\overline{TS}$	PE1	SYNCA	Transfer start	I/O	Yes
M14	$\overline{TIP}$	PE0	SYNCB	Transfer in progress	I/O	Yes
Chip Selects						
L16:L15:L14:L13	$\overline{CS}$[3:0]	PJ[3:0]	—	Chip selects 3-0	I/O	Yes
C6:B6:A5	A[23:21]	PF[7:5]	$\overline{CS}$[6:4]	Chip selects 6-4	O	Yes
SDRAM Controller						
H15	$\overline{SRAS}$	PSD5	—	SDRAM row address strobe	I/O	—
H16	$\overline{SCAS}$	PSD4	—	SDRAM column address strobe	I/O	—
G15	$\overline{DRAMW}$	PSD3	—	SDRAM write enable	I/O	—
H13:G16	$\overline{SDRAM\_CS}$[1:0]	PSD[2:1]	—	SDRAM chip selects	I/O	—
H14	SCKE	PSD0	—	SDRAM clock enable	I/O	—

Table 14-3. MCF5282 Signals and Pin Numbers Sorted by Function (continued)

MAPBGA Pin	Pin Functions			Description	Primary I/O	Internal Pull-up[1]
	Primary[2]	Secondary	Tertiary			
External Interrupts Port						
B15:B16:C14:C15: C16: D14:D15	$\overline{\text{IRQ}}$[7:1]	PNQ[7:1]	—	External interrupt request	I/O	—
Ethernet For MCF5280, MCF5281 and MCF5282 only						
C10	EMDIO	PAS5	URXD2	Management channel serial data	I/O	—
B10	EMDC	PAS4	UTXD2	Management channel clock	I/O	—
A8	ETXCLK	PEH7	—	MAC Transmit clock	I/O	—
D6	ETXEN	PEH6	—	MAC Transmit enable	I/O	—
D7	ETXD0	PEH5	—	MAC Transmit data	I/O	—
B11	ECOL	PEH4	—	MAC Collision	I/O	—
A10	ERXCLK	PEH3	—	MAC Receive clock	I/O	—
C8	ERXDV	PEH2	—	MAC Receive enable	I/O	—
D9	ERXD0	PEH1	—	MAC Receive data	I/O	—
A11	ECRS	PEH0	—	MAC Carrier sense	I/O	—
A7:B7:C7	ETXD[3:1]	PEL[7:5]	—	MAC Transmit data	I/O	—
D10	ETXER	PEL4	—	MAC Transmit error	I/O	—
A9:B9:C9	ERXD[3:1]	PEL[3:1]	—	MAC Receive data	I/O	—
B8	ERXER	PEL0	—	MAC Receive error	I/O	—
GPIO For MCF5214 and MCF5216 only						
C10	PAS5	URXD2	—	GPIO/serial data	I/O	—
B10	NC	—	—	No connect	—	—
A8	PEL0	—	—	GPIO	I/O	—
D6	NC	—	—	No connect	—	—
D7	NC	—	—	No connect	—	—
B11	NC	—	—	No connect	—	—
A10	PAS4	UTXD2	—	GPIO/serial data	I/O	—
C8	NC	—	—	No connect	—	—
D9	NC	—	—	No connect	—	—
A11	NC	—	—	No connect	—	—

Table 14-3. MCF5282 Signals and Pin Numbers Sorted by Function (continued)

MAPBGA Pin	Pin Functions			Description	Primary I/O	Internal Pull-up[1]
	Primary[2]	Secondary	Tertiary			
A7:B7:C7	PEL[7:5]	—	—	GPIO	I/O	—
D10	PEL4	—	—	GPIO	I/O	—
A9:B9:C9	PEL[3:1]	—	—	GPIO	I/O	—
B8	NC	—	—	No connect	—	—
FlexCAN						
D16	CANRX	PAS3	URXD2	FlexCAN Receive data	I/O	—
E13	CANTX	PAS2	UTXD2	FlexCAN Transmit data	I/O	—
I^2C						
E14	SDA	PAS1	URXD2	I^2C Serial data	I/O	Yes[5]
E15	SCL	PAS0	UTXD2	I^2C Serial clock	I/O	Yes[6]
QSPI						
F13	QSPI_DOUT	PQS0	—	QSPI data out	I/O	—
E16	QSPI_DIN	PQS1	—	QSPI data in	I/O	—
F14	QSPI_CLK	PQS2	—	QSPI clock	I/O	—
G14:G13:F16:F15	QSPI_CS[3:0]	PQS[6:3]	—	QSPI chip select	I/O	—
UARTs						
R7	URXD1	PUA3	—	U1 receive data	I/O	—
P7	UTXD1	PUA2	—	U1 transmit data	I/O	—
N6	URXD0	PUA1	—	U0 receive data	I/O	—
T7	UTXD0	PUA0	—	U0 transmit data	I/O	—
D16	CANRX	PAS3	URXD2	U2 receive data	I/O	—
E13	CANTX	PAS2	UTXD2	U2 transmit data	I/O	—
E14	SDA	PAS1	URXD2	U2 receive data	I/O	Yes[5]
E15	SCL	PAS0	UTXD2	U2 transmit data	I/O	Yes[6]
K16	DTIN3	PTC3	$\overline{\text{URTS1}}$/ $\overline{\text{URTS0}}$	U1/U0 Request to Send	I/O	—
K15	DTOUT3	PTC2	$\overline{\text{URTS1}}$/ $\overline{\text{URTS0}}$	U1/U0 Request to Send	I/O	—
K14	DTIN2	PTC1	$\overline{\text{UCTS1}}$/ $\overline{\text{UCTS0}}$	U1/U0 Clear to Send	I/O	—
K13	DTOUT2	PTC0	$\overline{\text{UCTS1}}$/ $\overline{\text{UCTS0}}$	U1/U0 Clear to Send	I/O	—

Table 14-3. MCF5282 Signals and Pin Numbers Sorted by Function (continued)

MAPBGA Pin	Pin Functions			Description	Primary I/O	Internal Pull-up[1]
	Primary[2]	Secondary	Tertiary			
J16	DTIN1	PTD3	$\overline{\text{URTS1}}$/$\overline{\text{URTS0}}$	U1/U0 Request to Send	I/O	—
J15	DTOUT1	PTD2	$\overline{\text{URTS1}}$/$\overline{\text{URTS0}}$	U1/U0 Request to Send	I/O	—
J14	DTIN0	PTD1	$\overline{\text{UCTS1}}$/$\overline{\text{UCTS0}}$	U1/U0 Clear to Send	I/O	—
J13	DTOUT0	PTD0	$\overline{\text{UCTS1}}$/$\overline{\text{UCTS0}}$	U1/U0 Clear to Send	I/O	—
Note: The below two pins are for the MCF5280, MCF5281, and MCF5282 only.						
C10	EMDIO	PAS5	URXD2	U2 receive data	I/O	—
B10	EMDC	PAS4	UTXD2	U2 transmit data	I/O	—
Note: The below two pins are for the MCF5214 and MCF5216 only.						
C10	PAS5	URXD2	—	U2 receive data	I/O	—
B10	NC	—	—	No connect	I/O	—
General Purpose Timers						
T13:R13:P13:N13	GPTA[3:0]	PTA[3:0]	—	Timer A IC/OC/PAI	I/O	Yes
T12:R12:P12:N12	GPTB[3:0]	PTB[3:0]	—	Timer B IC/OC/PAI	I/O	Yes
N14	SIZ1	PE3	SYNCA	Timer A synchronization input	I/O	Yes[3]
M16	SIZ0	PE2	SYNCB	Timer B synchronization input	I/O	Yes[4]
M15	$\overline{\text{TS}}$	PE1	SYNCA	Timer A synchronization input	I/O	Yes
M14	$\overline{\text{TIP}}$	PE0	SYNCB	Timer B synchronization input	I/O	Yes
DMA Timers						
K16	DTIN3	PTC3	$\overline{\text{URTS1}}$/$\overline{\text{URTS0}}$	Timer 3 in	I/O	—
K15	DTOUT3	PTC2	$\overline{\text{URTS1}}$/$\overline{\text{URTS0}}$	Timer 3 out	I/O	—
K14	DTIN2	PTC1	$\overline{\text{UCTS1}}$/$\overline{\text{UCTS0}}$	Timer 2 in	I/O	—
K13	DTOUT2	PTC0	$\overline{\text{UCTS1}}$/$\overline{\text{UCTS0}}$	Timer 2 out	I/O	—
J16	DTIN1	PTD3	$\overline{\text{URTS1}}$/$\overline{\text{URTS0}}$	Timer 1 in	I/O	—
J15	DTOUT1	PTD2	$\overline{\text{URTS1}}$/$\overline{\text{URTS0}}$	Timer 1 out	I/O	—

Table 14-3. MCF5282 Signals and Pin Numbers Sorted by Function (continued)

MAPBGA Pin	Pin Functions			Description	Primary I/O	Internal Pull-up[1]
	Primary[2]	Secondary	Tertiary			
J14	DTIN0	PTD1	$\overline{\text{UCTS1}}$/ $\overline{\text{UCTS0}}$	Timer 0 in	I/O	—
J13	DTOUT0	PTD0	$\overline{\text{UCTS1}}$/ $\overline{\text{UCTS0}}$	Timer 0 out	I/O	—
Queued Analog-to-Digital Converter (QADC)						
T3	AN0	PQB0	ANW	Analog channel 0	I/O	—
R2	AN1	PQB1	ANX	Analog channel 1	I/O	—
T2	AN2	PQB2	ANY	Analog channel 2	I/O	—
R1	AN3	PQB3	ANZ	Analog channel 3	I/O	—
R4	AN52	PQA0	MA0	Analog channel 52	I/O	—
T4	AN53	PQA1	MA1	Analog channel 53	I/O	—
P3	AN55	PQA3	ETRIG1	Analog channel 55	I/O	—
R3	AN56	PQA4	ETRIG2	Analog channel 56	I/O	—
P4	VRH	—	—	High analog reference	I	—
T5	VRL	—	—	Low analog reference	I	—
Debug and JTAG Test Port Control						
R9	JTAG_EN	—	—	JTAG Enable	I	—
P9	DSCLK	$\overline{\text{TRST}}$	—	Debug clock / TAP reset	I	Yes[7]
T9	TCLK	—	—	TAP clock	I	Yes[7]
P10	$\overline{\text{BKPT}}$	TMS	—	Breakpoint/TAP test mode select	I	Yes[7]
R10	DSI	TDI	—	Debug data in / TAP data in	I	Yes[7]
T10	DSO	TDO	—	Debug data out / TAP data out	O	—
C12:D12:A13:B13	DDATA[3:0]	PDD[7:4]	—	Debug data	I/O	—
C13:A14:B14:A15	PST[3:0]	PDD[3:0]	—	Processor status data	I/O	—
Test						
N10	TEST	—		Test mode pin	I	—
Power Supplies						
R5	VDDA	—	—	Analog positive supply	I	—
P5:T1	VSSA	—	—	Analog ground	I	—
P2	VDDH	—	—	ESD positive supply	I	—
N8	VDDPLL	—	—	PLL positive supply	I	—

Table 14-3. MCF5282 Signals and Pin Numbers Sorted by Function (continued)

MAPBGA Pin	Pin Functions			Description	Primary I/O	Internal Pull-up[1]
	Primary[2]	Secondary	Tertiary			
P8	VSSPLL	—	—	PLL ground	I	—
A6:C11	VPP	—	—	Flash (stress) programming voltage	I	—
A12:C5:D5:D11	VDDF	—	—	Flash positive supply	I	—
B5:B12:	VSSF	—	—	Flash module ground	I	—
N11	VSTBY	—	—	Standby power	I	—
E6-E11:F5:F7-F10:F12:G5:G6:G11:G12:H5:H6:H11:H12:J5:J6:J11:J12:K5:K6:K11:K12:L5:L7-L10:L12:M6-M11	VDD	—	—	Positive supply	I	—
A1:A16:E5:E12:F6:F11:G7-G10:H7-H10:J7-J10:K7-K10:L6:L11:M5:M12:T16	VSS	—	—	Ground	I	—

1 Pull-ups are not active when GPIO functions are selected for the pins.

2 The primary functionality of a pin is not necessarily its default functionality. Pins that have GPIO functionality will default to GPIO inputs.

3 Pull-up is active only with the SYNCA function.

4 Pull-up is active only with the SYNCB function.

5 Pull-up is active only with the SDA function.

6 Pull-up is active only with SCL function.

7 Pull-up is active when JTAG_EN is driven high.

14.1.1 Single-Chip Mode

In single-chip mode, signals default to GPIO inputs after a system reset. Table 14-4 is a listing of signals that do not default to a GPIO function.

Table 14-4. Pin Reset States at Reset (Single-Chip Mode)

Signal	Reset	I/O
Clock and Reset Signals		
$\overline{\text{RSTI}}$	—	I
$\overline{\text{RSTO}}$	Low	O
EXTAL	—	I
XTAL	XTAL	O
CLKOUT	CLKOUT	O

Table 14-4. Pin Reset States at Reset (Single-Chip Mode) (continued)

Signal	Reset	I/O
Debug Support Signals		
JTAG_EN	—	I
DSCLK/$\overline{\text{TRST}}$	—	I
$\overline{\text{BKPT}}$/TMS	—	I
DSI/TDI	—	I
DSO/TDO	High	O
TCLK	—	I
DDATA[3:0]	DDATA{3:0]	O
PST[3:0]	PST[3:0]	O

14.1.2 External Boot Mode

When booting from external memory, the address bus, data bus, and bus control signals will default to their bus functionalities as shown in Table 14-5. As in single-chip mode, the signals listed in Table 14-4 will operate as described above. All other signals will default to GPIO inputs.

Table 14-5. Default Signal Functions After System Reset (External Boot Mode)

Signal	Reset	I/O
A[23:0]	A[23:0]	O
D[31:0]	—	I/O
$\overline{\text{BS}}$[3:0]	High	O
$\overline{\text{OE}}$	High	O
$\overline{\text{TA}}$	—	I
$\overline{\text{TEA}}$	—	I
R/$\overline{\text{W}}$	High	O
SIZ[1:0]	High	O
$\overline{\text{TS}}$	High	O
$\overline{\text{TIP}}$	High	O
$\overline{\text{CS}}$[6:0]	High	O

14.2 External Signals

The following sections describe the external signals on the device.

14.2.1 External Interface Module (EIM) Signals

These signals are used for doing transactions on the external bus.

14.2.1.1 Address Bus (A[23:0])

The 24 dedicated address signals, A[23:0], define the address of external byte, word, longword and 16-byte burst accesses. These three-state outputs are the 24 lsbs of the internal 32-bit address bus. The address lines also serve as the SDRAM addressing, providing multiplexed row and column address signals.

These pins are configured for GPIO ports F, G and H in single-chip mode. The A[23:21] pins can also be configured for $\overline{CS}$[6:4].

14.2.1.2 Data Bus (D[31:0])

These three-state bidirectional signals provide the general purpose data path between the MCU and all other devices. Data is sampled by the processor on the rising CLKOUT edge. The data bus port width and wait states are initially defined for the external boot chip select, $\overline{CS0}$, by D[19:18] during chip configuration at reset. The port width for each chip select and SDRAM bank is programmable. The data bus uses a default configuration if none of the chip selects or SDRAM bank match the address decode. The default configuration is a 32-bit port with external termination and burst-inhibited transfers. The data bus can transfer byte, word, or longword data widths. All 32 data bus signals are driven during writes, regardless of port width and operand size.

D[26:24, 21, 19:16] are used during chip configuration as inputs to configure the functions as described in Chapter 27, “Chip Configuration Module (CCM).”

These pins are configured as GPIO ports A, B, C and D in single-chip mode.

14.2.1.3 Byte Strobes ($\overline{BS}$[3:0])

The byte strobes ($\overline{BS}$[3:0]) define the byte lane of data on the data bus. During accesses, these outputs act as the byte select signals that indicate valid data is to be latched or driven onto a byte lane when driven low. For SRAM or Flash devices, the $\overline{BS}$[3:0] outputs should be connected to individual byte strobe signals. For SDRAM devices, the $\overline{BS}$[3:0] should be connected to individual SDRAM DQM signals. Note that most SDRAMs associate DQM3 with the MSB, in which case $\overline{BS3}$ is connected to the SDRAM's DQM3 input.

These pins can also be configured as GPIO PJ[7:4].

14.2.1.4 Output Enable ($\overline{OE}$)

This output signal indicates when an external device can drive data during external read cycles.

This pin can also be configured as GPIO PE7.

14.2.1.5 Transfer Acknowledge ($\overline{TA}$)

This signal indicates that the external data transfer is complete. During a read cycle, when the processor recognizes $\overline{TA}$, it latches the data and then terminates the bus cycle. During a write cycle, when the processor recognizes $\overline{TA}$, the bus cycle is terminated. If all bus cycles support fast termination, $\overline{TA}$ can be tied low.

This pin can also be configured as GPIO PE6.

14.2.1.6 Transfer Error Acknowledge ($\overline{\text{TEA}}$)

This signal indicates an error condition exists for the bus transfer. The bus cycle is terminated and the CPU begins execution of the access error exception. This signal is an input in master mode.

This pin can also be configured as GPIO PE5.

14.2.1.7 Read/Write (R/$\overline{\text{W}}$)

This output signal indicates the direction of the data transfer on the bus. A logic 1 indicates a read from a slave device and a logic 0 indicates a write to a slave device.

This pin can also be configured as GPIO PE4.

14.2.1.8 Transfer Size(SIZ[1:0])

When the device is in normal mode, static bus sizing lets the programmer change data bus width between 8, 16, and 32 bits for each chip select. The SIZ[1:0] outputs specify the data access size of the current external bus reference as shown in Table 14-6.

Table 14-6. Transfer Size Encoding

SIZ[1:0]	Transfer Size
00	Longword
01	Byte
10	Word
11	16-byte line

Note that for misaligned transfers, SIZ[1:0] indicate the size of each transfer. For example, if a longword access occurs at a misaligned offset of 0x1, a byte is transferred first (SIZ[1:0] = 01), a word is next transferred at offset 0x2 (SIZ[1:0] = 10), then the final byte is transferred at offset 0x4 (SIZ[1:0] = 01).

For aligned transfers larger than the port size, SIZ[1:0] behaves as follows:

- If bursting is used, SIZ[1:0] stays at the size of transfer.
- If bursting is inhibited, SIZ[1:0] first shows the size of the transfer and then shows the port size.

For burst-inhibited transfers, SIZ[1:0] changes with each $\overline{\text{TS}}$ assertion to reflect the next transfer size. For transfers to port sizes smaller than the transfer size, SIZ[1:0] indicates the size of the entire transfer on the first access and the size of the current port transfer on subsequent transfers. For example, for a longword write to an 8-bit port, SIZ[1:0] = 00 for the first byte transfer and 01 for the next three.

These pins can also be configured as GPIO PE[3:2] or SYNCA, SYNCB.

14.2.1.9 Transfer Start ($\overline{\text{TS}}$)

The device asserts $\overline{\text{TS}}$ during the first CLKOUT cycle of a transfer when address and attributes ($\overline{\text{TIP}}$, R/$\overline{\text{W}}$, and SIZ[1:0]) are valid. $\overline{\text{TS}}$ is negated in the following CLKOUT cycle.

This pin can also be configured as GPIO PE1 or SYNCA.

14.2.1.10 Transfer In Progress (TIP)

The TIP output is asserted indicating a bus transfer is in progress. It is negated during idle bus cycles. Note that TIP is held asserted on back-to-back cycles.

NOTE

TIP is not asserted during SDRAM accesses.

This pin can also be configured as GPIO PE0 or SYNCB.

14.2.1.11 Chip Selects (CS[6:0])

Each chip select can be programmed for a base address location and for masking addresses, port size and burst-capability indication, wait-state generation, and internal/external termination.

Reset clears all chip select programming; CS0 is the only chip select initialized out of reset. CS0 is also unique because it can function at reset as a global chip select that allows boot ROM to be selected at any defined address space. The port size for boot CS0 is set during chip configuration by the levels on D[19:18] on the rising edge of RSTI, as described in Chapter 27, "Chip Configuration Module (CCM)." The chip-select implementation is described in Chapter 12, "Chip Select Module."

These pins can also be configured as A[23:21] and GPIO PJ[3:0].

14.2.2 SDRAM Controller Signals

These signals are used for SDRAM accesses.

14.2.2.1 SDRAM Row Address Strobe (SRAS)

This output is the SDRAM synchronous row address strobe.

This pin is configured as GPIO PSD5 in single-chip mode.

14.2.2.2 SDRAM Column Address Strobe (SCAS)

This output is the SDRAM synchronous column address strobe.

This pin is configured as GPIO PSD4 in single-chip mode.

14.2.2.3 SDRAM Write Enable (DRAMW)

The DRAM write signal (DRAMW) is asserted to signify that a DRAM write cycle is underway. A read cycle is indicated by the negation of DRAMW.

This pin is configured as GPIO PSD3 in single-chip mode.

14.2.2.4 SDRAM Bank Selects (SDRAM_CS[1:0])

These signals interface to the chip-select lines of the SDRAMs within a memory block. Thus, there is one SDRAM_CS line for each memory block (the processor supports two SDRAM memory blocks).

These pins is configured as GPIO PSD[2:1] in single-chip mode.

14.2.2.5 SDRAM Clock Enable (SCKE)

This output is the SDRAM clock enable.

This pin is configured as GPIO PSD0 in single-chip mode.

14.2.3 Clock and Reset Signals

The clock and reset signals configure the device and provide interface signals to the external system.

14.2.3.1 Reset In ($\overline{\text{RSTI}}$)

Asserting $\overline{\text{RSTI}}$ causes the device to enter reset exception processing. When $\overline{\text{RSTI}}$ is recognized the address bus, data bus, SIZ, R/$\overline{\text{W}}$, $\overline{\text{AS}}$, and $\overline{\text{TS}}$ are three-stated. $\overline{\text{RSTO}}$ is asserted automatically when $\overline{\text{RSTI}}$ is asserted.

14.2.3.2 Reset Out ($\overline{\text{RSTO}}$)

After $\overline{\text{RSTI}}$ is asserted, the PLL temporarily loses its lock, during which time $\overline{\text{RSTO}}$ is asserted. When the PLL regains its lock, $\overline{\text{RSTO}}$ negates again. This signal can be used to reset external devices.

14.2.3.3 EXTAL

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

14.2.3.4 XTAL

This output is an internal oscillator connection to the external crystal.

14.2.3.5 Clock Output (CLKOUT)

The internal PLL generates CLKOUT. This output reflects the internal system clock.

14.2.4 Chip Configuration Signals

14.2.4.1 $\overline{\text{RCON}}$

If the external $\overline{\text{RCON}}$ signal is asserted, 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 27.6, “Functional Description”). The internal configuration signals are driven to reflect the levels on the external configuration pins to allow for module configuration.

14.2.4.2 CLKMOD[1:0]

The state of the CLKMOD[1:0] pins during reset determines the clock mode after reset.

14.2.5 External Interrupt Signals

14.2.5.1 External Interrupts ($\overline{\text{IRQ}}$[7:1])

These inputs are the external interrupt sources. See Chapter 11, “Edge Port Module (EPORT)” for more information on these interrupt sources and their corresponding registers.

These pins are configured as GPIO PNQ[7:1] in single-chip mode.

14.2.6 Ethernet Module Signals

The following signals are used by the Ethernet module for data and clock signals.

NOTE

These signals are not available on the MCF5214 and MCF5216.

14.2.6.1 Management Data (EMDIO)

The bidirectional EMDIO signal transfers control information between the external PHY and the media-access controller. Data is synchronous to EMDC and applies to MII mode operation. This signal is an input after reset. When the FEC is operated in 10 Mbps 7-wire interface mode, this signal should be connected to VSS.

This pin can also be configured as GPIO PAS5 or URXD2.

14.2.6.2 Management Data Clock (EMDC)

EMDC is an output clock which provides a timing reference to the PHY for data transfers on the EMDIO signal and applies to MII mode operation.

This pin can also be configured as GPIO PAS4 or UTXD2.

14.2.6.3 Transmit Clock (ETXCLK)

This is an input clock which provides a timing reference for ETXEN, ETXD[3:0] and ETXER.

This pin can also be configured as GPIO PEH7.

14.2.6.4 Transmit Enable (ETXEN)

The transmit enable (ETXEN) output indicates when valid nibbles are present on the MII. This signal is asserted with the first nibble of a preamble and is negated before the first ETXCLK following the final nibble of the frame.

This pin can also be configured as GPIO PEH6.

14.2.6.5 Transmit Data 0 (ETXD0)

ETXD0 is the serial output Ethernet data and is only valid during the assertion of ETXEN. This signal is used for 10 Mbps Ethernet data. This signal is also used for MII mode data in conjunction with ETXD[3:1].

This pin can also be configured as GPIO PEH5.

14.2.6.6 Collision (ECOL)

The ECOL input is asserted upon detection of a collision and remains asserted while the collision persists. This signal is not defined for full-duplex mode.

This pin can also be configured as GPIO PEH4.

14.2.6.7 Receive Clock (ERXCLK)

The receive clock (ERXCLK) input provides a timing reference for ERXDV, ERXD[3:0], and ERXER.

This pin can also be configured as GPIO PEH3.

14.2.6.8 Receive Data Valid (ERXDV)

Asserting the receive data valid (ERXDV) input indicates that the PHY has valid nibbles present on the MII. ERXDV should remain asserted from the first recovered nibble of the frame through to the last nibble. Assertion of ERXDV must start no later than the SFD and exclude any EOF.

This pin can also be configured as GPIO PEH2.

14.2.6.9 Receive Data 0 (ERXD0)

ERXD0 is the Ethernet input data transferred from the PHY to the media-access controller when ErXDV is asserted. This signal is used for 10 Mbps Ethernet data. This signal is also used for MII mode Ethernet data in conjunction with ERXD[3:1]. This pin can also be configured as GPIO PEH1.

14.2.6.10 Carrier Receive Sense (ECRS)

ECRS is an input signal which, when asserted, signals that transmit or receive medium is not idle, and applies to MII mode operation.

This pin can also be configured as GPIO PEH0.

14.2.6.11 Transmit Data 1–3 (ETXD[3:1])

These pins contain the serial output Ethernet data and are valid only during assertion of ETXEN in MII mode.

These pins can also be configured as GPIO PEL[7:5].

14.2.6.12 Transmit Error (ETXER)

When the ETXER output is asserted for one or more E_TXCLKs while ETXEN is also asserted, the PHY sends one or more illegal symbols. ETXER has no effect at 10 Mbps or when ETXEN is negated, and applies to MII mode operation.

These pins can also be configured as GPIO PEL4.

14.2.6.13 Receive Data 1–3 (ERXD[3:1])

These pins contain the Ethernet input data transferred from the PHY to the media-access controller when ERXDV is asserted in MII mode operation.

These pins can also be configured as GPIO PEL[3:1].

14.2.6.14 Receive Error (ERXER)

ERXER is an input signal which when asserted along with ERXDV signals that the PHY has detected an error in the current frame. When ERXDV is not asserted ERXER has no effect, and applies to MII mode operation.

These pins can also be configured as GPIO PEL0.

14.2.7 Queued Serial Peripheral Interface (QSPI) Signals

14.2.7.1 QSPI Synchronous Serial Output (QSPI_DOUT)

The QSPI_DOUT output provides the serial data from the QSPI and can be programmed to be driven on the rising or falling edge of QSPICLK. Each byte is sent msb first.

This pin can also be configured as GPIO PQS0.

14.2.7.2 QSPI Synchronous Serial Data Input (QSPI_DIN)

The QSPI_DIN input provides the serial data to the QSPI and can be programmed to be sampled on the rising or falling edge of QSPICLK. Each byte is written to RAM lsb first.

This pin can also be configured as GPIO PQS1.

14.2.7.3 QSPI Serial Clock (QSPI_CLK)

The QSPI serial clock (QSPI_CLK) provides the serial clock from the QSPI. The polarity and phase of QSPI_CLK are programmable. The output frequency is programmed according to the following formula, in which n can be any value between 2 and 255: QSPI_CLK = CLKOUT/(2n).

This pin can also be configured as GPIO PQS2.

14.2.7.4 QSPI Chip Selects (QSPI_CS[3:0])

The synchronous peripheral chip selects (QSPI_CS[3:0]) outputs provide QSPI peripheral chip selects that can be programmed to be active high or low.

This pin can also be configured as GPIO PQS[6:3].

14.2.8 FlexCAN Signals

14.2.8.1 FlexCAN Transmit (CANTX)

Controller Area Network Transmit data output.

This pin can also be configured as GPIO PAS2.

14.2.8.2 FlexCAN Receive (CANRX)

Controller Area Network Transmit data input.

This pin can also be configured as GPIO PAS3.

14.2.9 I²C Signals

The I²C module acts as a two-wire, bidirectional serial interface between the processor and peripherals with an I²C interface (such as LCD controller, A-to-D converter, or D-to-A converter). Devices connected to the I²C must have open-drain or open-collector outputs.

14.2.9.1 Serial Clock (SCL)

This bidirectional open-drain signal is the clock signal for the I²C interface. Either it is driven by the I²C module when the bus is in the master mode or it becomes the clock input when the I²C is in the slave mode.

This pin can also be configured as GPIO PAS0 or UTXD2.

14.2.9.2 Serial Data (SDA)

This bidirectional open-drain signal is the data input/output for the I²C interface.

This pin can also be configured as GPIO PAS1 or URXD2.

14.2.10 UART Module Signals

The signals in the following sections are used to transfer serial data between three UART modules and external peripherals.

14.2.10.1 Transmit Serial Data Output (UTXD[2:0])

UTXD[2:0] are the 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.

The UTXD[1:0] pins can be configured as GPIO ports PUA2 and PUA0. The UTXD2 output is offered on 3 pins and is a secondary function of the EMDC/ GPIO port PAS4 pin, CANTX/GPIO port PAS2 pin, and SCL/GPIO port PAS0 pin.

14.2.10.2 Receive Serial Data Input (URXD[2:0])

URXD[2:0] are the receiver serial data inputs for the UART modules. Data received on these pins is sampled on the rising edge of the serial clock source lsb first. When the UART clock is stopped for power-down mode, any transition on this pin restarts it.

The URXD[1:0] pins can be configured as GPIO ports PUA3 and PUA1. The URXD2 input is offered on 3 pins and is a secondary function of the EMDIO/GPIO port PAS5 pin, CANRX/GPIO port PAS3 pin, and SDA/GPIO port PAS1 pin.

14.2.10.3 Clear-to-Send ($\overline{\text{UCTS}}$[1:0])

The $\overline{\text{UCTS}}$[1:0] signals are the clear-to-send ($\overline{\text{CTS}}$) inputs, indicating to the UART modules that they can begin data transmission.

The $\overline{\text{UCTS}}$[1:0] inputs are each offered as secondary functions on four pins--DTIN2, DTOUT2, DTIN0 and DTOUT0.

14.2.10.4 Request-to-Send ($\overline{\text{URTS}}$[1:0])

The $\overline{\text{URTS}}$[1:0] signals are automatic request to send outputs from the UART modules. $\overline{\text{URTS}}$[1:0] can also be configured to be asserted and negated as a function of the Rx FIFO level.

The $\overline{\text{URTS}}$[1:0] outputs are each offered as secondary functions on four pins: DTIN3, DTOUT3, DTIN1 and DTOUT1.

14.2.11 General Purpose Timer Signals

These pins provide the external interface to the general purpose timer functions.

14.2.11.1 GPTA[3:0]

These pins provide the external interface to the timer A functions.

These pins can also be configured as GPIO PTA[3:0].

14.2.11.2 GPTB[3:0]

These pins provide the external interface to the timer B functions.

These pins can also be configured as GPIO PTB[3:0].

14.2.11.3 External Clock Input (SYNCA/SYNCB)

These pins are used to clear the clock for each of the two timers, and are provided as a means of synchronization to externally clocked or timed events.

14.2.12 DMA Timer Signals

This section describes the signals of the four DMA timer modules.

14.2.12.1 DMA Timer 0 Input (DTIN0)

The DMA timer 0 input (DTIN0) can be programmed to cause events to occur in DMA timer 0. It can either clock the event counter or provide a trigger to the timer value capture logic.

This pin can also be configured as GPIO PTD1, secondary function $\overline{\text{UCTS1}}$, or secondary function $\overline{\text{UCTS0}}$.

14.2.12.2 DMA Timer 0 Output (DTOUT0)

The programmable DMA timer output (DTOUT0) pulse or toggle on various timer events.

This pin can also be configured as GPIO PTD0, secondary function $\overline{\text{UCTS1}}$, or secondary function $\overline{\text{UCTS0}}$.

14.2.12.3 DMA Timer 1 Input (DTIN1)

The DMA timer 1 input (DTIN1) can be programmed to cause events to occur in DMA timer 1. This can either clock the event counter or provide a trigger to the timer value capture logic.

This pin can also be configured as GPIO PTD3, secondary function $\overline{\text{URTS1}}$, or secondary function $\overline{\text{URTS0}}$.

14.2.12.4 DMA Timer 1 Output (DTOUT1)

The programmable DMA timer output (DTOUT1) pulse or toggle on various timer events.

This pin can also be configured as GPIO PTD2, secondary function $\overline{\text{URTS1}}$, or secondary function $\overline{\text{URTS0}}$.

14.2.12.5 DMA Timer 2 Input (DTIN2)

The DMA timer 2 input (DTIN2) can be programmed to cause events to occur in DMA timer 2. It can either clock the event counter or provide a trigger to the timer value capture logic.

This pin can also be configured as GPIO PTC1, secondary function $\overline{\text{UCTS1}}$, or secondary function $\overline{\text{UCTS0}}$.

14.2.12.6 DMA Timer 2 Output (DTOUT2)

The programmable DMA timer output (DTOUT2) pulse or toggle on various timer events.

This pin can also be configured as GPIO PTC0, secondary function $\overline{\text{UCTS1}}$, or secondary function $\overline{\text{UCTS0}}$.

14.2.12.7 DMA Timer 3 Input (DTIN3)

The DMA timer 3 input (DTIN3) can be programmed as an input that causes events to occur in DMA timer 3. This can either clock the event counter or provide a trigger to the timer value capture logic. This pin can also be configured as GPIO PTC3, secondary function $\overline{\text{URTS1}}$, or secondary function $\overline{\text{URTS0}}$.

14.2.12.8 DMA Timer 3 Output (DTOUT3)

The programmable DMA timer output (DTOUT0) pulse or toggle on various timer events.

This pin can also be configured as GPIO PTC2, secondary function $\overline{\text{URTS1}}$, or secondary function $\overline{\text{URTS0}}$.

14.2.13 Analog-to-Digital Converter Signals

These pins provide the analog inputs to the QADC.

The PQA and PQB pins may also be used as general purpose digital I/O.

14.2.13.1 QADC Analog Input (AN0/ANW)

This PQB signal is the direct analog input AN0. When using external multiplexing this pin can also be configured as multiplexed input ANW.

This pin can also be configured as GPIO PQB0.

14.2.13.2 QADC Analog Input (AN1/ANX)

This PQB signal is the direct analog input AN1. When using external multiplexing this pin can also be configured as multiplexed input ANX.

This pin can also be configured as GPIO PQB1.

14.2.13.3 QADC Analog Input (AN2/ANY)

This PQB signal is the direct analog input AN2. When using external multiplexing this pin can also be configured as multiplexed input ANY.

This pin can also be configured as GPIO PQB2.

14.2.13.4 QADC Analog Input (AN3/ANZ)

This PQB signal is the direct analog input AN3. When using external multiplexing this pin can also be configured as multiplexed input ANZ.

This pin can also be configured as GPIO PQB3.

14.2.13.5 QADC Analog Input (AN52/MA0)

This PQA signal is the direct analog input AN52. When using external multiplexing this pin can also be configured as an output signal, MA0, to select the output of the external multiplexer.

This pin can also be configured as GPIO PQA0.

14.2.13.6 QADC Analog Input (AN53/MA1)

This PQA signal is the direct analog input AN53. When using external multiplexing this pin can also be configured as an output signal, MA1, to select the output of the external multiplexer.

This pin can also be configured as GPIO PQA1.

14.2.13.7 QADC Analog Input (AN55/TRIG1)

This PQA signal is the direct analog input AN55. This pin can also be configured as an input signal, TRIG1, to trigger the execution of one of the two queues.

This pin can also be configured as GPIO PQA3.

14.2.13.8 QADC Analog Input (AN56/TRIG2)

This PQA signal is the direct analog input AN56. This pin can also be configured as an input signal, TRIG2, to trigger the execution of one of the two queues.

This pin can also be configured as GPIO PQA4.

14.2.14 Debug Support Signals

These signals are used as the interface to the on-chip JTAG controller and also to interface to the BDM logic.

14.2.14.1 JTAG_EN

This input signal is used to select between multiplexed debug module and JTAG signals at reset. If JTAG_EN is low, the part is in normal and background debug mode (BDM); if it is high, it is in normal and JTAG mode.

14.2.14.2 Development Serial Clock/Test Reset (DSCLK/$\overline{\text{TRST}}$)

Debug mode operation: DSCLK is selected. DSCLK is the development serial clock for the serial interface to the debug module. The maximum DSCLK frequency is 1/5 CLKIN.

JTAG mode operation: $\overline{\text{TRST}}$ is selected. $\overline{\text{TRST}}$ asynchronously resets the internal JTAG controller to the test logic reset state, causing the JTAG instruction register to choose the bypass instruction. When this occurs, JTAG logic is benign and does not interfere with normal device functionality.

Although $\overline{\text{TRST}}$ is asynchronous, Freescale recommends that it makes an asserted-to-negated transition only while TMS is held high. $\overline{\text{TRST}}$ has an internal pull-up resistor so if it is not driven low, it defaults to a logic level of 1. If $\overline{\text{TRST}}$ is not used, it can be tied to ground or, if TCK is clocked, to V_{DD}. Tying $\overline{\text{TRST}}$ to ground places the JTAG controller in test logic reset state immediately. Tying it to V_{DD} causes the JTAG controller (if TMS is a logic level of 1) to eventually enter test logic reset state after 5 TCK clocks.

14.2.14.3 Breakpoint/Test Mode Select ($\overline{\text{BKPT}}$/TMS)

Debug mode operation: If JTAG_EN is low, $\overline{\text{BKPT}}$ is selected. $\overline{\text{BKPT}}$ signals a hardware breakpoint to the processor in debug mode.

JTAG mode operation: TMS is selected. The TMS input provides information to determine the JTAG test operation mode. The state of TMS and the internal 16-state JTAG controller state machine at the rising edge of TCK determine whether the JTAG controller holds its current state or advances to the next state. This directly controls whether JTAG data or instruction operations occur. TMS has an internal pull-up resistor so that if it is not driven low, it defaults to a logic level of 1. But if TMS is not used, it should be tied to V_{DD}.

14.2.14.4 Development Serial Input/Test Data (DSI/TDI)

Debug mode operation: If JTAG_EN is low, DSI is selected. DSI provides the single-bit communication for debug module commands.

JTAG mode operation: TDI is selected. TDI provides the serial data port for loading the various JTAG boundary scan, bypass, and instruction registers. Shifting in data depends on the state of the JTAG controller state machine and the instruction in the instruction register. Shifts occur on the TCK rising edge. TDI has an internal pull-up resistor, so when not driven low it defaults to high. But if TDI is not used, it should be tied to V_{DD}.

14.2.14.5 Development Serial Output/Test Data (DSO/TDO)

Debug mode operation: DSO is selected. DSO provides single-bit communication for debug module responses.

JTAG mode operation: TDO is selected. The TDO output provides the serial data port for outputting data from JTAG logic. Shifting out data depends on the JTAG controller state machine and the instruction in the instruction register. Data shifting occurs on the falling edge of TCK. When TDO is not outputting test data, it is three-stated. TDO can be three-stated to allow bused or parallel connections to other devices having JTAG.

14.2.14.6 Test Clock (TCLK)

TCK is the dedicated JTAG test logic clock independent of the processor clock. Various JTAG operations occur on the rising or falling edge of TCK. Holding TCK high or low for an indefinite period does not cause JTAG test logic to lose state information. If TCK is not used, it must be tied to ground.

14.2.14.7 Debug Data (DDATA[3:0])

Debug data signals (DDATA[3:0]) display captured processor addresses, data and breakpoint status.

These pins can also be configured as GPIO PDD[7:4].

14.2.14.8 Processor Status Outputs (PST[3:0])

PST[3:0] outputs indicate core status, as shown below in Table 14-7. Debug mode timing is synchronous with the processor clock; status is unrelated to the current bus transfer.

These pins can also be configured as GPIO PDD[3:0].

Table 14-7. Processor Status Encoding

PST[3:0]	Definition
0000	Continue execution
0001	Begin execution of an instruction
0010	Reserved
0011	Entry into user mode
0100	Begin execution of PULSE and WDDATA instruction
0101	Begin execution of taken branch
0110	Reserved
0111	Begin execution of RTE instruction
1000	Begin one-byte transfer on DDATA
1001	Begin two-byte transfer on DDATA
1010	Begin three-byte transfer on DDATA
1011	Begin four-byte transfer on DDATA
1100	Exception Processing
1101	Emulator-Mode Exception Processing
1110	Processor is stopped
1111	Processor is halted

14.2.15 Test Signals

14.2.15.1 Test (TEST)

This input signal is reserved for factory testing only and should be connected to VSS to prevent unintentional activation of test functions.

14.2.16 Power and Reference Signals

These signals provide system power, ground and references to the device. Multiple pins are provided for adequate current capability. All power supply pins must have adequate bypass capacitance for high-frequency noise suppression.

14.2.16.1 QADC Analog Reference (VRH, VRL)

These signals serve as the high (VRH) and low (VRL) reference potentials for the analog converter in the QADC.

14.2.16.2 QADC Analog Supply (VDDA, VSSA)

These are dedicated power supply signals to isolate the sensitive QADC analog circuitry from the normal levels of noise present on the digital power supply.

14.2.16.3 PLL Analog Supply (VDDPLL, VSSPLL)

These are dedicated power supply signals to isolate the sensitive PLL analog circuitry from the normal levels of noise present on the digital power supply.

14.2.16.4 QADC Positive Supply (VDDH)

This pin supplies positive power to the ESD structures in the QADC pads.

14.2.16.5 Power for Flash Erase/Program (VPP)

This pin is used for Flash stress testing and can be left unconnected in normal device operation.

14.2.16.6 Power and Ground for Flash Array (VDDF, VSSF)

These signals supply a power and ground to the Flash array.

14.2.16.7 Standby Power (VSTBY)

This pin is used to provide standby voltage to the RAM array if VDD is lost.

14.2.16.8 Positive Supply (VDD)

This pin supplies positive power to the core logic and I/O pads.

14.2.16.9 Ground (VSS)

This pin is the negative supply (ground) to the chip.

Chapter 15
Synchronous DRAM Controller Module

This chapter describes configuration and operation of the synchronous DRAM (SDRAM) controller. It begins with a general description and brief glossary, and includes a description of signals involved in DRAM operations. The remainder of the chapter describes the programming model and signal timing, as well as the command set required for synchronous operations. It also includes extensive examples that the designer can follow to better understand how to configure the DRAM controller for synchronous operations.

15.1 Overview

The synchronous DRAM controller module provides glueless integration of SDRAM with the ColdFire product. The key features of the DRAM controller include the following:

- Support for two independent blocks of SDRAM
- Interface to standard SDRAM components
- Programmable $\overline{\text{SRAS}}$, $\overline{\text{SCAS}}$, and refresh timing
- Support for 8-, 16-, and 32-bit wide SDRAM blocks

15.1.1 Definitions

The following terminology is used in this chapter:

- SDRAM block: Any group of DRAM memories selected by one of the $\overline{\text{SRAS}}$[1:0] signals. Thus, the processor can support two independent memory blocks. The base address of each block is programmed in the DRAM address and control registers (DACR0 and DACR1).
- SDRAM: RAMs that operate like asynchronous DRAMs but with a synchronous clock, a pipelined, multiple-bank architecture, and a faster speed.
- SDRAM bank: An internal partition in an SDRAM device. For example, a 64-Mbit SDRAM component might be configured as four 512K x 32 banks. Banks are selected through the SDRAM component's bank select lines.

15.1.2 Block Diagram and Major Components

The basic components of the SDRAM controller are shown in Figure 15-1.

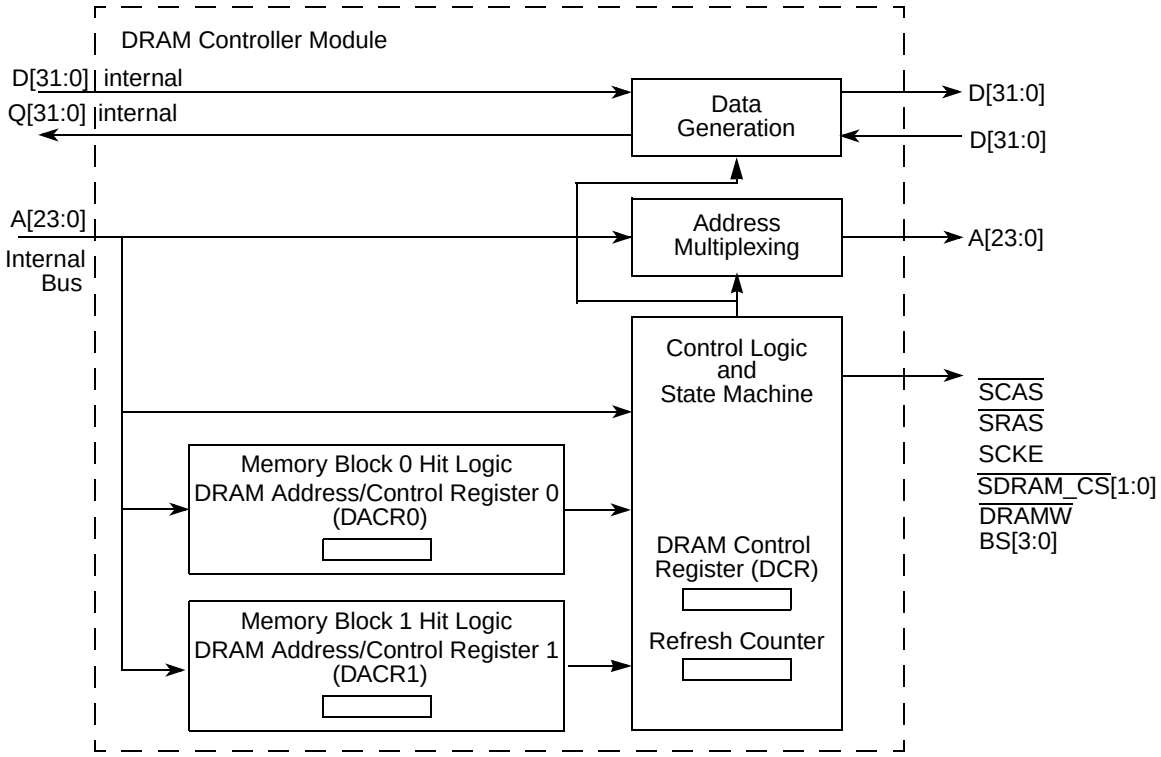


Figure 15-1. Synchronous DRAM Controller Block Diagram

The DRAM controller's major components are as follows:

- DRAM address and control registers (DACR0 and DACR1)—The DRAM controller consists of two configuration register units, one for each supported memory block. DACR0 is accessed at IPSBAR + 0x048; DACR1 is accessed at IPSBAR + 0x050. The register information is passed on to the hit logic.
- Control logic and state machine—Generates all SDRAM signals, taking hit information and bus-cycle characteristic data from the block logic in order to generate SDRAM accesses. Handles refresh requests from the refresh counter.
  - DRAM control register (DCR)—Contains data to control refresh operation of the DRAM controller. Both memory blocks are refreshed concurrently as controlled by DCR[RC].
  - Refresh counter—Determines when refresh should occur; controlled by the value of DCR[RC]. It generates a refresh request to the control block.
- Hit logic—Compares address and attribute signals of a current SDRAM bus cycle to both DACRs to determine if an SDRAM block is being accessed. Hits are passed to the control logic along with characteristics of the bus cycle to be generated.
- Address multiplexing—Multiplexes addresses to allow column and row addresses to share pins. This allows glueless interface to SDRAMs.
- Data Generation—Controls the data input and data output transmission between the on-platform and off-platform data buses.

15.2 SDRAM Controller Operation

By running synchronously with the system clock, SDRAM can (after an initial latency period) be accessed on every clock; 5-1-1-1 is a typical burst rate to the SDRAM. Unlike the MCF5272, this processor does not have an independent SDRAM clock signal. For this processor, the timing of the SDRAM controller is controlled by the CLKOUT signal.

Note that because the processor cannot have more than one page open at a time, it does not support interleaving.

SDRAM controllers are more sophisticated than asynchronous DRAM controllers. Not only must they manage addresses and data, but they must send special commands for such functions as precharge, read, write, burst, auto-refresh, and various combinations of these functions. Table 15-1 lists common SDRAM commands.

Table 15-1. SDRAM Commands

Command	Definition
ACTV	Activate. Executed before READ or WRITE executes; SDRAM registers and decodes row address.
MRS	Mode register set.
NOP	No-op. Does not affect SDRAM state machine; DRAM controller control signals negated; $\overline{\text{SDRAM\_CS}}$[1:0] asserted.
PALL	Precharge all. Precharges all internal banks of an SDRAM component; executed before new page is opened.
READ	Read access. SDRAM registers column address and decodes that a read access is occurring.
REF	Refresh. Refreshes internal bank rows of an SDRAM component.
SELF	Self refresh. Refreshes internal bank rows of an SDRAM component when it is in low-power mode.
SELFX	Exit self refresh. This command is sent to the DRAM controller when DCR[IS] is cleared.
WRITE	Write access. SDRAM registers column address and decodes that a write access is occurring.

SDRAMs operate differently than asynchronous DRAMs, particularly in the use of data pipelines and commands to initiate special actions. Commands are issued to memory using specific encodings on address and control pins. Soon after system reset, a command must be sent to the SDRAM mode register to configure SDRAM operating parameters.

15.2.1 DRAM Controller Signals

Table 15-2 describes the behavior of DRAM signals in synchronous mode.

Table 15-2. Synchronous DRAM Signal Connections

Signal	Description
$\overline{\text{SRAS}}$	Synchronous row address strobe. Indicates a valid SDRAM row address is present and can be latched by the SDRAM. $\overline{\text{SRAS}}$ should be connected to the corresponding SDRAM $\overline{\text{SRAS}}$.
$\overline{\text{SCAS}}$	Synchronous column address strobe. Indicates a valid column address is present and can be latched by the SDRAM. $\overline{\text{SCAS}}$ should be connected to the corresponding SDRAM $\overline{\text{SCAS}}$.
$\overline{\text{DRAMW}}$	DRAM read/write. Asserted for write operations and negated for read operations.
$\overline{\text{SDRAM\_CS}}$[1:0]	Row address strobe. Select each memory block of SDRAMs connected to the processor. One $\overline{\text{SDRAM\_CS}}$ signal selects one SDRAM block and connects to the corresponding $\overline{\text{CS}}$ signals.
SCKE	Synchronous DRAM clock enable. Connected directly to the CKE (clock enable) signal of SDRAMs. Enables and disables the clock internal to SDRAM. When CKE is low, memory can enter a power-down mode in which operations are suspended or capable of entering self-refresh mode. SCKE functionality is controlled by DCR[COC]. For designs using external multiplexing, setting COC allows SCKE to provide command-bit functionality.
BS[3:0]	Column address strobe. BS[3:0] function as byte enables to the SDRAMs. They connect to the BS signals (or mask qualifiers) of the SDRAMs.

15.2.2 Memory Map for SDRAMC Registers

The DRAM controller registers memory map is shown in Table 15-3.

Table 15-3. DRAM Controller Registers

IPSBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x040	DRAM control register (DCR) [p. 15-4]		—	
0x044	—			
0x048	DRAM address and control register 0 (DACR0) [p. 15-6]			
0x04C	DRAM mask register block 0 (DMR0) [p. 15-8]			
0x050	DRAM address and control register 1 (DACR1) [p. 15-6]			
0x054	DRAM mask register block 1 (DMR1) [p. 15-8]			

15.2.2.1 DRAM Control Register (DCR)

The DCR, shown in Figure 15-2, controls refresh logic.

	15	14	13	12	11	10	9	8	0
Field	—		NAM	COC	IS	RTIM		RC	
Reset	Uninitialized								
R/W	R/W								
Addr	IPSBAR + 0x040								

Figure 15-2. DRAM Control Register (DCR)

Table 15-4 describes DCR fields.

Table 15-4. DCR Field Descriptions

Bits	Name	Description
15-14	—	Reserved, should be cleared.
13	NAM	No address multiplexing. Some implementations require external multiplexing. For example, when linear addressing is required, the SDRAM should not multiplex addresses on SDRAM accesses. 0 The SDRAM controller multiplexes the external address bus to provide column addresses. 1 The SDRAM controller does not multiplex the external address bus to provide column addresses.
12	COC	Command on SDRAM clock enable (SCKE). Implementations that use external multiplexing (NAM = 1) must support command information to be multiplexed onto the SDRAM address bus. 0 SCKE functions as a clock enable; self-refresh is initiated by the SDRAM controller through DCR[IS]. 1 SCKE drives command information. Because SCKE is not a clock enable, self-refresh cannot be used (setting DCR[IS]). Thus, external logic must be used if this functionality is desired. External multiplexing is also responsible for putting the command information on the proper address bit.
11	IS	Initiate self-refresh command. 0 Take no action or issue a SELFX command to exit self refresh. 1 If DCR[COC] = 0, the SDRAM controller sends a SELF command to both SDRAM blocks to put them in low-power, self-refresh state where they remain until IS is cleared. When IS is cleared, the controller sends a SELFX command for the SDRAMs to exit self-refresh. The refresh counter is suspended while the SDRAMs are in self-refresh; the SDRAM controls the refresh period.
10–9	RTIM	Refresh timing. Determines the timing operation of auto-refresh in the SDRAM controller. Specifically, it determines the number of bus clocks inserted between a REF command and the next possible ACTV command. This same timing is used for both memory blocks controlled by the SDRAM controller. This corresponds to t_{RC} in the SDRAM specifications. 00 3 clocks 01 6 clocks 1x 9 clocks
8–0	RC	Refresh count. Controls refresh frequency. The number of bus clocks between refresh cycles is (RC + 1) x 16. Refresh can range from 16–8192 bus clocks to accommodate both standard and low-power SDRAMs with bus clock operation from less than 2 MHz to greater than 50 MHz. The following example calculates RC for an auto-refresh period for 4096 rows to receive 64 ms of refresh every 15.625 µs for each row (1031 bus clocks at 66 MHz). This operation is the same as in asynchronous mode. # of bus clocks = 1031 = (RC field + 1) x 16 RC = (1031 bus clocks/16) -1 = 63.44, which rounds to 63; therefore, RC = 0x3F.

15.2.2.2 DRAM Address and Control Registers (DACR0/DACR1)

The DACR*n* registers, shown in Figure 15-3, contain the base address compare value and the control bits for memory blocks 0 and 1 of the SDRAM controller. Address and timing are also controlled by bits in DACR*n*.

	31 18	17 16	15	14	13 12	11	10 9 8	7	6	5 4	3	2 0
Field	BA	—	RE	—	CASL	—	CBM	—	IMRS	PS	IP	—
Reset	Uninitialized		0	Uninitialized					0	Uninitialized		
R/W	R/W											
Address	IPSBAR+0x048 (DACR0); 0x050 (DACR1)											

Figure 15-3. DRAM Address and Control Register (DACR*n*)

Table 15-5 describes DACR*n* fields.

Table 15-5. DACR*n* Field Descriptions

Bit	Name	Description
31–18	BA	Base address register. With DCMR[BAM], determines the address range in which the associated DRAM block is located. Each BA bit is compared with the corresponding address of the current bus cycle. If all unmasked bits match, the address hits in the associated DRAM block. BA functions the same as in asynchronous operation.
17–16	—	Reserved, should be cleared.
15	RE	Refresh enable. Determines when the DRAM controller generates a refresh cycle to the DRAM block. 0 Do not refresh associated DRAM block 1 Refresh associated DRAM block
14	—	Reserved, should be cleared.
13–12	CASL	$\overline{\text{CAS}}$ latency. Affects the following SDRAM timing specifications. Timing nomenclature varies with manufacturers. Refer to the SDRAM specification for the appropriate timing nomenclature:
11	—	Reserved, should be cleared.

Parameter	Number of Bus Clocks			
	CASL= 00	CASL = 01	CASL= 10	CASL= 11
t_{RCD}—$\overline{\text{SRAS}}$ assertion to $\overline{\text{SCAS}}$ assertion	1	2	3	3
t_{CASL}—$\overline{\text{SCAS}}$ assertion to data out	1	2	3	3
t_{RAS}—ACTV command to precharge command	2	4	6	6
t_{RP}—Precharge command to ACTV command	1	2	3	3
t_{RWL},t_{RDL}—Last data input to precharge command	1	1	1	1
t_{EP}—Last data out to precharge command	1	1	1	1

Table 15-5. DACR*n* Field Descriptions (continued)

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>10–8</td><td>CBM</td><td>Command and bank MUX [2:0]. Because different SDRAM configurations cause the command and bank select lines to correspond to different addresses, these resources are programmable. CBM determines the addresses onto which these functions are multiplexed.
Note: It is important to set CBM according to the location of the command bit.

<table>
<tr><th>CBM</th><th>Command Bit</th><th>Bank Select Bits</th></tr>
<tr><td>000</td><td>17</td><td>18 and up</td></tr>
<tr><td>001</td><td>18</td><td>19 and up</td></tr>
<tr><td>010</td><td>19</td><td>20 and up</td></tr>
<tr><td>011</td><td>20</td><td>21 and up</td></tr>
<tr><td>100</td><td>21</td><td>22 and up</td></tr>
<tr><td>101</td><td>22</td><td>23 and up</td></tr>
<tr><td>110</td><td>23</td><td>24 and up</td></tr>
<tr><td>111</td><td>24</td><td>25 and up</td></tr>
</table>
This encoding and the address multiplexing scheme handle common SDRAM organizations. Bank select bits include a base bit and all address bits above for SDRAMs with multiple bank select bits.</td></tr>
<tr><td>7</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>6</td><td>IMRS</td><td>Initiate mode register set (MRS) command. Setting IMRS generates a MRS command to the associated SDRAMs. In initialization, IMRS should be set only after all DRAM controller registers are initialized and PALL and REFRESH commands have been issued. After IMRS is set, the next access to an SDRAM block programs the SDRAM’s mode register. Thus, the address of the access should be programmed to place the correct mode information on the SDRAM address pins. Because the SDRAM does not register this information, it doesn’t matter if the IMRS access is a read or a write or what, if any, data is put onto the data bus. The DRAM controller clears IMRS after the MRS command finishes.
0 Take no action
1 Initiate MRS command</td></tr>
<tr><td>5–4</td><td>PS</td><td>Port size. Indicates the port size of the associated block of SDRAM, which allows for dynamic sizing of associated SDRAM accesses. PS functions the same in asynchronous operation.
00 32-bit port
01 8-bit port
1x 16-bit port</td></tr>
<tr><td>3</td><td>IP</td><td>Initiate precharge all (PALL) command. The DRAM controller clears IP after the PALL command is finished. Accesses via IP should be no wider than the port size programmed in PS.
0 Take no action.
1 A PALL command is sent to the associated SDRAM block. During initialization, this command is executed after all DRAM controller registers are programmed. After IP is set, the next write to an appropriate SDRAM address generates the PALL command to the SDRAM block.</td></tr>
<tr><td>2–0</td><td>—</td><td>Reserved, should be cleared.</td></tr>
</table>

15.2.2.3 DRAM Controller Mask Registers (DMR0/DMR1)

The DMR*n*, Figure 15-4, includes mask bits for the base address and for address attributes.

	31 18	17 9	8	7	6	5	4	3	2	1	0
Field	BAM	—	WP	—	C/I	AM	SC	SD	UC	UD	V
Reset	Uninitialized										0
R/W	R/W										
Addr	IPSBAR + 0x04C (DMR0), 0x054 (DMR1)										

Figure 15-4. DRAM Controller Mask Registers (DMR*n*)

Table 15-6 describes DMR*n* fields.

Table 15-6. DMR*n* Field Descriptions

Bits	Name	Description
31–18	BAM	Base address mask. Masks the associated DACR*n*[BA]. Lets the DRAM controller connect to various DRAM sizes. Mask bits need not be contiguous (see Section 15.3, “SDRAM Example.”) 0 The associated address bit is used in decoding the DRAM hit to a memory block. 1 The associated address bit is not used in the DRAM hit decode.
17–9	—	Reserved, should be cleared.
8	WP	Write protect. Determines whether the associated block of DRAM is write protected. 0 Allow write accesses 1 Ignore write accesses. The DRAM controller ignores write accesses to the memory block and an address exception occurs. Write accesses to a write-protected DRAM region are compared in the chip select module for a hit. If no hit occurs, an external bus cycle is generated. If this external bus cycle is not acknowledged, an access exception occurs.
7	—	Reserved, should be cleared.
6–1	AM*x*	Address modifier masks. Determine which accesses can occur in a given DRAM block. 0 Allow access type to hit in DRAM 1 Do not allow access type to hit in DRAM
0	V	Valid. Cleared at reset to ensure that the DRAM block is not erroneously decoded. 0 Do not decode DRAM accesses. 1 Registers controlling the DRAM block are initialized; DRAM accesses can be decoded.

Bit	Associated Access Type	Access Definition
C/I	CPU space/interrupt acknowledge	MOVEC instruction or interrupt acknowledge cycle
AM	Alternate master	DMA master
SC	Supervisor code	Any supervisor-only instruction access
SD	Supervisor data	Any data fetched during the instruction access
UC	User code	Any user instruction
UD	User data	Any user data

15.2.3 General Synchronous Operation Guidelines

To reduce system logic and to support a variety of SDRAM sizes, the DRAM controller provides SDRAM control signals as well as a multiplexed row address and column address to the SDRAM.

15.2.3.1 Address Multiplexing

Table 15-7 shows the generic address multiplexing scheme for SDRAM configurations. All possible address connection configurations can be derived from this table.

NOTE

Because the processor has 24 extermal address lines, the maximum SDRAM address size is 128 Mbits.

The following tables provide a more comprehensive, step-by-step way to determine the correct address line connections for interfacing the ColdFire processor to the SDRAM. To use the tables, find the one that corresponds to the number of column address lines on the SDRAM and to the port size as seen by the processor, which is not necessarily the SDRAM port size. For example, if two 1M x 16-bit SDRAMs together form a 1M x 32-bit memory, the port size is 32 bits. Most SDRAMs likely have fewer address lines than are shown in the tables, so follow only the connections shown until all SDRAM address lines are connected.

Table 15-7. Generic Address Multiplexing Scheme

Address Pin	Row Address	Column Address	Notes Relating to Port Sizes
17	17	0	8-bit port only
16	16	1	8- and 16-bit ports only
15	15	2	
14	14	3	
13	13	4	
12	12	5	
11	11	6	
10	10	7	
9	9	8	
17	17	16	32-bit port only
18	18	17	16-bit port only or 32-bit port with only 8 column address lines
19	19	18	16-bit port only when at least 9 column address lines are used
20	20	19	
21	21	20	
22	22	21	
23	23	22	

Table 15-8. Processor to SDRAM Interface (8-Bit Port, 9-Column Address Lines)

Processor Pins	A17	A16	A15	A14	A13	A12	A11	A10	A9	A18	A19	A20	A21	A22	A23
Row	17	16	15	14	13	12	11	10	9	18	19	20	21	22	23
Column	0	1	2	3	4	5	6	7	8						
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14

Table 15-9. Processor to SDRAM Interface (8-Bit Port,10-Column Address Lines)

Processor Pins	A17	A16	A15	A14	A13	A12	A11	A10	A9	A19	A20	A21	A22	A23
Row	17	16	15	14	13	12	11	10	9	19	20	21	22	23
Column	0	1	2	3	4	5	6	7	8	18				
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13

Table 15-10. Processor to SDRAM Interface (8-Bit Port,11-Column Address Lines)

Processor Pins	A17	A16	A15	A14	A13	A12	A11	A10	A9	A19	A21	A22	A23
Row	17	16	15	14	13	12	11	10	9	19	21	22	23
Column	0	1	2	3	4	5	6	7	8	18	20		
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12

Table 15-11. Processor to SDRAM Interface (8-Bit Port,12-Column Address Lines)

Processor Pins	A17	A16	A15	A14	A13	A12	A11	A10	A9	A19	A21	A23
Row	17	16	15	14	13	12	11	10	9	19	21	23
Column	0	1	2	3	4	5	6	7	8	18	20	22
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11

Table 15-12. Processor to SDRAM Interface (8-Bit Port,13-Column Address Lines)

Processor Pins	A17	A16	A15	A14	A13	A12	A11	A10	A9	A19	A21	A23
Row	17	16	15	14	13	12	11	10	9	19	21	23
Column	0	1	2	3	4	5	6	7	8	18	20	22
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11

Table 15-13. Processor to SDRAM Interface (16-Bit Port, 8-Column Address Lines)

Processor Pins	A16	A15	A14	A13	A12	A11	A10	A9	A17	A18	A19	A20	A21	A22	A23
Row	16	15	14	13	12	11	10	9	17	18	19	20	21	22	23
Column	1	2	3	4	5	6	7	8							
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14

Table 15-14. Processor to SDRAM Interface (16-Bit Port, 9-Column Address Lines)

Processor Pins	A16	A15	A14	A13	A12	A11	A10	A9	A18	A19	A20	A21	A22	A23
Row	16	15	14	13	12	11	10	9	18	19	20	21	22	23
Column	1	2	3	4	5	6	7	8	17					
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13

Table 15-15. Processor to SDRAM Interface (16-Bit Port, 10-Column Address Lines)

Processor Pins	A16	A15	A14	A13	A12	A11	A10	A9	A18	A20	A21	A22	A23
Row	16	15	14	13	12	11	10	9	18	20	21	22	23
Column	1	2	3	4	5	6	7	8	17	19			
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12

Table 15-16. Processor to SDRAM Interface (16-Bit Port, 11-Column Address Lines)

Processor Pins	A16	A15	A14	A13	A12	A11	A10	A9	A18	A20	A22	A23
Row	16	15	14	13	12	11	10	9	18	20	22	23
Column	1	2	3	4	5	6	7	8	17	19	21	
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11

Table 15-17. Processor to SDRAM Interface (16-Bit Port, 12-Column Address Lines)

Processor Pins	A16	A15	A14	A13	A12	A11	A10	A9	A18	A20	A22
Row	16	15	14	13	12	11	10	9	18	20	22
Column	1	2	3	4	5	6	7	8	17	19	21
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10

Table 15-18. Processor to SDRAM Interface (16-Bit Port, 13-Column-Address Lines)

Processor Pins	A16	A15	A14	A13	A12	A11	A10	A9	A18	A20	A22
Row	16	15	14	13	12	11	10	9	18	20	22
Column	1	2	3	4	5	6	7	8	17	19	21
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10

Table 15-19. Processor to SDRAM Interface (32-Bit Port, 8-Column Address Lines)

Processor Pins	A15	A14	A13	A12	A11	A10	A9	A17	A18	A19	A20	A21	A22	A23
Row	15	14	13	12	11	10	9	17	18	19	20	21	22	23
Column	2	3	4	5	6	7	8	16						
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13

Table 15-20. Processor to SDRAM Interface (32-Bit Port, 9-Column Address Lines)

Processor Pins	A15	A14	A13	A12	A11	A10	A9	A17	A19	A20	A21	A22	A23
Row	15	14	13	12	11	10	9	17	19	20	21	22	23
Column	2	3	4	5	6	7	8	16	18				
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12

Table 15-21. Processor to SDRAM Interface (32-Bit Port, 10-Column Address Lines)

Processor Pins	A15	A14	A13	A12	A11	A10	A9	A17	A19	A21	A22	A23
Row	15	14	13	12	11	10	9	17	19	21	22	23
Column	2	3	4	5	6	7	8	16	18	20		
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11

Table 15-22. Processor to SDRAM Interface (32-Bit Port, 11-Column Address Lines)

Processor Pins	A15	A14	A13	A12	A11	A10	A9	A17	A19	A21	A23
Row	15	14	13	12	11	10	9	17	19	21	23
Column	2	3	4	5	6	7	8	16	18	20	22
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10

Table 15-23. Processor to SDRAM Interface (32-Bit Port, 12-Column Address Lines)

Processor Pins	A15	A14	A13	A12	A11	A10	A9	A17	A19	A21	A23
Row	15	14	13	12	11	10	9	17	19	21	23
Column	2	3	4	5	6	7	8	16	18	20	22
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10

15.2.3.2 SDRAM Byte Strobe Connections

Figure 15-5 shows SDRAM connections for port sizes of 32, 16, or 8 bits.

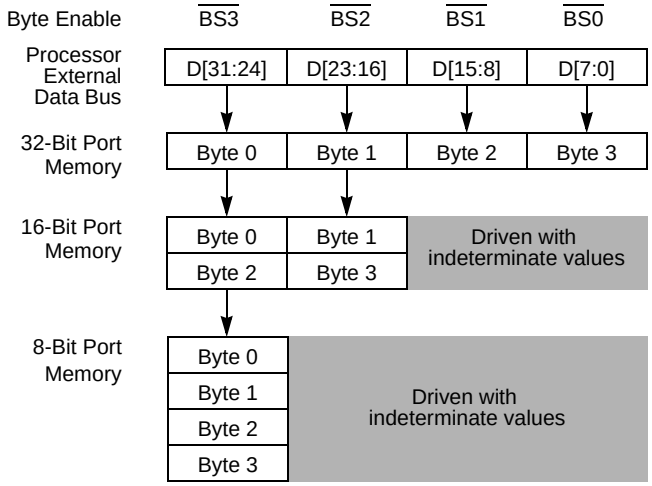


Figure 15-5. Connections for External Memory Port Sizes

15.2.3.3 Interfacing Example

The tables in the previous section can be used to configure the interface in the following example. To interface one 2M x 32-bit x 4 bank SDRAM component (8 columns), use the connections shown in Table 15-24.

Table 15-24. SDRAM Hardware Connections

SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10 = CMD	BA0	BA1
Processor Pins	A15	A14	A13	A12	A11	A10	A9	A17	A18	A19	A20	A21	A22

15.2.3.4 Burst Page Mode

SDRAM can efficiently provide data when an SDRAM page is opened. As soon as $\overline{\text{SCAS}}$ is issued, the SDRAM accepts a new address and asserts $\overline{\text{SCAS}}$ every CLKOUT for as long as accesses occur in that page. In burst page mode, there are multiple read or write operations for every ACTV command in the SDRAM if the requested transfer size exceeds the port size of the associated SDRAM. The primary cycle of the transfer generates the ACTV and READ or WRITE commands; secondary cycles generate only READ or WRITE commands. As soon as the transfer completes, the PALL command is generated to prepare for the next access.

Note that in synchronous operation, burst mode and address incrementing during burst cycles are controlled by the DRAM controller. Thus, instead of the SDRAM enabling its internal burst incrementing capability, the processor controls this function. This means that the burst function that is enabled in the mode register of SDRAMs must be disabled when interfacing to the processor.

Figure 15-6 shows a burst read operation. In this example, DACR[CASL] = 01 for an $\overline{\text{SRAS}}$-to-$\overline{\text{SCAS}}$ delay (t_{RCD}) of 2 system clock cycles. Because t_{RCD} is equal to the read CAS latency ($\overline{\text{SCAS}}$ assertion to data out), this value is also 2 system clock cycles. Notice that NOPs are executed until the last data is read. A PALL command is executed one cycle after the last data transfer.

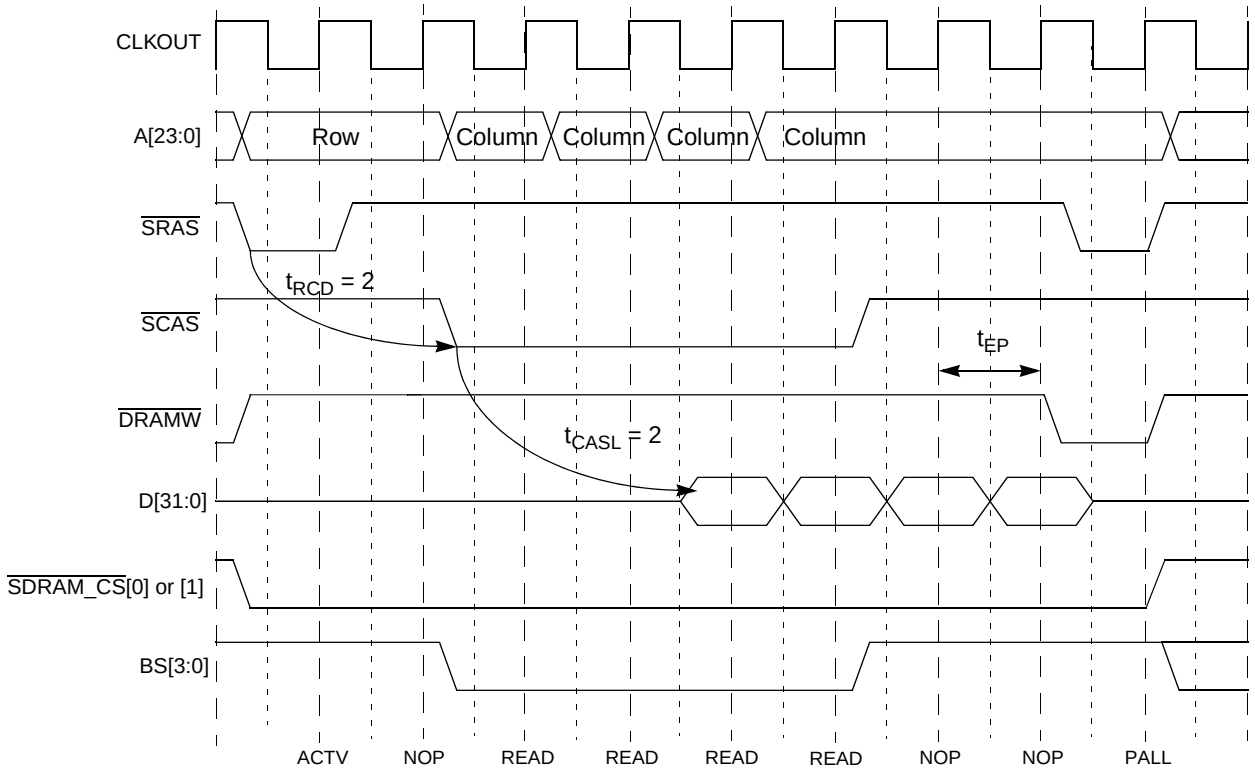


Figure 15-6. Burst Read SDRAM Access

Figure 15-7 shows the burst write operation. In this example, DACR[CASL] = 01, which creates an $\overline{\text{SRAS}}$-to-$\overline{\text{SCAS}}$ delay (t_{RCD}) of 2 system clock cycles. Note that data is available upon $\overline{\text{SCAS}}$ assertion and a burst write cycle completes two cycles sooner than a burst read cycle with the same t_{RCD}. The next bus cycle is initiated sooner, but cannot begin an SDRAM cycle until the precharge-to-ACTV delay completes.

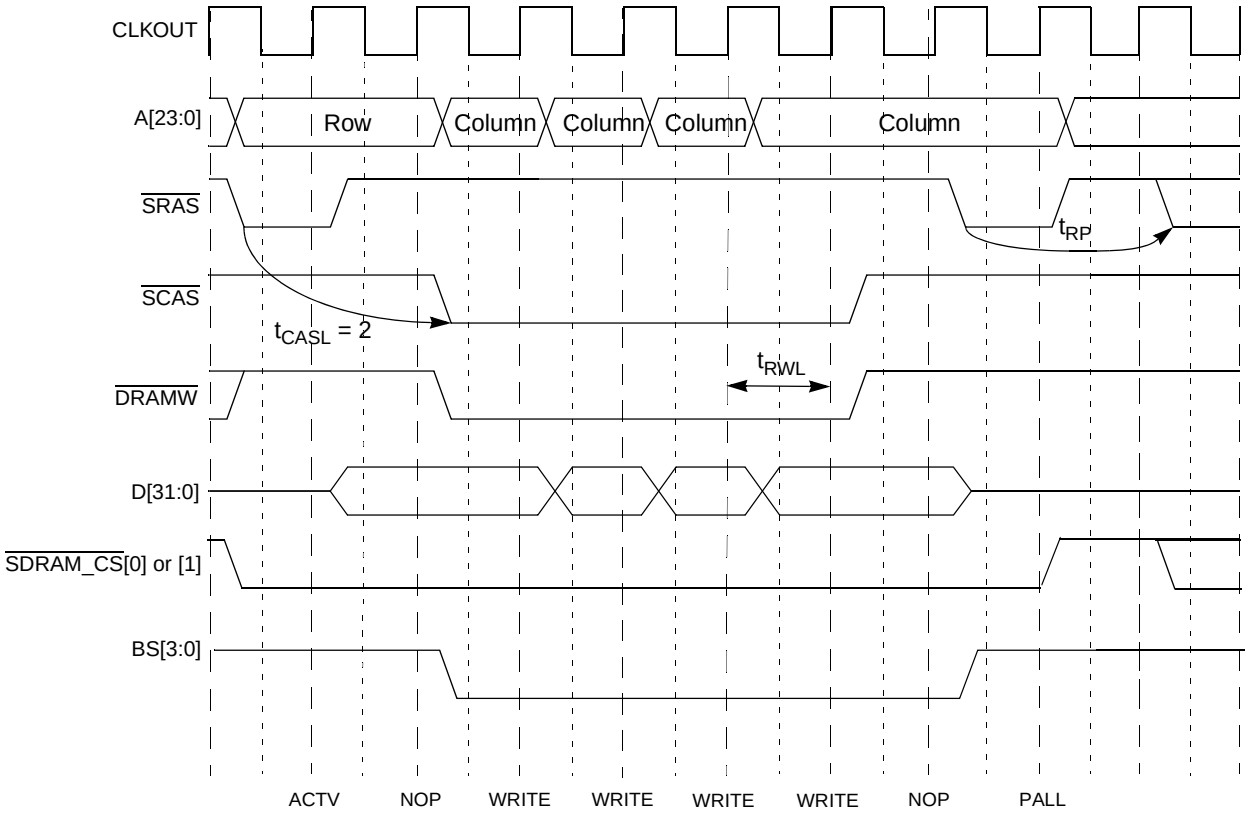


Figure 15-7. Burst Write SDRAM Access

Accesses in synchronous burst page mode always cause the following sequence:

1. ACTV command
2. NOP commands to assure $\overline{\text{SRAS}}$-to-$\overline{\text{SCAS}}$ delay (if CAS latency is 1, there are no NOP commands).
3. Required number of READ or WRITE commands to service the transfer size with the given port size.
4. Some transfers need more NOP commands to assure the ACTV-to-precharge delay.
5. PALL command
6. Required number of idle clocks inserted to assure precharge-to-ACTV delay.

15.2.3.5 Auto-Refresh Operation

The DRAM controller is equipped with a refresh counter and control. This logic is responsible for providing timing and control to refresh the SDRAM without user interaction. Once the refresh counter is set, and refresh is enabled, the counter counts to zero. At this time, an internal refresh request flag is set and the counter begins counting down again. The DRAM controller completes any active burst operation and then performs a PALL operation. The DRAM controller then initiates a refresh cycle and clears the refresh request flag. This refresh cycle includes a delay from any precharge to the auto-refresh command, the auto-refresh command, and then a delay until any ACTV command is allowed. Any SDRAM access initiated during the auto-refresh cycle is delayed until the cycle is completed.

Figure 15-8 shows the auto-refresh timing. In this case, there is an SDRAM access when the refresh request becomes active. The request is delayed by the precharge to ACTV delay programmed into the active SDRAM bank by the CAS bits. The REF command is then generated and the delay required by DCR[RTIM] is inserted before the next ACTV command is generated. In this example, the next bus cycle is initiated, but does not generate an SDRAM access until T_{RC} is finished. Because both chip selects are active during the REF command, it is passed to both blocks of external SDRAM.

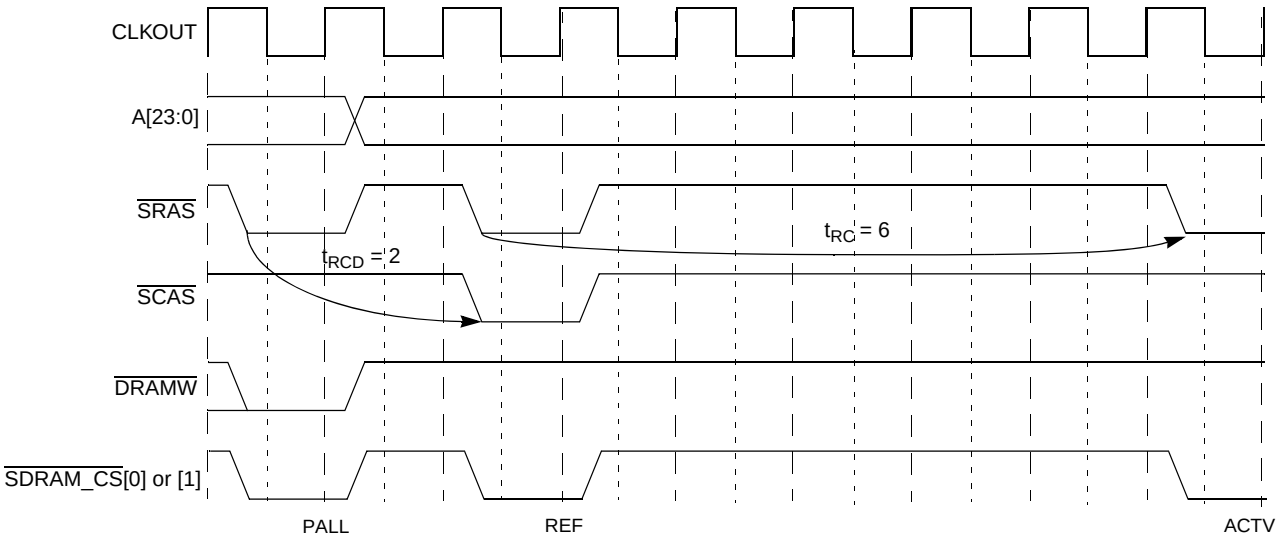


Figure 15-8. Auto-Refresh Operation

15.2.3.6 Self-Refresh Operation

Self-refresh is a method of allowing the SDRAM to enter into a low-power state, while at the same time to perform an internal refresh operation and to maintain the integrity of the data stored in the SDRAM. The DRAM controller supports self-refresh with DCR[IS]. When IS is set, the SELF command is sent to the SDRAM. When IS is cleared, the SELFX command is sent to the DRAM controller. Figure 15-9 shows the self-refresh operation.

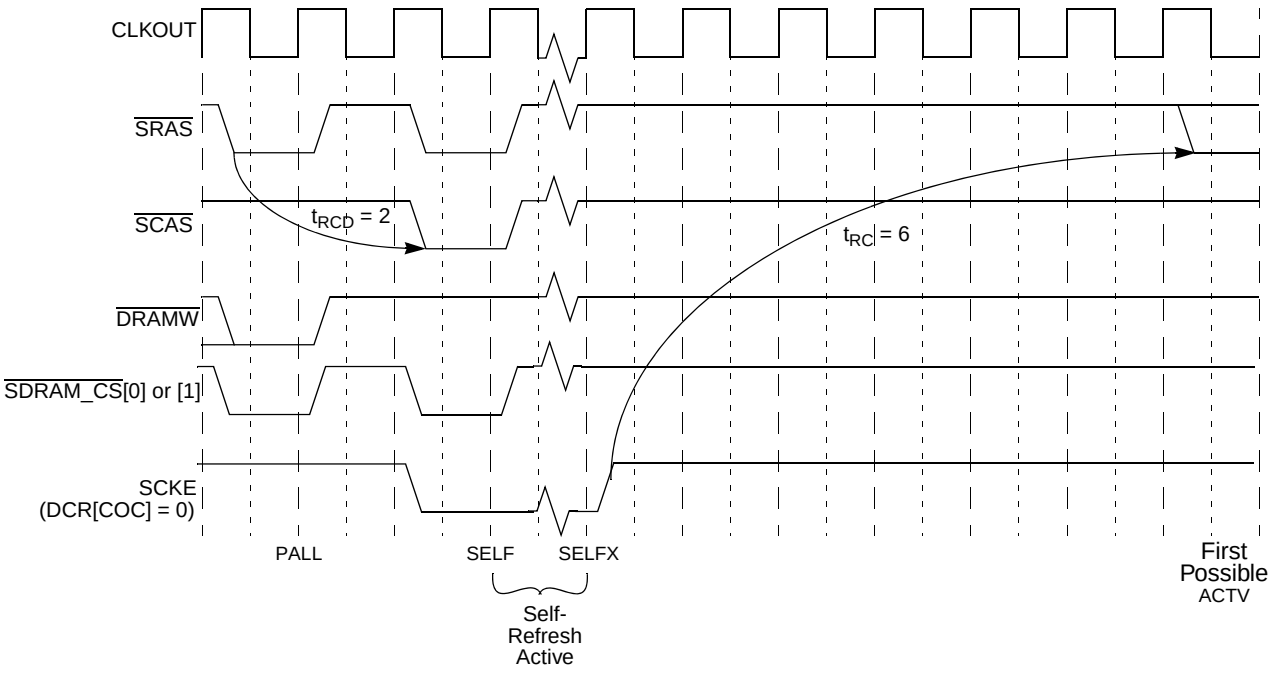


Figure 15-9. Self-Refresh Operation

15.2.4 Initialization Sequence

Synchronous DRAMs have a prescribed initialization sequence. The DRAM controller supports this sequence with the following procedure:

1. SDRAM control signals are reset to idle state. Wait the prescribed period after reset before any action is taken on the SDRAMs. This is normally around 100 µs.
2. Initialize the DCR, DACR, and DMR in their operational configuration. Do not yet enable PALL or REF commands.
3. Issue a PALL command to the SDRAMs by setting DACR[IP] and accessing a SDRAM location. Wait the time (determined by t_{RP}) before any other execution.
4. Enable refresh (set DACR[RE]) and wait for at least 8 refreshes to occur.
5. Before issuing the MRS command, determine if the DMR mask bits need to be modified to allow the MRS to execute properly
6. Issue the MRS command by setting DACR[IMRS] and accessing a location in the SDRAM. Note that mode register settings are driven on the SDRAM address bus, so care must be taken to change DMR[BAM] if the mode register configuration does not fall in the address range determined by the address mask bits. After the mode register is set, DMR mask bits can be restored to their desired configuration.

15.2.4.1 Mode Register Settings

It is possible to configure the operation of SDRAMs, namely their burst operation and CAS latency, through the SDRAM component's mode register. CAS latency is a function of the speed of the SDRAM and the bus clock of the DRAM controller. The DRAM controller operates at a CAS latency of 1, 2, or 3.

Although the DRAM controller supports bursting operations, it does not use the bursting features of the SDRAMs. Because the processor can burst operand sizes of 1, 2, 4, or 16 bytes long, the concept of a fixed burst length in the SDRAMs mode register becomes problematic. Therefore, the processor DRAM controller generates the burst cycles rather than the SDRAM device. Because the processor generates a new address and a READ or WRITE command for each transfer within the burst, the SDRAM mode register should be set either not to burst or to a burst length of one. This allows bursting to be controlled by the processor.

The SDRAM mode register is written by setting the associated block's DACR[IMRS]. First, the base address and mask registers must be set to the appropriate configuration to allow the mode register to be set. Note that improperly set DMR mask bits may prevent access to the mode register address. Thus, the user should determine the mapping of the mode register address to the processor address bits to find out if an access is blocked. If the DMR setting prohibits mode register access, the DMR should be reconfigured to enable the access and then set to its necessary configuration after the MRS command executes.

The associated CBM bits should also be initialized. After DACR[IMRS] is set, the next access to the SDRAM address space generates the MRS command to that SDRAM. The address of the access should be selected to place the correct mode information on the SDRAM address pins. The address is not multiplexed for the MRS command. The MRS access can be a read or write. The important thing is that the address output of that access needs the correct mode programming information on the correct address bits.

Figure 15-10 shows the MRS command, which occurs in the first clock of the bus cycle.

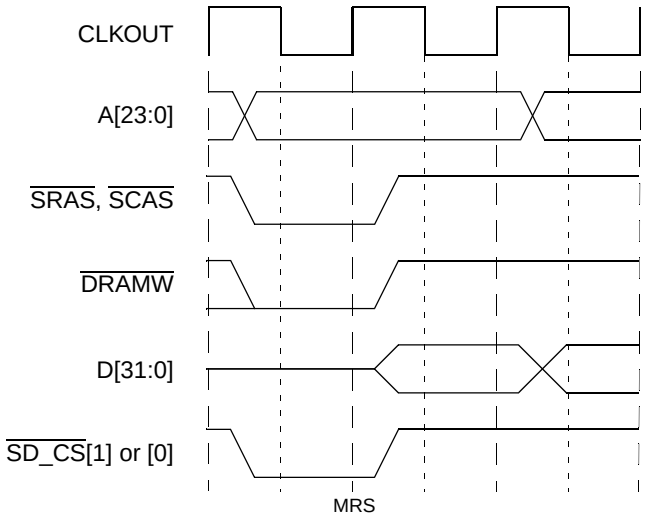


Figure 15-10. Mode Register Set (MRS) Command

15.3 SDRAM Example

This example interfaces a 512K x 32-bit x 4 bank SDRAM component to processor operating at 40 MHz. Table 15-25 lists design specifications for this example.

Table 15-25. SDRAM Example Specifications

Parameter	Specification
Speed grade (-8E)	40 MHz (25-ns period)
10 rows, 8 columns	
Two bank-select lines to access four internal banks	
ACTV-to-read/write delay (t_{RCD})	20 ns (min.)
Period between auto-refresh and ACTV command (t_{RC})	70 ns
ACTV command to precharge command (t_{RAS})	48 ns (min.)
Precharge command to ACTV command (t_{RP})	20 ns (min.)
Last data input to PALL command (t_{RWL})	1 bus clock (25 ns)
Auto-refresh period for 4096 rows (t_{REF})	64 mS

15.3.1 SDRAM Interface Configuration

To interface this component to the DRAM controller, use the connection table that corresponds to a 32-bit port size with 8 columns (Table 15-24). Two pins select one of four banks when the part is functional. Table 15-26 shows the proper hardware connections.

Table 15-26. SDRAM Hardware Connections

Processor Pins	A15	A14	A13	A12	A11	A10	A9	A17	A18	A19	A20	A21	A22
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10 = CMD	BA0	BA1

15.3.2 DCR Initialization

At power-up, the DCR has the following configuration if synchronous operation and SDRAM address multiplexing are desired.

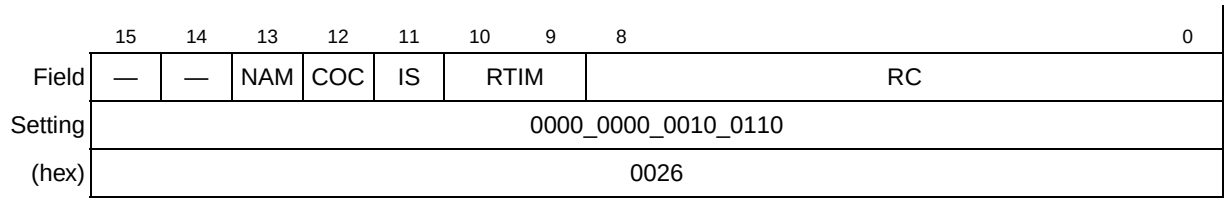


Figure 15-11. Initialization Values for DCR

This configuration results in a value of 0x0026 for DCR, as shown in Table 15-27.

Table 15-27. DCR Initialization Values

Bits	Name	Setting	Description
15	—	0	Reserved.
14	—	0	Reserved.
13	NAM	0	Indicating SDRAM controller multiplexes address lines internally
12	COC	0	SCKE is used as clock enable instead of command bit because user is not multiplexing address lines externally and requires external command feed.
11	IS	0	At power-up, allowing power self-refresh state is not appropriate because registers are being set up.
10–9	RTIM	00	Because t_{RC} value is 70 ns, indicating a 3-clock refresh-to-ACTV timing.
8–0	RC	0x26	Specification indicates auto-refresh period for 4096 rows to be 64 mS or refresh every 15.625 µs for each row, or 625 bus clocks at 40 MHz. Because DCR[RC] is incremented by 1 and multiplied by 16, RC = (625 bus clocks/16) -1 = 38.06 = 0x38

15.3.3 DACR Initialization

As shown in Figure 15-12, the SDRAM is programmed to access only the second 512-Kbyte block of each 1-Mbyte partition in the SDRAM (each 16 Mbytes). The starting address of the SDRAM is 0xFF88_0000. Continuous page mode feature is used.

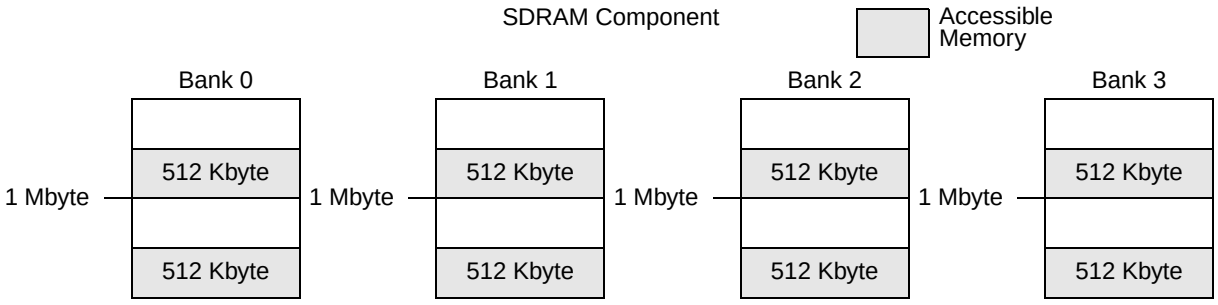


Figure 15-12. SDRAM Configuration

The DACRs should be programmed as shown in Figure 15-13.

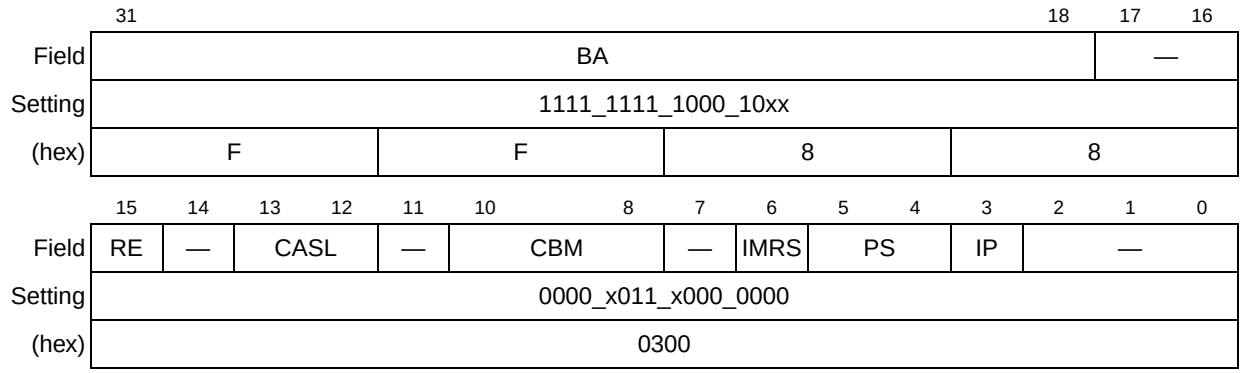


Figure 15-13. DACR Register Configuration

This configuration results in a value of DACR0 = 0xFF88_0300, as described in Table 15-28. DACR1 initialization is not needed because there is only one block. Subsequently, DACR1[RE,IMRS,IP] should be cleared; everything else is a don't care.

Table 15-28. DACR Initialization Values

Bits	Name	Setting	Description
31–18	BA	1111_1111_1000_10	Base address. So DACR0[31–16] = 0xFF88, placing the starting address of the SDRAM accessible memory at 0xFF88_0000.
17–16	—		Reserved. Don't care.
15	RE	0	Keeps auto-refresh disabled because registers are being set up at this time.
14	—		Reserved. Don't care.
13–12	CASL	00	Indicates a delay of data 1 cycle after $\overline{\text{SCAS}}$ is asserted
11	—		Reserved. Don't care.
10–8	CBM	011	Command bit is pin 20 and bank selects are 21 and up.
7	—		Reserved. Don't care.
6	IMRS	0	Indicates MRS command has not been initiated.
5–4	PS	00	32-bit port.

Table 15-28. DACR Initialization Values (continued)

Bits	Name	Setting	Description
3	IP	0	Indicates precharge has not been initiated.
2–0	—		Reserved. Don't care.

15.3.4 DMR Initialization

Again, in this example only the second 512-Kbyte block of each 1-Mbyte space is accessed in each bank. In addition, the SDRAM component is mapped only to readable and writable supervisor and user data. The DMRs have the following configuration.

	31 … 18	17	16
Field	BAM	—	
Setting	0000_0000_0111_01xx		
(hex)	0 074		

	15 … 9	8	7	6	5	4	3	2	1	0
Field	—	WP	—	C/I	AM	SC	SD	UC	UD	V
Setting	xxxx_xxx0_x111_0101									
(hex)	0075									

Figure 15-14. DMR0 Register

With this configuration, the DMR0 = 0x0074_0075, as described in Table 15-29.

Table 15-29. DMR0 Initialization Values

Bits	Name	Setting	Description
31–18	BAM		With bits 17 and 16 as don't cares, BAM = 0x0074, which leaves bank select bits and upper 512K select bits unmasked. Note that bits 22 and 21 are set because they are used as bank selects; bit 20 is set because it controls the 1-Mbyte boundary address.
17–16	—		Reserved. Don't care.
15–9	—		Reserved. Don't care.
8	WP	0	Allow reads and writes
7	—		Reserved. Don't care.
6	C/I	1	Disable CPU space access.
5	AM	1	Disable alternate master access.
4	SC	1	Disable supervisor code accesses.
3	SD	0	Enable supervisor data accesses.
2	UC	1	Disable user code accesses.
1	UD	0	Enable user data accesses.
0	V	1	Enable accesses.

15.3.5 Mode Register Initialization

When DACR[IMRS] is set, a bus cycle initializes the mode register. If the mode register setting is read on A[10:0] of the SDRAM on the first bus cycle, the bit settings on the corresponding processor address pins must be determined while being aware of masking requirements.

Table 15-30 lists the desired initialization setting:

Table 15-30. Mode Register Initialization

Processor Pins	SDRAM Pins	Mode Register Initialization	
A20	A10	Reserved	X
A19	A9	WB	0
A18	A8	Opmode	0
A17	A7	Opmode	0
A9	A6	CASL	0
A10	A5	CASL	0
A11	A4	CASL	1
A12	A3	BT	0
A13	A2	BL	0
A14	A1	BL	0
A15	A0	BL	0

Next, this information is mapped to an address to determine the hexadecimal value.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field																
Setting	xxxx_xxxx_xxxx_000x															
(hex)	0000															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field																V
Setting	0000_100x_xxxx_xxxx															
(hex)	0800															

Table 15-31. Mode Register Mapping to A[31:0]

Although A[31:20] corresponds to the address programmed in DACR0, according to how DACR0 and DMR0 are initialized, bit 19 must be set to hit in the SDRAM. Thus, before the mode register bit is set, DMR0[19] must be set to enable masking.

15.3.6 Initialization Code

The following assembly code initializes the SDRAM example.

Power-Up Sequence:

```
move.w   #0x0026, d0//Initialize DCR
move.w   d0, DCR
move.l   #0xFF880300, d0 //Initialize DACR0
move.l   d0, DACR0
move.l   #0x00740075, d0//Initialize DMR0
move.l   d0, DMR0
```

Precharge Sequence:

```
move.l   #0xFF880308, d0//Set DACR0[IP]
move.l   d0, DACR0
move.l   #0xBEADDEED, d0//Write and value to memory location to init. precharge
move.l   d0, 0xFF880000
```

Refresh Sequence:

```
move.l   #0xFF888300, d0//Enable refresh bit in DACR0
move.l   d0, DACR0
```

Mode Register Initialization Sequence:

```
move.l   #0x00600075, d0//Mask bit 19 of address
move.l   d0, DMR0
move.l   #0xFF888340, d0//Enable DACR0[IMRS]; DACR0[RE] remains set
move.l   d0, DACR0
move.l   #0x00000000, d0//Access SDRAM address to initialize mode register
move.l   d0, 0xFF800800
```

Chapter 16
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.

16.1 Overview

The DMA controller module provides an efficient way to move blocks of data with minimal processor interaction. The DMA module, shown in Figure 16-1, 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 UART, SDRAM controller, and GPIOs.

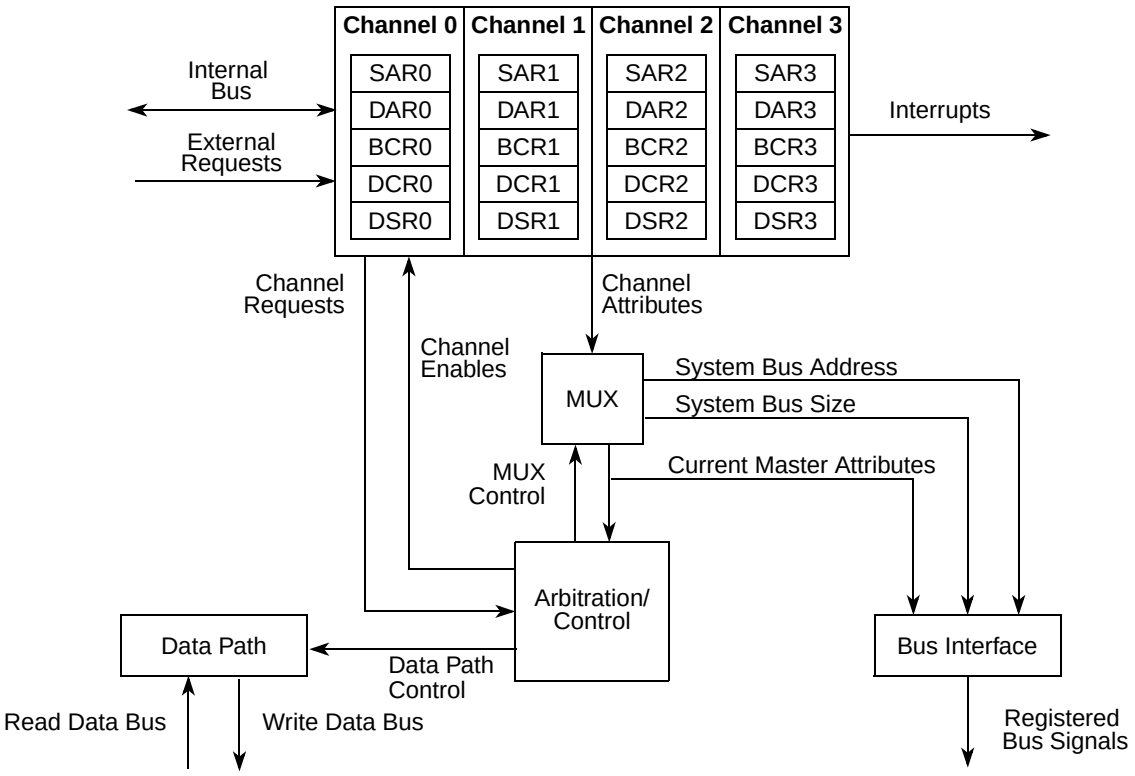


Figure 16-1. DMA Signal Diagram

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. For details on the connections associated with DMA request inputs, see Section 16.2, “DMA Request Control (DMAREQC).”

16.1.1 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

16.2 DMA Request Control (DMAREQC)

The DMAREQC register provides a software-controlled connection matrix for DMA requests. It logically routes DMA requests from the DMA timers and UARTs to the four channels of the DMA controller. Writing to this register determines the exact routing of the DMA request to the four channels of the DMA modules. If DCR*n*[EEXT] is set and the channel is idle, the assertion of the appropriate DREQ*n* activates channel *n*.

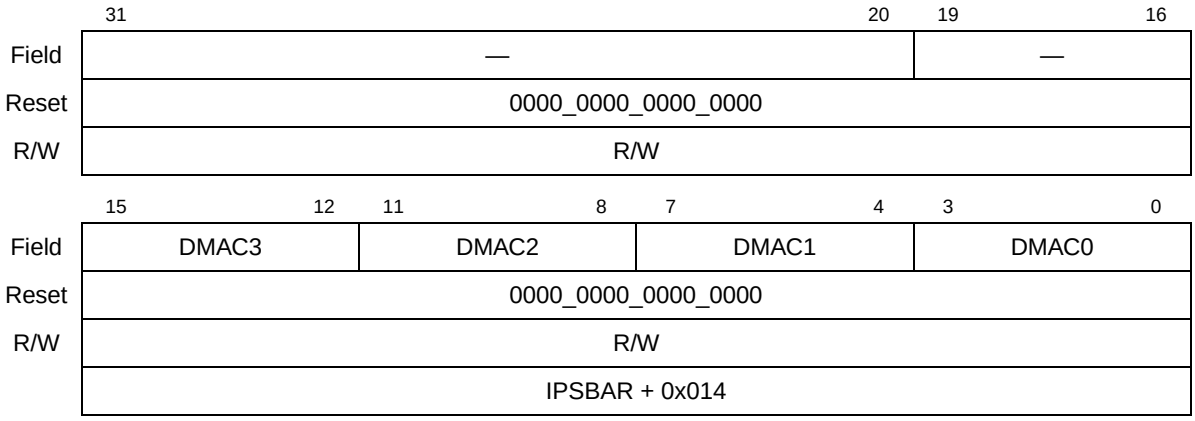


Figure 16-2. DMA Request Control Register (DMAREQC)

Table 16-1. DMAREQC Field Description

Bits	Name	Description

Table 16-1. DMAREQC Field Description (continued)

31–16	—	Reserved. Should be cleared.
15–0	DMAC*n*	DMA Channel *n*. Each four bit field defines the logical connection between the DMA requestors and that DMA channel. There are seven possible requesters (4 DMA Timers and 3 UARTs). Any request can be routed to any of the DMA channels. Effectively, the DMAREQC provides a software-controlled routing matrix of the 7 DMA request signals to the 4 channels of the DMA module. DMAC3 controls DMA channel 3. DMAC2 controls DMA channel 2. DMAC1 controls DMA channel 1. DMAC0 controls DMA channel 0. 1000 UART0. 1001 UART1. 1010 UART2. 0100 DMA Timer 0. 0101 DMA Timer 1. 0110 DMA Timer 2. 0111 DMA Timer 3. All other values are reserved and will not generate a DMA request.

16.3 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 16.5.1, “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 16-3 for more information.

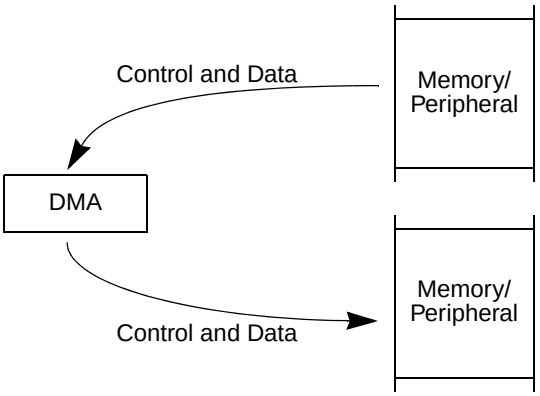


Figure 16-3. Dual-Address Transfer

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 16.4.5, “DMA Status Registers (DSR0–DSR3).”

16.4 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 16-2 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 16-2. 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. 16-5]			
	0x104	Destination address register 0 (DAR0) [p. 16-6]			
	0x108	DMA control register 0 (DCR0) [p. 16-7]			
	0x10C	Byte count register 0 (BCR24BIT = 0) [1]		Reserved	
	0x10C	Reserved	Byte count register 0 (BCR24BIT = 1) [1] (BCR0) [p. 16-7]		
	0x110	DMA status register 0 (DSR0) [p. 16-10]	Reserved		
1	0x140	Source address register 1 (SAR1) [p. 16-5]			
	0x144	Destination address register 1 (DAR1) [p. 16-6]			
	0x148	DMA control register 1 (DCR1) [p. 16-7]			
	0x14C	Byte count register 1 (BCR24BIT = 0) [1]		Reserved	
	0x14C	Reserved	Byte count register 1 (BCR24BIT = 1) [1] (BCR1) [p. 16-7]		
	0x150	DMA status register 1 (DSR1) [p. 16-10]	Reserved		
2	0x180	Source address register 2 (SAR2) [p. 16-5]			
	0x184	Destination address register 2 (DAR2) [p. 16-6]			
	0x188	DMA control register 2 (DCR2) [p. 16-7]			
	0x18C	Byte count register 2 (BCR24BIT = 0) [1]		Reserved	
	0x18C	Reserved	Byte count register 2 (BCR24BIT = 1) [1] (BCR2) [p. 16-7]		
	0x190	DMA status register 2 (DSR2) [p. 16-10]	Reserved		
3	0x1C0	Source address register 3 (SAR3) [p. 16-5]			
	0x1C4	Destination address register 3 (DAR3) [p. 16-6]			
	0x1C8	DMA control register 3 (DCR3) [p. 16-7]			
	0x1CC	Byte count register 3 (BCR24BIT = 0)[1]		Reserved	
	0x1CC	Reserved	Byte count register 3 (BCR24BIT = 1)[1] (BCR3) [p. 16-7]		
	0x1D0	DMA status register 3 (DSR3) [p. 16-10]	Reserved		

[1] The DMA module originally supported a left-justified 16-bit byte count register (BCR). This function was later reimplemented as a right-justified 24-bit BCR. The operation of the DMA and the interpretation of the BCR is controlled by the MPARK[BCR24BIT]. See Section 8.5.3, “Bus Master Park Register (MPARK)" for more details.

16.4.1 Source Address Registers (SAR0–SAR3)

SAR*n*, shown in Figure 16-4, 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 16-4. 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.

NOTE

Flash accesses (reads/writes) by a bus master other than the core (DMA controller or Fast Ethernet 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 Flash location when IPSBAR is still at its default location of 0x4000_0000, the source register would be loaded with 0x4400_0000. Backdoor Flash read accesses can be made with the bus master, but it takes two cycles longer than a direct read of the Flash when using the FLASHBAR address.

16.4.2 Destination Address Registers (DAR0–DAR3)

DAR*n*, shown in Figure 16-5, 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 16-5. Destination Address Registers (DAR*n*)

NOTE

The DMA does not maintain coherency with the cache. Therefore, DMAs should not transfer data to cacheable memory unless software is used to maintain the cache coherency.

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.

16.4.3 Byte Count Registers (BCR0–BCR3)

BCR*n*, shown in Figure 16-6 and Figure 16-7, hold 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 16-6 shows BCR*n* for BCR24BIT = 1.

	31–24	23–0
Field	—	BCR
Reset	—	0000_0000_0000_0000_0000_0000
R/W	R/W	
Address	IPSBAR + 0x10C, 0x14C, 0x18C, 0x1CC	

Figure 16-6. Byte Count Registers (BCR*n*)—BCR24BIT = 1

Figure 16-7 shows BCR*n* for BCR24BIT = 0.

	15–0
Field	BCR
Reset	0000_0000_0000_0000
R/W	R/W
Address	IPSBAR + 0x10C, 0x14C, 0x18C, 0x1CC

Figure 16-7. Byte Count Registers (BCR*n*)—BCR24BIT = 0

DSR*n*[DONE], shown in Figure 16-9, 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 16.4.5, “DMA Status Registers (DSR0–DSR3).”

16.4.4 DMA Control Registers (DCR0–DCR3)

DCR*n*, shown in Figure 16-8, 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.

	31	30	29	28	27 25	24	23	22	21 20	19	18 17	16
Field	INT	EEXT	CS	AA	BWC	—	—	SINC	SSIZE	DINC	DSIZE	START
Reset	0000_0000_0000_0000											
R/W	R/W											

	15	14 0
Field	AT [1]	—
Reset	0	N/A
R/W	R/W	
Address	IPSBAR + 0x108, 0x148, 0x188, 0x1C8	

Figure 16-8. DMA Control Registers (DCR*n*)

[1] Available only if BCR24BIT = 1, otherwise reserved.

Table 16-3 describes DCR*n* fields.

Table 16-3. 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 16.5.4.1, “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 16-3. DCR*n* Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>27–25</td><td>BWC</td><td>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.
<table>
<tr><th>Encoding</th><th>BCR24BIT = 0</th><th>BCR24BIT = 1</th></tr>
<tr><td>000</td><td colspan="2">DMA has priority and does not negate its request until transfer completes.</td></tr>
<tr><td>001</td><td>512</td><td>16384</td></tr>
<tr><td>010</td><td>1024</td><td>32768</td></tr>
<tr><td>011</td><td>2048</td><td>65536</td></tr>
<tr><td>100</td><td>4096</td><td>131072</td></tr>
<tr><td>101</td><td>8192</td><td>262144</td></tr>
<tr><td>110</td><td>16384</td><td>524288</td></tr>
<tr><td>111</td><td>32768</td><td>1048576</td></tr>
</table>
</td></tr>
<tr><td>24–23</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>22</td><td>SINC</td><td>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.</td></tr>
<tr><td>21–20</td><td>SSIZE</td><td>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)</td></tr>
<tr><td>19</td><td>DINC</td><td>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.</td></tr>
<tr><td>18–17</td><td>DSIZE</td><td>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)</td></tr>
<tr><td>16</td><td>START</td><td>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.</td></tr>
</table>

Table 16-3. DCR*n* Field Descriptions (continued)

Bits	Name	Description
15	AT	AT is available only if MPARK[BCR24BIT] = 1. DMA acknowledge type. Controls whether acknowledge information is provided for the entire transfer or only the final transfer. 0 Entire transfer. DMA acknowledge information is displayed anytime the channel is selected as the result of an external request. 1 Final transfer (when BCR reaches zero). For dual-address transfer, the acknowledge information is displayed for both the read and write cycles.
14–0	—	Reserved, should be cleared.

16.4.5 DMA Status Registers (DSR0–DSR3)

In response to an event, the DMA controller writes to the appropriate DSR*n* bit, Figure 16-9. 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 + 0x110, 0x150, 0x190, 0x1D0							

Figure 16-9. DMA Status Registers (DSR*n*)

Table 16-4 describes DSR*n* fields.

Table 16-4. 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.

Table 16-4. DSR*n* Field Descriptions (continued)

Bits	Name	Description
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.

16.5 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 DCR*n*[SSIZE,DSIZE] are consistent with the source and destination addresses. If they are not consistent, the configuration error bit, DSR*n*[CE], is set. If misalignment is detected, no transfer occurs, DSR*n*[CE] is set, and, depending on the DCR configuration, an interrupt event is issued. Note that if the auto-align bit, DCR*n*[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 DCR*n*[SSIZE] and DCR*n*[DSIZE]. See Section 16.4.4, “DMA Control Registers (DCR0–DCR3).”

Source and destination address registers (SAR*n* and DAR*n*) can be programmed in the DCR*n* to increment at the completion of a successful transfer.

16.5.1 Transfer Requests (Cycle-Steal and Continuous Modes)

The DMA channel supports internal and external requests. A request is issued by setting DCR*n*[START] or by asserting DREQ*n*. Setting DCR*n*[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 (DCR*n*[CS] = 1)—Only one complete transfer from source to destination occurs for each request. If DCR*n*[EEXT] is set, a request can be either internal or external. An internal request is selected by setting DCR*n*[START]. An external request is initiated by asserting DREQ*n* while DCR*n*[EEXT] is set. Note that multiple transfers will occur if DREQ*n* is continuously asserted.
- Continuous mode (DCR*n*[CS] = 0)—After an internal or external request, the DMA continuously transfers data until BCR*n* reaches zero or a multiple of DCR*n*[BWC] or until DSR*n*[DONE] is set. If BCR*n* is a multiple of BWC, the DMA request signal is negated until the bus cycle terminates to allow the internal arbiter to switch masters. DCR*n*[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 BCR*n* 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.

16.5.2 Data Transfer Modes

Each channel supports dual-address transfers, described in the next section.

16.5.2.1 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, DSR*n*[REQ] is set.

- Dual-address read—The DMA controller drives the SAR*n* value onto the internal address bus. If DCR*n*[SINC] is set, the SAR*n* increments by the appropriate number of bytes upon a successful read cycle. When the appropriate number of read cycles complete (multiple reads if the destination size is larger than the source), the DMA initiates the write portion of the transfer.

  If a termination error occurs, DSR*n*[BES,DONE] are set and DMA transactions stop.
- Dual-address write—The DMA controller drives the DAR*n* value onto the address bus. If DCR*n*[DINC] is set, DAR*n* increments by the appropriate number of bytes at the completion of a successful write cycle. BCR*n* decrements by the appropriate number of bytes. DSR*n*[DONE] is set when BCR*n* reaches zero. If the BCR*n* is greater than zero, another read/write transfer is initiated. If the BCR*n* is a multiple of DCR*n*[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, DSR*n*[BES,DONE] are set and DMA transactions stop.

16.5.3 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.

16.5.3.1 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 DCR*n*[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 DCR*n*[BWC] bits.

16.5.3.2 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 DCR*n*[BWC] value of sequential channels are equal, the channels are prioritized in ascending order.

The SAR*n* 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 DAR*n* should contain the destination (write) address. If the transfer is from a peripheral device to memory, or from memory to memory, the DAR*n* is loaded with the starting address of the data block to be written. If the transfer is from memory to a peripheral device, DAR*n* is loaded with the address of the peripheral data register. This address can be any aligned byte address.

SAR*n* and DAR*n* change after each cycle depending on DCR*n*[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.

BCR*n*[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. DSR*n*[DONE] must be cleared for channel startup.

As soon as the channel has been initialized, it is started by writing a one to DCR*n*[START] or asserting DREQ*n*, depending on the status of DCR*n*[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.

16.5.4 Data Transfer

This section describes auto-alignment and bandwidth control for DMA transfers.

16.5.4.1 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.

16.5.4.2 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 Section 8.5.3, “Bus Master Park Register (MPARK).”

16.5.5 Termination

An unsuccessful transfer can terminate for one of the following reasons:

- Error conditions—When the processor encounters a read or write cycle that terminates with an error condition, DSR*n*[BES] is set for a read and DSR*n*[BED] is set for a write before the transfer is halted. If the error occurred in a write cycle, data in the internal holding register is lost.
- Interrupts—If DCR*n*[INT] is set, the DMA drives the appropriate internal interrupt signal. The processor can read DSR*n* to determine whether the transfer terminated successfully or with an error. DSR*n*[DONE] is then written with a one to clear the interrupt and the DONE and error bits.

Chapter 17
Fast Ethernet Controller (FEC)

17.1 Introduction

This chapter provides a feature-set overview, a functional block diagram, and transceiver connection information for the 10 and 100 Mbps MII (media independent interface), as well as the 7-wire serial interface. Additionally, detailed descriptions of operation and the programming model are included.

NOTE

The MCF5214 and MCF5216 do NOT contain an FEC module.

17.1.1 Overview

The Ethernet media access controller (MAC) supports 10 and 100 Mbps Ethernet/IEEE 802.3 networks. An external transceiver interface and transceiver function are required to complete the interface to the media. The FEC supports three different standard MAC-PHY (physical) interfaces for connection to an external Ethernet transceiver. The FEC supports the 10/100 Mbps MII and the 10 Mbps-only 7-wire interface.

NOTE

The GPIO module must be configured to enable the peripheral function of the appropriate pins (refer to Chapter 26, “General Purpose I/O Module”) prior to configuring the FEC.

17.1.2 Block Diagram

Figure 17-1 shows the block diagram of the FEC. The FEC is implemented with a combination of hardware and microcode. The off-chip (Ethernet) interfaces are compliant with industry and IEEE 802.3 standards.

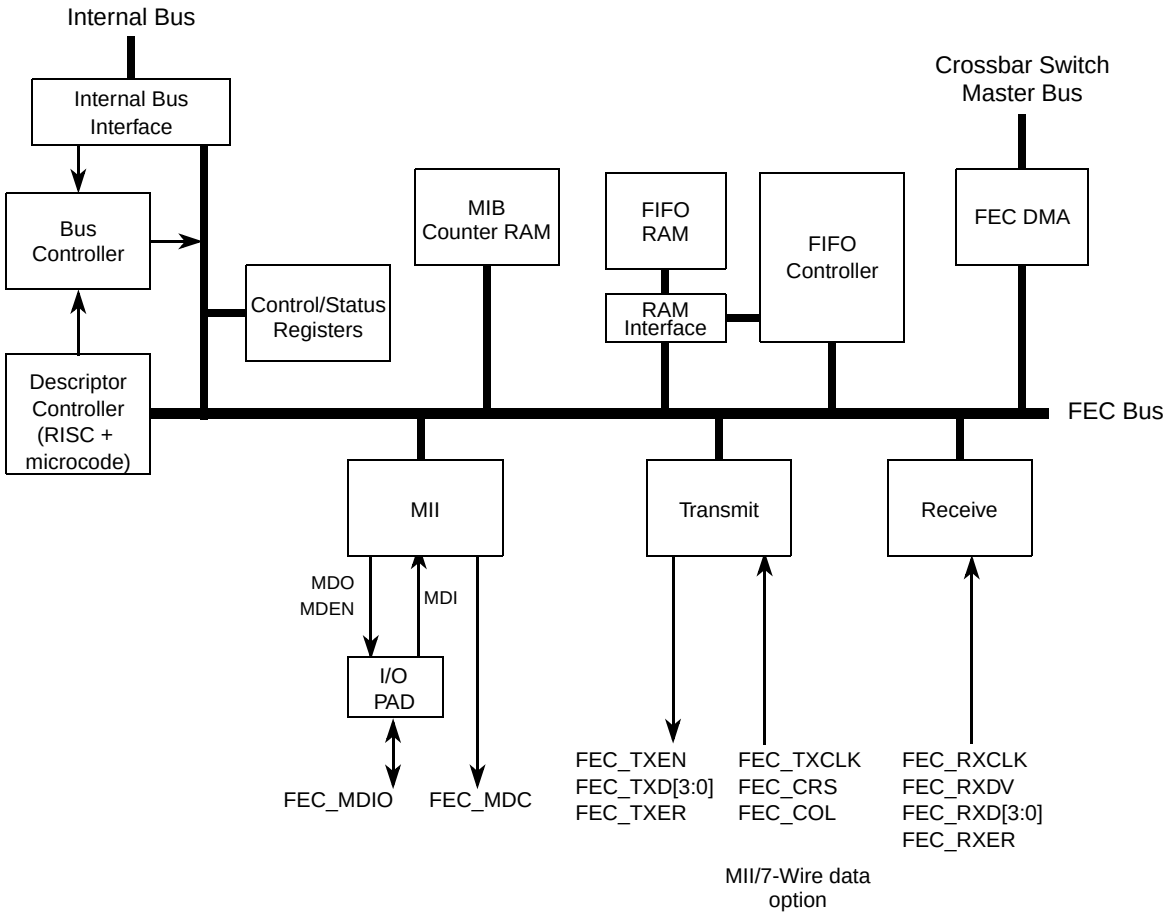


Figure 17-1. FEC Block Diagram

The descriptor controller is a RISC-based controller providing these functions in the FEC:

- Initialization (those internal registers not initialized by you or hardware)
- High level control of the DMA channels (initiating DMA transfers)
- Interpreting buffer descriptors
- Address recognition for receive frames
- Random number generation for transmit collision backoff timer

NOTE

DMA references in this section refer to the FEC's DMA engine. This DMA engine transfers FEC data only and is not related to the eDMA controller described in Chapter 16, "DMA Controller Module," nor to the DMA timers described in Chapter 21, "DMA Timers (DTIM0–DTIM3)."

The RAM is the focal point of all data flow in the Fast Ethernet controller and divides into transmit and receive FIFOs. The FIFO boundaries are programmable using the FRSR register. User data flows to/from the DMA block from/to the receive/transmit FIFOs. Transmit data flows from the transmit FIFO into the transmit block, and receive data flows from the receive block into the receive FIFO.

You control the FEC by writing into control registers located in each block. The CSR (control and status registers) block provides global control (Ethernet reset and enable) and interrupt managing registers.

The MII block provides a serial channel for control/status communication with the external physical layer device (transceiver). This serial channel consists of the FEC_MDC (management data clock) and FEC_MDIO (management data input/output) lines of the MII interface.

The FEC DMA block (not to be confused with the device's eDMA controller) provides multiple channels allowing transmit data, transmit descriptor, receive data and receive descriptor accesses to run independently.

The transmit and receive blocks provide the Ethernet MAC functionality (with some assist from microcode).

The message information block (MIB) maintains counters for a variety of network events and statistics. It is not necessary for operation of the FEC, but provides valuable counters for network management. The counters supported are the RMON (RFC 1757) Ethernet Statistics group and some of the IEEE 802.3 counters. See Section 17.4.1, "MIB Block Counters Memory Map," for more information.

17.1.3 Features

The FEC incorporates the following features:

- Support for three different Ethernet physical interfaces:
  - 100-Mbps IEEE 802.3 MII
  - 10-Mbps IEEE 802.3 MII
  - 10-Mbps 7-wire interface (industry standard)
- IEEE 802.3 full duplex flow control
- Programmable max frame length supports IEEE 802.1 VLAN tags and priority
- Support for full-duplex operation (200 Mbps throughput) with a minimum internal bus clock rate of 50 MHz
- Support for half-duplex operation (100 Mbps throughput) with a minimum internal bus clock rate of 50 MHz
- Retransmission from transmit FIFO following a collision (no processor bus utilization)
- Automatic internal flushing of the receive FIFO for runts (collision fragments) and address recognition rejects (no processor bus utilization)
- Address recognition
  - Frames with broadcast address may be always accepted or always rejected
  - Exact match for single 48-bit individual (unicast) address
  - Hash (64-bit hash) check of individual (unicast) addresses
  - Hash (64-bit hash) check of group (multicast) addresses
  - Promiscuous mode

17.2 Modes of Operation

The primary operational modes are described in this section.

17.2.1 Full and Half Duplex Operation

Full duplex mode is for use on point-to-point links between switches or end node to switch. Half duplex mode works in connections between an end node and a repeater or between repeaters. TCR[FDEN] controls duplex mode selection.

When configured for full duplex mode, flow control may be enabled. Refer to the TCR[RFC_PAUSE,TFC_PAUSE] bits, the RCR[FCE] bit, and Section 17.5.11, “Full Duplex Flow Control,” for more details.

17.2.2 Interface Options

The following interface options are supported. A detailed discussion of the interface configurations is provided in Section 17.5.6, “Network Interface Options.”

17.2.2.1 10 Mbps and 100 Mbps MII Interface

The IEEE 802.3 standard defines the media independent interface (MII) for 10/100 Mbps operation. The MAC-PHY interface may be configured to operate in MII mode by setting RCR[MII_MODE].

FEC_TXCLK and FEC_RXCLK pins driven by the external transceiver determine the operation speed. The transceiver auto-negotiates the speed or software controls it via the serial management interface (FEC_MDC/FEC_MDIO pins) to the transceiver. Refer to the MMFR and MSCR register descriptions, as well as the section on the MII, for a description of how to read and write registers in the transceiver via this interface.

17.2.2.2 10 Mpbs 7-Wire Interface Operation

The FEC supports 7-wire interface used by many 10 Mbps Ethernet transceivers. The RCR[MII_MODE] bit controls this functionality. If this bit is cleared, MII mode is disabled and the 10 Mbps 7-wire mode is enabled.

17.2.3 Address Recognition Options

The address options supported are promiscuous, broadcast reject, individual address (hash or exact match), and multicast hash match. Address recognition options are discussed in detail in Section 17.5.9, “Ethernet Address Recognition.”

17.2.4 Internal Loopback

Internal loopback mode is selected via RCR[LOOP]. Loopback mode is discussed in detail in Section 17.5.14, “MII Internal and External Loopback.”

17.3 External Signal Description

Table 17-1 describes the various FEC signals, as well as indicating which signals work in available modes.

Table 17-1. FEC Signal Descriptions

Signal Name	MII	7-wire	Description
FEC_COL	X	X	Asserted upon detection of a collision and remains asserted while the collision persists. This signal is not defined for full-duplex mode.
FEC_CRS	X	—	When asserted, indicates that transmit or receive medium is not idle.
FEC_MDC	X	—	Output clock which provides a timing reference to the PHY for data transfers on the FEC_MDIO signal.
FEC_MDIO	X	—	Transfers control information between the external PHY and the media-access controller. Data is synchronous to FEC_MDC. This signal is an input after reset. When the FEC is operated in 10Mbps 7-wire interface mode, this signal should be connected to VSS.
FEC_RXCLK	X	X	Provides a timing reference for FEC_RXDV, FEC_RXD[3:0], and FEC_RXER.
FEC_RXDV	X	X	Asserting the FEC_RXDV input indicates that the PHY has valid nibbles present on the MII. FEC_RXDV should remain asserted from the first recovered nibble of the frame through to the last nibble. Assertion of FEC_RXDV must start no later than the SFD and exclude any EOF.
FEC_RXD0	X	X	This pin contains the Ethernet input data transferred from the PHY to the media-access controller when FEC_RXDV is asserted.
FEC_RXD1	X	—	This pin contains the Ethernet input data transferred from the PHY to the media access controller when FEC_RXDV is asserted.
FEC_RXD[3:2]	X	—	These pins contain the Ethernet input data transferred from the PHY to the media access controller when FEC_RXDV is asserted.
FEC_RXER	X	—	When asserted with FEC_RXDV, indicates that the PHY has detected an error in the current frame. When FEC_RXDV is not asserted FEC_RXER has no effect.
FEC_TXCLK	X	X	Input clock which provides a timing reference for FEC_TXEN, FEC_TXD[3:0] and FEC_TXER.
FEC_TXD0	X	X	The serial output Ethernet data and is only valid during the assertion of FEC_TXEN.
FEC_TXD1	X	—	This pin contains the serial output Ethernet data and is valid only during assertion of FEC_TXEN.
FEC_TXD[3:2]	X	—	These pins contain the serial output Ethernet data and are valid only during assertion of FEC_TXEN.
FEC_TXEN	X	X	Indicates when valid nibbles are present on the MII. This signal is asserted with the first nibble of a preamble and is negated before the first FEC_TXCLK following the final nibble of the frame.
FEC_TXER	X	—	When asserted for one or more clock cycles while FEC_TXEN is also asserted, the PHY sends one or more illegal symbols. FEC_TXER has no effect at 10 Mbps or when FEC_TXEN is negated.

17.4 Memory Map/Register Definition

The FEC is programmed by a combination of control/status registers (CSRs) and buffer descriptors. The CSRs control operation modes and extract global status information. The descriptors pass data buffers and related buffer information between the hardware and software.

Each FEC implementation requires a 1-Kbyte memory map space, which is divided into two sections of 512 bytes each for:

- Control/status registers
- Event/statistic counters held in the MIB block

Table 17-2 defines the top level memory map.

Table 17-2. Module Memory Map

Address	Function
IPSBAR + 0x1000 – 11FF	Control/Status Registers
IPSBAR + 0x1200 – 12FF	MIB Block Counters

Table 17-3 shows the FEC register memory map.

Table 17-3. FEC Register Memory Map

IPSBAR Offset	Register	Width (bits)	Access	Reset Value	Section/Page
0x1004	Interrupt Event Register (EIR)	32	R/W	0x0000_0000	17.4.2/17-9
0x1008	Interrupt Mask Register (EIMR)	32	R/W	0x0000_0000	17.4.3/17-10
0x1010	Receive Descriptor Active Register (RDAR)	32	R/W	0x0000_0000	17.4.4/17-11
0x1014	Transmit Descriptor Active Register (TDAR)	32	R/W	0x0000_0000	17.4.5/17-12
0x1024	Ethernet Control Register (ECR)	32	R/W	0xF000_0000	17.4.6/17-12
0x1040	MII Management Frame Register (MMFR)	32	R/W	Undefined	17.4.7/17-13
0x1044	MII Speed Control Register (MSCR)	32	R/W	0x0000_0000	17.4.8/17-15
0x1064	MIB Control/Status Register (MIBC)	32	R/W	0x0000_0000	17.4.9/17-16
0x1084	Receive Control Register (RCR)	32	R/W	0x05EE_0001	17.4.10/17-16
0x10C4	Transmit Control Register (TCR)	32	R/W	0x0000_0000	17.4.11/17-17
0x10E4	Physical Address Low Register (PALR)	32	R/W	Undefined	17.4.12/17-18
0x10E8	Physical Address High Register (PAUR)	32	R/W	See Section	17.4.13/17-19
0x10EC	Opcode/Pause Duration (OPD)	32	R/W	See Section	17.4.14/17-19
0x1118	Descriptor Individual Upper Address Register (IAUR)	32	R/W	Undefined	17.4.15/17-20
0x111C	Descriptor Individual Lower Address Register (IALR)	32	R/W	Undefined	17.4.16/17-20
0x1120	Descriptor Group Upper Address Register (GAUR)	32	R/W	Undefined	17.4.17/17-21
0x1124	Descriptor Group Lower Address Register (GALR)	32	R/W	Undefined	17.4.18/17-21
0x1144	Transmit FIFO Watermark (TFWR)	32	R/W	0x0000_0000	17.4.19/17-22
0x114C	FIFO Receive Bound Register (FRBR)	32	R	0x0000_0600	17.4.20/17-22
0x1150	FIFO Receive FIFO Start Register (FRSR)	32	R	0x0000_0500	17.4.21/17-23
0x1180	Pointer to Receive Descriptor Ring (ERDSR)	32	R/W	Undefined	17.4.22/17-23
0x1184	Pointer to Transmit Descriptor Ring (ETDSR)	32	R/W	Undefined	17.4.23/17-24
0x1188	Maximum Receive Buffer Size (EMRBR)	32	R/W	Undefined	17.4.24/17-24

17.4.1 MIB Block Counters Memory Map

The MIB counters memory map (Table 17-4) defines the locations in the MIB RAM space where hardware-maintained counters reside. The counters are divided into two groups:

- RMON counters include the Ethernet statistics counters defined in RFC 1757
- A counter is included to count truncated frames since only frame lengths up to 2047 bytes are supported

The transmit and receive RMON counters are independent, which ensures accurate network statistics when operating in full duplex mode.

The included IEEE counters support the mandatory and recommended counter packages defined in Section 5 of ANSI/IEEE Std. 802.3 (1998 edition). The FEC supports IEEE Basic Package objects, but these do not require counters in the MIB block. In addition, some of the recommended package objects supported do not require MIB counters. Counters for transmit and receive full duplex flow control frames are also included.

Table 17-4. MIB Counters Memory Map

IPSBAR Offset	Register
0x1200	Count of frames not counted correctly (RMON_T_DROP)
0x1204	RMON Tx packet count (RMON_T_PACKETS)
0x1208	RMON Tx broadcast packets (RMON_T_BC_PKT)
0x120C	RMON Tx multicast packets (RMON_T_MC_PKT)
0x1210	RMON Tx packets with CRC/align error (RMON_T_CRC_ALIGN)
0x1214	RMON Tx packets < 64 bytes, good CRC (RMON_T_UNDERSIZE)
0x1218	RMON Tx packets > MAX_FL bytes, good CRC (RMON_T_OVERSIZE)
0x121C	RMON Tx packets < 64 bytes, bad CRC (RMON_T_FRAG)
0x1220	RMON Tx packets > MAX_FL bytes, bad CRC (RMON_T_JAB)
0x1224	RMON Tx collision count (RMON_T_COL)
0x1228	RMON Tx 64 byte packets (RMON_T_P64)
0x122C	RMON Tx 65 to 127 byte packets (RMON_T_P65TO127)
0x1230	RMON Tx 128 to 255 byte packets (RMON_T_P128TO255)
0x1234	RMON Tx 256 to 511 byte packets (RMON_T_P256TO511)
0x1238	RMON Tx 512 to 1023 byte packets (RMON_T_P512TO1023)
0x123C	RMON Tx 1024 to 2047 byte packets (RMON_T_P1024TO2047)
0x1240	RMON Tx packets with > 2048 bytes (RMON_T_P_GTE2048)
0x1244	RMON Tx Octets (RMON_T_OCTETS)
0x1248	Count of transmitted frames not counted correctly (IEEE_T_DROP)
0x124C	Frames transmitted OK (IEEE_T_FRAME_OK)
0x1250	Frames transmitted with single collision (IEEE_T_1COL)

Table 17-4. MIB Counters Memory Map (continued)

IPSBAR Offset	Register
0x1254	Frames transmitted with multiple collisions (IEEE_T_MCOL)
0x1258	Frames transmitted after deferral delay (IEEE_T_DEF)
0x125C	Frames transmitted with late collision (IEEE_T_LCOL)
0x1260	Frames transmitted with excessive collisions (IEEE_T_EXCOL)
0x1264	Frames transmitted with Tx FIFO underrun (IEEE_T_MACERR)
0x1268	Frames transmitted with carrier sense error (IEEE_T_CSERR)
0x126C	Frames transmitted with SQE error (IEEE_T_SQE)
0x1270	Flow control pause frames transmitted (IEEE_T_FDXFC)
0x1274	Octet count for frames transmitted without error (IEEE_T_OCTETS_OK)
0x1280	Count of received frames not counted correctly (RMON_R_DROP)
0x1284	RMON Rx packet count (RMON_R_PACKETS)
0x1288	RMON Rx broadcast packets (RMON_R_BC_PKT)
0x128C	RMON Rx multicast packets (RMON_R_MC_PKT)
0x1290	RMON Rx packets with CRC/Align error (RMON_R_CRC_ALIGN)
0x1294	RMON Rx packets < 64 bytes, good CRC (RMON_R_UNDERSIZE)
0x1298	RMON Rx packets > MAX_FL bytes, good CRC (RMON_R_OVERSIZE)
0x129C	RMON Rx packets < 64 bytes, bad CRC (RMON_R_FRAG)
0x12A0	RMON Rx packets > MAX_FL bytes, bad CRC (RMON_R_JAB)
0x12A4	Reserved (RMON_R_RESVD_0)
0x12A8	RMON Rx 64 byte packets (RMON_R_P64)
0x12AC	RMON Rx 65 to 127 byte packets (RMON_R_P65TO127)
0x12B0	RMON Rx 128 to 255 byte packets (RMON_R_P128TO255)
0x12B4	RMON Rx 256 to 511 byte packets (RMON_R_P256TO511)
0x12B8	RMON Rx 512 to 1023 byte packets (RMON_R_P512TO1023)
0x12BC	RMON Rx 1024 to 2047 byte packets (RMON_R_P1024TO2047)
0x12C0	RMON Rx packets with > 2048 bytes (RMON_R_P_GTE2048)
0x12C4	RMON Rx octets (RMON_R_OCTETS)
0x12C8	Count of received frames not counted correctly (IEEE_R_DROP)
0x12CC	Frames received OK (IEEE_R_FRAME_OK)
0x12D0	Frames received with CRC error (IEEE_R_CRC)
0x12D4	Frames received with alignment error (IEEE_R_ALIGN)
0x12D8	Receive FIFO overflow count (IEEE_R_MACERR)

Table 17-4. MIB Counters Memory Map (continued)

IPSBAR Offset	Register
0x12DC	Flow control pause frames received (IEEE_R_FDXFC)
0x12E0	Octet count for frames received without error (IEEE_R_OCTETS_OK)

17.4.2 Ethernet Interrupt Event Register (EIR)

When an event occurs that sets a bit in EIR, an interrupt occurs if the corresponding bit in the interrupt mask register (EIMR) is also set. Writing a 1 to an EIR bit clears it; writing 0 has no effect. This register is cleared upon hardware reset.

These interrupts can be divided into operational interrupts, transceiver/network error interrupts, and internal error interrupts. Interrupts which may occur in normal operation are GRA, TXF, TXB, RXF, RXB, and MII. Interrupts resulting from errors/problems detected in the network or transceiver are HBERR, BABR, BABT, LC, and RL. Interrupts resulting from internal errors are HBERR and UN.

Some of the error interrupts are independently counted in the MIB block counters:

- HBERR - IEEE_T_SQE
- BABR - RMON_R_OVERSIZE (good CRC), RMON_R_JAB (bad CRC)
- BABT - RMON_T_OVERSIZE (good CRC), RMON_T_JAB (bad CRC)
- LATE_COL - IEEE_T_LCOL
- COL_RETRY_LIM - IEEE_T_EXCOL
- XFIFO_UN - IEEE_T_MACERR

Software may choose to mask off these interrupts because these errors are visible to network management via the MIB counters.

IPSBAR Offset: 0x1004 — Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HB ERR	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EB ERR	LC	RL	UN	0	0	0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 17-2. Ethernet Interrupt Event Register (EIR)

Table 17-5. EIR Field Descriptions

Field	Description
31 HBERR	Heartbeat error. Indicates TCR[HBC] is set and that the COL input was not asserted within the heartbeat window following a transmission.
30 BABR	Babbling receive error. Indicates a frame was received with length in excess of RCR[MAX_FL] bytes.
29 BABT	Babbling transmit error. Indicates the transmitted frame length exceeds RCR[MAX_FL] bytes. Usually this condition is caused by a frame that is too long is placed into the transmit data buffer(s). Truncation does not occur.
28 GRA	Graceful stop complete. Indicates the graceful stop is complete. During graceful stop the transmitter is placed into a pause state after completion of the frame currently being transmitted. This bit is set by one of three conditions: 1) A graceful stop initiated by the setting of the TCR[GTS] bit is now complete. 2) A graceful stop initiated by the setting of the TCR[TFC_PAUSE] bit is now complete. 3) A graceful stop initiated by the reception of a valid full duplex flow control pause frame is now complete. Refer to Section 17.5.11, “Full Duplex Flow Control.”
27 TXF	Transmit frame interrupt. Indicates a frame has been transmitted and the last corresponding buffer descriptor has been updated.
26 TXB	Transmit buffer interrupt. Indicates a transmit buffer descriptor has been updated.
25 RXF	Receive frame interrupt. Indicates a frame has been received and the last corresponding buffer descriptor has been updated.
24 RXB	Receive buffer interrupt. Indicates a receive buffer descriptor not the last in the frame has been updated.
23 MII	MII interrupt. Indicates the MII has completed the data transfer requested.
22 EBERR	Ethernet bus error. Indicates a system bus error occurred when a DMA transaction is underway. When the EBERR bit is set, ECR[ETHER_EN] is cleared, halting frame processing by the FEC. When this occurs, software needs to ensure that the FIFO controller and DMA also soft reset.
21 LC	Late collision. Indicates a collision occurred beyond the collision window (slot time) in half duplex mode. The frame truncates with a bad CRC and the remainder of the frame is discarded.
20 RL	Collision retry limit. Indicates a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. This error can only occur in half duplex mode.
19 UN	Transmit FIFO underrun. Indicates the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.
18–0	Reserved, must be cleared.

17.4.3 Interrupt Mask Register (EIMR)

The EIMR register controls which interrupt events are allowed to generate actual interrupts. All implemented bits in this CSR are read/write. A hardware reset clears this register. If the corresponding bits in the EIR and EIMR registers are set, an interrupt is generated. The interrupt signal remains asserted until a 1 is written to the EIR bit (write 1 to clear) or a 0 is written to the EIMR bit.

IPSBAR Offset: 0x1008 — Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HB ERR	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EB ERR	LC	RL	UN	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 17-3. Ethernet Interrupt Mask Register (EIMR)

Table 17-6. EIMR Field Descriptions

Field	Description
31–19 See Figure 17-3 and Table 17-5	Interrupt mask. Each bit corresponds to an interrupt source defined by the EIR register. The corresponding EIMR bit determines whether an interrupt condition can generate an interrupt. At every processor clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR bit reflects the state of the interrupt signal even if the corresponding EIMR bit is set. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
18–0	Reserved, must be cleared.

17.4.4 Receive Descriptor Active Register (RDAR)

RDAR is a command register, written by the user, indicating the receive descriptor ring is updated (the driver produced empty receive buffers with the empty bit set).

When the register is written, the RDAR bit is set. This is independent of the data actually written by the user. When set, the FEC polls the receive descriptor ring and processes receive frames (provided ECR[ETHER_EN] is also set). After the FEC polls a receive descriptor whose empty bit is not set, FEC clears the RDAR bit and ceases receive descriptor ring polling until the user sets the bit again, signifying that additional descriptors are placed into the receive descriptor ring.

The RDAR register is cleared at reset and when ECR[ETHER_EN] is cleared.

IPSBAR Offset: 0x1010 — Access: User 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	RDAR	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																																
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 17-4. Receive Descriptor Active Register (RDAR)

Table 17-7. RDAR Field Descriptions

Field	Description
31–25	Reserved, must be cleared.
24 RDAR	Set to 1 when this register is written, regardless of the value written. Cleared by the FEC device when no additional empty descriptors remain in the receive ring. Also cleared when ECR[ETHER_EN] is cleared.
23–0	Reserved, must be cleared.

17.4.5 Transmit Descriptor Active Register (TDAR)

The TDAR is a command register which the user writes to indicate the transmit descriptor ring is updated (transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor).

When the register is written, the TDAR bit is set. This value is independent of the data actually written by the user. When set, the FEC polls the transmit descriptor ring and processes transmit frames (provided ECR[ETHER_EN] is also set). After the FEC polls a transmit descriptor that is a ready bit not set, FEC clears the TDAR bit and ceases transmit descriptor ring polling until the user sets the bit again, signifying additional descriptors are placed into the transmit descriptor ring.

The TDAR register is cleared at reset, when ECR[ETHER_EN] is cleared, or when ECR[RESET] is set.

IPSBAR Offset: 0x1014 — Access: User 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	TDAR	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																																
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 17-5. Transmit Descriptor Active Register (TDAR)

Table 17-8. TDAR Field Descriptions

Field	Description
31–25	Reserved, must be cleared.
24 TDAR	Set to 1 when this register is written, regardless of the value written. Cleared by the FEC device when no additional ready descriptors remain in the transmit ring. Also cleared when ECR[ETHER_EN] is cleared.
23–0	Reserved, must be cleared.

17.4.6 Ethernet Control Register (ECR)

ECR is a read/write user register, though hardware may alter fields in this register as well. The ECR enables/disables the FEC.

IPSBAR Offset: 0x1024 Access: User 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	1	1	1	1	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	ETHER_EN	RESET
W																																
Reset	1	1	1	1	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 17-6. Ethernet Control Register (ECR)

Table 17-9. ECR Field Descriptions

Field	Description
31–2	Reserved, must be cleared.
1 ETHER_EN	When this bit is set, FEC is enabled, and reception and transmission are possible. When this bit is cleared, reception immediately stops and transmission stops after a bad CRC is appended to any currently transmitted frame. The buffer descriptor(s) for an aborted transmit frame are not updated after clearing this bit. When ETHER_EN is cleared, the DMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. Hardware alters the ETHER_EN bit under the following conditions: • ECR[RESET] is set by software, in which case ETHER_EN is cleared • An error condition causes the EIR[EBERR] bit to set, in which case ETHER_EN is cleared
0 RESET	When this bit is set, the equivalent of a hardware reset is performed but it is local to the FEC. ECR[ETHER_EN] is cleared and all other FEC registers take their reset values. Also, any transmission/reception currently in progress is abruptly aborted. This bit is automatically cleared by hardware during the reset sequence. The reset sequence takes approximately eight internal bus clock cycles after this bit is set.

17.4.7 MII Management Frame Register (MMFR)

The MMFR is user-accessible and does not reset to a defined value. The MMFR register is used to communicate with the attached MII compatible PHY device(s), providing read/write access to their MII registers. Performing a write to the MMFR causes a management frame to be sourced unless the MSCR is programmed to 0. If MSCR is cleared while MMFR is written and then MSCR is written with a non-zero value, an MII frame is generated with the data previously written to the MMFR. This allows MMFR and MSCR to be programmed in either order if MSCR is currently zero.

IPSBAR Offset: 0x1040 Access: User 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 W	ST				OP				PA					RA			TA				DATA												
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 17-7. MII Management Frame Register (MMFR)

Table 17-10. MMFR Field Descriptions

Field	Description
31–30 ST	Start of frame delimiter. These bits must be programmed to 0b01 for a valid MII management frame.
29–28 OP	Operation code. 00 Write frame operation, but not MII compliant. 01 Write frame operation for a valid MII management frame. 10 Read frame operation for a valid MII management frame. 11 Read frame operation, but not MII compliant.
27–23 PA	PHY address. This field specifies one of up to 32 attached PHY devices.
22–18 RA	Register address. This field specifies one of up to 32 registers within the specified PHY device.
17–16 TA	Turn around. This field must be programmed to 10 to generate a valid MII management frame.
15–0 DATA	Management frame data. This is the field for data to be written to or read from the PHY register.

To perform a read or write operation on the MII Management Interface, write the MMFR register. To generate a valid read or write management frame, ST field must be written with a 01 pattern, and the TA field must be written with a 10. If other patterns are written to these fields, a frame is generated, but does not comply with the IEEE 802.3 MII definition.

To generate an IEEE 802.3-compliant MII Management Interface write frame (write to a PHY register), the user must write {01 01 PHYAD REGAD 10 DATA} to the MMFR register. Writing this pattern causes the control logic to shift out the data in the MMFR register following a preamble generated by the control state machine. During this time, contents of the MMFR register are altered as the contents are serially shifted and are unpredictable if read by the user. After the write management frame operation completes, the MII interrupt is generated. At this time, contents of the MMFR register match the original value written.

To generate an MII management interface read frame (read a PHY register), the user must write {01 10 PHYAD REGAD 10 XXXX} to the MMFR register (the content of the DATA field is a don’t care). Writing this pattern causes the control logic to shift out the data in the MMFR register following a preamble generated by the control state machine. During this time, contents of the MMFR register are altered as the contents are serially shifted and are unpredictable if read by the user. After the read management frame operation completes, the MII interrupt is generated. At this time, the contents of the MMFR register match the original value written except for the DATA field whose contents are replaced by the value read from the PHY register.

If the MMFR register is written while frame generation is in progress, the frame contents are altered. Software must use the MII interrupt to avoid writing to the MMFR register while frame generation is in progress.

17.4.8 MII Speed Control Register (MSCR)

The MSCR provides control of the MII clock (FEC_MDC pin) frequency and allows a preamble drop on the MII management frame.

IPSBAR Offset: 0x1044 Access: User 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	DIS_PRE	MII_SPEED						0
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 17-8. MII Speed Control Register (MSCR)

Table 17-11. MSCR Field Descriptions

Field	Description
31–8	Reserved, must be cleared.
7 DIS_PRE	Setting this bit causes the preamble (32 ones) not to be prepended to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY device(s) does not require it.
6–1 MII_SPEED	Controls the frequency of the MII management interface clock (FEC_MDC) relative to the internal bus clock. A value of 0 in this field turns off the FEC_MDC and leaves it in low voltage state. Any non-zero value results in the FEC_MDC frequency of 1/(MII_SPEED × 2) of the internal bus frequency.
0	Reserved, must be cleared.

The MII_SPEED field must be programmed with a value to provide an FEC_MDC frequency of less than or equal to 2.5 MHz to be compliant with the IEEE 802.3 MII specification. The MII_SPEED must be set to a non-zero value to source a read or write management frame. After the management frame is complete, the MSCR register may optionally be set to 0 to turn off the FEC_MDC. The FEC_MDC generated has a 50% duty cycle except when MII_SPEED changes during operation (change takes effect following a rising or falling edge of FEC_MDC).

If the internal bus clock is 25 MHz, programming this register to 0x0000_0005 results in an FEC_MDC as stated the equation below.

$$25\ \text{MHz} \times \frac{1}{5 \times 2} = 2.5\ \text{MHz} \qquad \textit{Eqn. 17-1}$$

A table showing optimum values for MII_SPEED as a function of internal bus clock frequency is provided below.

Table 17-12. Programming Examples for MSCR

Internal FEC Clock Frequency	MSCR[MII_SPEED]	FEC_MDC frequency
25 MHz	0x5	2.50 MHz
33 MHz	0x7	2.36 MHz
40 MHz	0x8	2.50 MHz

Table 17-12. Programming Examples for MSCR (continued)

Internal FEC Clock Frequency	MSCR[MII_SPEED]	FEC_MDC frequency
50 MHz	0xA	2.50 MHz
66 MHz	0xE	2.36 MHz

17.4.9 MIB Control Register (MIBC)

The MIBC is a read/write register controlling and observing the state of the MIB block. User software accesses this register if there is a need to disable the MIB block operation. For example, to clear all MIB counters in RAM:

1. Disable the MIB block
2. Clear all the MIB RAM locations
3. Enable the MIB block

The MIB_DIS bit is reset to 1. See Table 17-4 for the locations of the MIB counters.

IPSBAR Offset: 0x1064 Access: User 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	MIB_DIS	MIB_IDLE	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																																
Reset	1	1	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 17-9. MIB Control Register (MIBC)

Table 17-13. MIBC Field Descriptions

Field	Description
31 MIB_DIS	A read/write control bit. If set, the MIB logic halts and not update any MIB counters.
30 MIB_IDLE	A read-only status bit. If set the MIB block is not currently updating any MIB counters.
29–0	Reserved.

17.4.10 Receive Control Register (RCR)

RCR controls the operational mode of the receive block and must be written only when ECR[ETHER_EN] is cleared (initialization time).

IPSBAR Offset: 0x1084 Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	FCE	BC_REJ	PROM	MII_MODE	DRT	LOOP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 17-10. Receive Control Register (RCR)

Table 17-14. RCR Field Descriptions

Field	Description
31–27	Reserved, must be cleared.
26–16 MAX_FL	Maximum frame length. Resets to decimal 1518. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL causes the BABT interrupt to occur. Receive frames longer than MAX_FL causes the BABR interrupt to occur and sets the LG bit in the end of frame receive buffer descriptor. The recommended default value to be programmed is 1518 or 1522 if VLAN tags are supported.
15–6	Reserved, must be cleared.
5 FCE	Flow control enable. If asserted, the receiver detects PAUSE frames. Upon PAUSE frame detection, the transmitter stops transmitting data frames for a given duration.
4 BC_REJ	Broadcast frame reject. If asserted, frames with DA (destination address) equal to FFFF_FFFF_FFFF are rejected unless the PROM bit is set. If BC_REJ and PROM are set, frames with broadcast DA are accepted and the M (MISS) is set in the receive buffer descriptor.
3 PROM	Promiscuous mode. All frames are accepted regardless of address matching.
2 MII_MODE	Media independent interface mode. Selects the external interface mode for transmit and receive blocks. 0 7-wire mode (used only for serial 10 Mbps) 1 MII mode
1 DRT	Disable receive on transmit. 0 Receive path operates independently of transmit (use for full duplex or to monitor transmit activity in half duplex mode). 1 Disable reception of frames while transmitting (normally used for half duplex mode).
0 LOOP	Internal loopback. If set, transmitted frames are looped back internal to the device and transmit output signals are not asserted. The internal bus clock substitutes for the FEC_TXCLK when LOOP is asserted. DRT must be set to 0 when setting LOOP.

17.4.11 Transmit Control Register (TCR)

TCR is read/write and configures the transmit block. This register is cleared at system reset. Bits 2 and 1 must be modified only when ECR[ETHER_EN] is cleared.

IPSBAR Offset: 0x10C4 Access: User 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	RFC_ PAUSE	TFC_ PAUSE	FDEN	HBC	GTS
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 17-11. Transmit Control Register (TCR)

Table 17-15. TCR Field Descriptions

Field	Description
31–5	Reserved, must be cleared.
4 RFC_PAUSE	Receive frame control pause. This read-only status bit is asserted when a full duplex flow control pause frame is received and the transmitter pauses for the duration defined in this pause frame. This bit automatically clears when the pause duration is complete.
3 TFC_PAUSE	Transmit frame control pause. Transmits a PAUSE frame when asserted. When this bit is set, the MAC stops transmission of data frames after the current transmission is complete. At this time, GRA interrupt in the EIR register is asserted. With transmission of data frames stopped, MAC transmits a MAC Control PAUSE frame. Next, the MAC clears the TFC_PAUSE bit and resumes transmitting data frames. If the transmitter pauses due to user assertion of GTS or reception of a PAUSE frame, the MAC may continue transmitting a MAC Control PAUSE frame.
2 FDEN	Full duplex enable. If set, frames transmit independent of carrier sense and collision inputs. This bit should only be modified when ECR[ETHER_EN] is cleared.
1 HBC	Heartbeat control. If set, the heartbeat check performs following end of transmission and the HB bit in the status register is set if the collision input does not assert within the heartbeat window. This bit should only be modified when ECR[ETHER_EN] is cleared.
0 GTS	Graceful transmit stop. When this bit is set, MAC stops transmission after any frame currently transmitted is complete and GRA interrupt in the EIR register is asserted. If frame transmission is not currently underway, the GRA interrupt is asserted immediately. After transmission finishes, clear GTS to restart. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS is set, transmission stops after the collision. The frame is transmitted again after GTS is cleared. There may be old frames in the transmit FIFO that transmit when GTS is reasserted. To avoid this, clear ECR[ETHER_EN] following the GRA interrupt.

17.4.12 Physical Address Lower Register (PALR)

PALR contains the lower 32 bits (bytes 0,1,2,3) of the 48-bit address used in the address recognition process to compare with the DA (destination address) field of receive frames with an individual DA. In addition, this register is used in bytes 0 through 3 of the 6-byte source address field when transmitting PAUSE frames. This register is not reset and you must initialize it.

IPSBAR Offset: 0x10E4 Access: User read/write

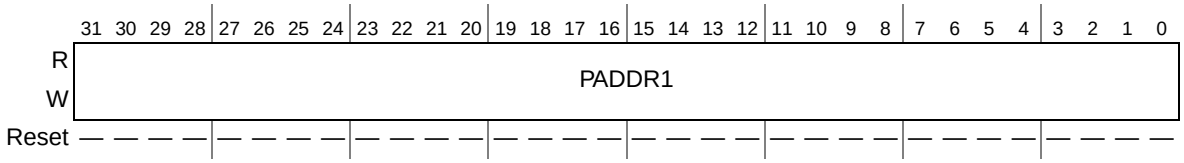


Figure 17-12. Physical Address Lower Register (PALR)

Table 17-16. PALR Field Descriptions

Field	Description
31–0 PADDR1	Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match and the source address field in PAUSE frames.

17.4.13 Physical Address Upper Register (PAUR)

PAUR contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to compare with the DA (destination address) field of receive frames with an individual DA. In addition, this register is used in bytes 4 and 5 of the 6-byte Source Address field when transmitting PAUSE frames. Bits 15:0 of PAUR contain a constant type field (0x8808) for transmission of PAUSE frames. The upper 16 bits of this register are not reset and you must initialize it.

IPSBAR Offset: 0x10E8 Access: User read/write

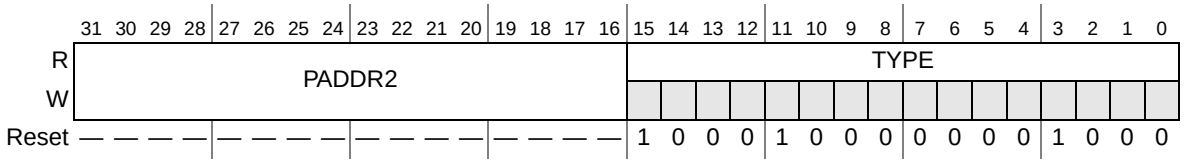


Figure 17-13. Physical Address Upper Register (PAUR)

Table 17-17. PAUR Field Descriptions

Field	Description
31–16 PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match, and the source address field in PAUSE frames.
15–0 TYPE	Type field in PAUSE frames. These 16 read-only bits are a constant value of 0x8808.

17.4.14 Opcode/Pause Duration Register (OPD)

The OPD is read/write accessible. This register contains the 16-bit opcode and 16-bit pause duration fields used in transmission of a PAUSE frame. The opcode field is a constant value, 0x0001. When another node detects a PAUSE frame, that node pauses transmission for the duration specified in the pause duration field. The lower 16 bits of this register are not reset and you must initialize them.

IPSBAR Offset: 0x10EC — Access: User 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	OPCODE																PAUSE_DUR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 17-14. Opcode/Pause Duration Register (OPD)

Table 17-18. OPD Field Descriptions

Field	Description
31–16 OPCODE	Opcode field used in PAUSE frames. These read-only bits are a constant, 0x0001.
15–0 PAUSE_DUR	Pause Duration field used in PAUSE frames.

17.4.15 Descriptor Individual Upper Address Register (IAUR)

IAUR contains the upper 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the destination address (DA) field of receive frames with an individual DA. This register is not reset and you must initialize it.

IPSBAR Offset: 0x1118 — Access: User 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	IADDR1																															
W																																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 17-15. Descriptor Individual Upper Address Register (IAUR)

Table 17-19. IAUR Field Descriptions

Field	Description
31–0 IADDR1	The upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

17.4.16 Descriptor Individual Lower Address Register (IALR)

IALR contains the lower 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the DA field of receive frames with an individual DA. This register is not reset and you must initialize it.

IPSBAR Offset: 0x111C — Access: User read/write

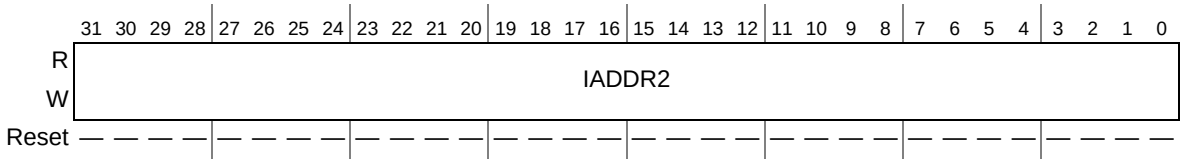


Figure 17-16. Descriptor Individual Lower Address Register (IALR)

Table 17-20. IALR Field Descriptions

Field	Description
31–0 IADDR2	The lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

17.4.17 Descriptor Group Upper Address Register (GAUR)

GAUR contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. You must initialize this register.

IPSBAR Offset: 0x1120 — Access: User read/write

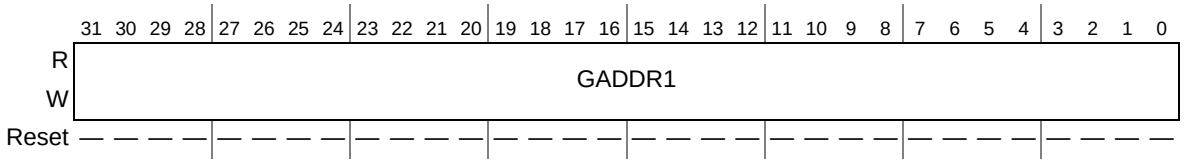


Figure 17-17. Descriptor Group Upper Address Register (GAUR)

Table 17-21. GAUR Field Descriptions

Field	Description
31–0 GADDR1	The GADDR1 register contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

17.4.18 Descriptor Group Lower Address Register (GALR)

GALR contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. You must initialize this register.

IPSBAR Offset: 0x1124 — Access: User read/write

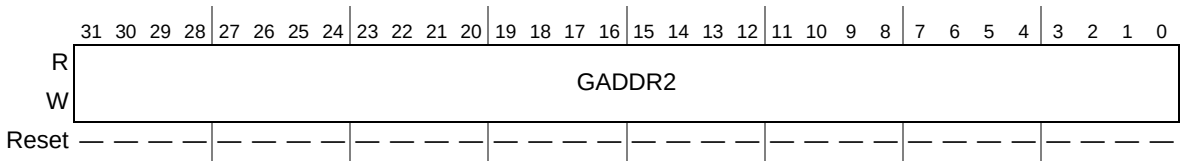


Figure 17-18. Descriptor Group Lower Address Register (GALR)

Table 17-22. GALR Field Descriptions

Field	Description
31–0 GADDR2	The GADDR2 register contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

17.4.19 Transmit FIFO Watermark Register (TFWR)

The TFWR controls the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows you to minimize transmit latency (TFWR = 00 or 01) or allow for larger bus access latency (TFWR = 11) due to contention for the system bus. Setting the watermark to a high value minimizes the risk of transmit FIFO underrun due to contention for the system bus. The byte counts associated with the TFWR field may need to be modified to match a given system requirement (worst case bus access latency by the transmit data DMA channel).

IPSBAR Offset: 0x1144 — Access: User 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	TFWR	
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 17-19. Transmit FIFO Watermark Register (TFWR)

Table 17-23. TFWR Field Descriptions

Field	Description
31–2	Reserved, must be cleared.
1–0 TFWR	Number of bytes written to transmit FIFO before transmission of a frame begins 00 64 bytes written 01 64 bytes written 10 128 bytes written 11 192 bytes written

17.4.20 FIFO Receive Bound Register (FRBR)

FRBR indicates the upper address bound of the FIFO RAM. Drivers can use this value, along with the FRSR, to appropriately divide the available FIFO RAM between the transmit and receive data paths.

IPSBAR Offset: 0x114C — 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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	R_BOUND								0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 17-20. FIFO Receive Bound Register (FRBR)

Table 17-24. FRBR Field Descriptions

Field	Description
31–10	Reserved, read as 0 (except bit 10, which is read as 1).
9–2 R_BOUND	Read-only. Highest valid FIFO RAM address.
1–0	Reserved, read as 0.

17.4.21 FIFO Receive Start Register (FRSR)

FRSR indicates the starting address of the receive FIFO. FRSR marks the boundary between the transmit and receive FIFOs. The transmit FIFO uses addresses from the start of the FIFO to the location four bytes before the address programmed into the FRSR. The receive FIFO uses addresses from FRSR to FRBR inclusive.

Hardware initializes the FRSR register at reset. FRSR only needs to be written to change the default value.

IPSBAR Offset: 0x1150 — Access: User 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	1	R_FSTART								0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

Figure 17-21. FIFO Receive Start Register (FRSR)

Table 17-25. FRSR Field Descriptions

Field	Description
31–11	Reserved, must be cleared.
10	Reserved, must be set.
9–2 R_FSTART	Address of first receive FIFO location. Acts as delimiter between receive and transmit FIFOs. For proper operation, ensure that R_FSTART is set to 0x48 or greater.
1–0	Reserved, must be cleared.

17.4.22 Receive Descriptor Ring Start Register (ERDSR)

ERDSR points to the start of the circular receive buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however, it is recommended it be made 128-bit aligned (evenly divisible by 16).

This register is not reset and must be initialized prior to operation.

IPSBAR Offset: 0x1180 Access: User 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	R_DES_START								0	0
W										
Reset	— — — —	— — — —	— — — —	— — — —	— — — —	— — — —	— — — —	— —	—	—

Figure 17-22. Ethernet Receive Descriptor Ring Start Register (ERDSR)

Table 17-26. ERDSR Field Descriptions

Field	Description
31–2 R_DES_START	Pointer to start of receive buffer descriptor queue.
1–0	Reserved, must be cleared.

17.4.23 Transmit Buffer Descriptor Ring Start Registers (ETSDR)

ETSDR provides a pointer to the start of the circular transmit buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however, it is recommended it be made 128-bit aligned (evenly divisible by 16). You should write zeros to bits 1 and 0. Hardware ignores non-zero values in these two bit positions.

This register is undefined at reset and must be initialized prior to operation.

IPSBAR Offset: 0x1184 Access: User 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	X_DES_START								0	0
W										
Reset	— — — —	— — — —	— — — —	— — — —	— — — —	— — — —	— — — —	— —	—	—

Figure 17-23. Transmit Buffer Descriptor Ring Start Register (ETDSR)

Table 17-27. ETDSR Field Descriptions

Field	Description
31–2 X_DES_START	Pointer to start of transmit buffer descriptor queue.
1–0	Reserved, must be cleared.

17.4.24 Receive Buffer Size Register (EMRBR)

The EMRBR is a user-programmable register that dictates the maximum size of all receive buffers. This value should take into consideration that the receive CRC is always written into the last receive buffer. To allow one maximum size frame per buffer, EMRBR must be set to RCR[MAX_FL] or larger. To properly align the buffer, EMRBR must be evenly divisible by 16. To ensure this, bits 3–0 are forced low.

To minimize bus utilization (descriptor fetches), it is recommended that EMRBR be greater than or equal to 256 bytes.

The EMRBR register is undefined at reset and must be initialized by the user.

IPSBAR Offset: 0x1188 Access: User 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	R_BUF_SIZE							0	0	0	0
W																																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 17-24. Receive Buffer Size Register (EMRBR)

Table 17-28. EMRBR Field Descriptions

Field	Description
31–11	Reserved, must be cleared.
10–4 R_BUF_SIZE	Maximum size of receive buffer size in bytes. To minimize bus utilization (descriptor fetches), set this field to 256 bytes (0x10) or larger. 0x10 256 + 15 bytes (minimum size recommended) 0x11 272 + 15 bytes ... 0x7F 2032 + 15 bytes. The FEC writes up to 2047 bytes in the receive buffer. If data larger than 2047 is received, the FEC truncates it and shows 0x7FF in the receive descriptor
3–0	Reserved, must be cleared.

17.5 Functional Description

This section describes the operation of the FEC, beginning with the buffer descriptors, the hardware and software initialization sequence, then the software (Ethernet driver) interface for transmitting and receiving frames.

Following the software initialization and operation sections are sections providing a detailed description of the functions of the FEC.

17.5.1 Buffer Descriptors

This section provides a description of the operation of the driver/DMA via the buffer descriptors. It is followed by a detailed description of the receive and transmit descriptor fields.

17.5.1.1 Driver/DMA Operation with Buffer Descriptors

The data for the FEC frames resides in one or more memory buffers external to the FEC. Associated with each buffer is a buffer descriptor (BD), which contains a starting address (32-bit aligned pointer), data length, and status/control information (which contains the current state for the buffer). To permit maximum user flexibility, the BDs are also located in external memory and are read by the FEC DMA engine.

Software produces buffers by allocating/initializing memory and initializing buffer descriptors. Setting the RxBD[E] or TxBD[R] bit produces the buffer. Software writing to TDAR or RDAR tells the FEC that a buffer is placed in external memory for the transmit or receive data traffic, respectively. The hardware reads the BDs and consumes the buffers after they have been produced. After the data DMA is complete and the DMA engine writes the buffer descriptor status bits, hardware clears RxBD[E] or TxBD[R] to signal the buffer has been consumed. Software may poll the BDs to detect when the buffers are consumed or may rely on the buffer/frame interrupts. The driver may process these buffers, and they can return to the free list.

The ECR[ETHER_EN] bit operates as a reset to the BD/DMA logic. When ECR[ETHER_EN] is cleared, the DMA engine BD pointers are reset to point to the starting transmit and receive BDs. The buffer descriptors are not initialized by hardware during reset. At least one transmit and receive buffer descriptor must be initialized by software before ECR[ETHER_EN] is set.

The buffer descriptors operate as two separate rings. ERDSR defines the starting address for receive BDs and ETDSR defines the starting address for transmit BDs. The wrap (W) bit defines the last buffer descriptor in each ring. When W is set, the next descriptor in the ring is at the location pointed to by ERDSR and ETDSR for the receive and transmit rings, respectively. Buffer descriptor rings must start on a 32-bit boundary; however, it is recommended they are made 128-bit aligned.

17.5.1.1.1 Driver/DMA Operation with Transmit BDs

Typically, a transmit frame is divided between multiple buffers. An example is to have an application payload in one buffer, TCP header in a second buffer, IP header in a third buffer, and Ethernet/IEEE 802.3 header in a fouth buffer. The Ethernet MAC does not prepend the Ethernet header (destination address, source address, length/type field(s)), so the driver must provide this in one of the transmit buffers. The Ethernet MAC can append the Ethernet CRC to the frame. TxBD[TC], which must be set by the driver, determines whether the MAC or driver appends the CRC.

The driver (TxBD software producer) should set up Tx BDs so a complete transmit frame is given to the hardware at once. If a transmit frame consists of three buffers, the BDs should be initialized with pointer, length, and control (W, L, TC, ABC) and then the TxBD[R] bit should be set in reverse order (third, second, then first BD) to ensure that the complete frame is ready in memory before the DMA begins. If the TxBDs are set up in order, the DMA controller could DMA the first BD before the second was made available, potentially causing a transmit FIFO underrun.

In the FEC, the driver notifies the DMA that new transmit frame(s) are available by writing to TDAR. When this register is written to (data value is not significant) the FEC, RISC tells the DMA to read the next transmit BD in the ring. After started, the RISC + DMA continues to read and interpret transmit BDs in order and DMA the associated buffers until a transmit BD is encountered with the R bit cleared. At this point, the FEC polls this BD one more time. If the R bit is cleared the second time, RISC stops the transmit descriptor read process until software sets up another transmit frame and writes to TDAR.

When the DMA of each transmit buffer is complete, the DMA writes back to the BD to clear the R bit, indicating that the hardware consumer is finished with the buffer.

17.5.1.1.2 Driver/DMA Operation with Receive BDs

Unlike transmit, the length of the receive frame is unknown by the driver ahead of time. Therefore, the driver must set a variable to define the length of all receive buffers. In the FEC, this variable is written to the EMRBR register.

The driver (RxBD software producer) should set up some number of empty buffers for the Ethernet by initializing the address field and the E and W bits of the associated receive BDs. The hardware (receive DMA) consumes these buffers by filling them with data as frames are received and clearing the E bit and writing to the L bit (1 indicates last buffer in frame), the frame status bits (if L is set), and the length field.

If a receive frame spans multiple receive buffers, the L bit is only set for the last buffer in the frame. For non-last buffers, the length field in the receive BD is written by the DMA (at the same time the E bit is cleared) with the default receive buffer length value. For end-of-frame buffers, the receive BD is written with L set and information written to the status bits (M, BC, MC, LG, NO, CR, OV, TR). Some of the status bits are error indicators which, if set, indicate the receive frame should be discarded and not given to higher layers. The frame status/length information is written into the receive FIFO following the end of the frame (as a single 32-bit word) by the receive logic. The length field for the end of frame buffer is written with the length of the entire frame, not only the length of the last buffer.

For simplicity, the driver may assign a large enough default receive buffer length to contain an entire frame, keeping in mind that a malfunction on the network or out-of-spec implementation could result in giant frames. Frames of 2K (2048) bytes or larger are truncated by the FEC at 2047 bytes so software never sees a receive frame larger than 2047 bytes.

Similar to transmit, the FEC polls the receive descriptor ring after the driver sets up receive BDs and writes to the RDAR register. As frames are received, the FEC fills receive buffers and updates the associated BDs, then reads the next BD in the receive descriptor ring. If the FEC reads a receive BD and finds the E bit cleared, it polls this BD once more. If RxBD[E] is clear a second time, FEC stops reading receive BDs until the driver writes to RDAR.

17.5.1.2 Ethernet Receive Buffer Descriptor (RxBD)

In the RxBD, the user initializes the E and W bits in the first longword and the pointer in the second longword. When the buffer has been DMA'd, the Ethernet controller modifies the E, L, M, BC, MC, LG, NO, CR, OV, and TR bits and writes the length of the used portion of the buffer in the first longword. The M, BC, MC, LG, NO, CR, OV, and TR bits in the first longword of the buffer descriptor are only modified by the Ethernet controller when the L bit is set.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset + 0	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 2	Data Length															
Offset + 4	Rx Data Buffer Pointer - A[31:16]															
Offset + 6	Rx Data Buffer Pointer - A[15:0]															

Figure 17-25. Receive Buffer Descriptor (RxBD)

Table 17-29. Receive Buffer Descriptor Field Definitions

Word	Field	Description
Offset + 0	15 E	Empty. Written by the FEC (=0) and user (=1). 0 The data buffer associated with this BD is filled with received data, or data reception has aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 0	14 RO1	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
Offset + 0	13 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ERDSR.
Offset + 0	12 RO2	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
Offset + 0	11 L	Last in frame. Written by the FEC. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
Offset + 0	10–9	Reserved, must be cleared.
Offset + 0	8 M	Miss. Written by the FEC. This bit is set by the FEC for frames accepted in promiscuous mode, but flagged as a miss by the internal address recognition. Therefore, while in promiscuous mode, you can use the M-bit to quickly determine whether the frame was destined to this station. This bit is valid only if the L-bit is set and the PROM bit is set. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode.
Offset + 0	7 BC	Set if the DA is broadcast (FFFF_FFFF_FFFF).
Offset + 0	6 MC	Set if the DA is multicast and not BC.
Offset + 0	5 LG	Rx frame length violation. Written by the FEC. A frame length greater than RCR[MAX_FL] was recognized. This bit is valid only if the L-bit is set. The receive data is not altered in any way unless the length exceeds 2047 bytes.
Offset + 0	4 NO	Receive non-octet aligned frame. Written by the FEC. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error. This bit is valid only if the L-bit is set. If this bit is set, the CR bit is not set.
Offset + 0	3	Reserved, must be cleared.
Offset + 0	2 CR	Receive CRC error. Written by the FEC. This frame contains a CRC error and is an integral number of octets in length. This bit is valid only if the L-bit is set.
Offset + 0	1 OV	Overrun. Written by the FEC. A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, CR, and CL lose their normal meaning and are zero. This bit is valid only if the L-bit is set.
Offset + 0	0 TR	Set if the receive frame is truncated (frame length > 2047 bytes). If the TR bit is set, the frame must be discarded and the other error bits must be ignored as they may be incorrect.
Offset + 2	15–0 Data Length	Data length. Written by the FEC. Data length is the number of octets written by the FEC into this BD's data buffer if L equals 0 (the value is equal to EMRBR), or the length of the frame including CRC if L is set. It is written by the FEC once as the BD is closed.

Table 17-29. Receive Buffer Descriptor Field Definitions (continued)

Word	Field	Description
0ffset + 4	15–0 A[31:16]	RX data buffer pointer, bits [31:16][1]
Offset + 6	15–0 A[15:0]	RX data buffer pointer, bits [15:0]

1 The receive buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 16. The buffer must reside in memory external to the FEC. The Ethernet controller never modifies this value.

NOTE

When the software driver sets an E bit in one or more receive descriptors, the driver should follow with a write to RDAR.

17.5.1.3 Ethernet Transmit Buffer Descriptor (TxBD)

Data is presented to the FEC for transmission by arranging it in buffers referenced by the channel's TxBDs. The Ethernet controller confirms transmission by clearing the ready bit (TxBD[R]) when DMA of the buffer is complete. In the TxBD, the user initializes the R, W, L, and TC bits and the length (in bytes) in the first longword and the buffer pointer in the second longword.

The FEC clears the R bit when the buffer is transferred. Status bits for the buffer/frame are not included in the transmit buffer descriptors. Transmit frame status is indicated via individual interrupt bits (error conditions) and in statistic counters in the MIB block. See Section 17.4.1, "MIB Block Counters Memory Map," for more details.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset + 0	R	TO1	W	TO2	L	TC	ABC	—	—	—	—	—	—	—	—	—
Offset + 2	Data Length															
Offset + 4	Tx Data Buffer Pointer - A[31:16]															
Offset + 6	Tx Data Buffer Pointer - A[15:0]															

Figure 17-26. Transmit Buffer Descriptor (TxBD)

Table 17-30. Transmit Buffer Descriptor Field Definitions

Word	Field	Description
Offset + 0	15 R	Ready. Written by the FEC and you. 0 The data buffer associated with this BD is not ready for transmission. You are free to manipulate this BD or its associated data buffer. The FEC clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, prepared for transmission by you, has not been transmitted or currently transmits. You may write no fields of this BD after this bit is set.
Offset + 0	14 TO1	Transmit software ownership. This field is reserved for software use. This read/write bit is not modified by hardware nor does its value affect hardware.

Table 17-30. Transmit Buffer Descriptor Field Definitions (continued)

Word	Field	Description
Offset + 0	13 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ETDSR.
Offset + 0	12 TO2	Transmit software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware nor does its value affect hardware.
Offset + 0	11 L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame 1 The buffer is the last in the transmit frame
Offset + 0	10 TC	Transmit CRC. Written by user (only valid if L is set). 0 End transmission immediately after the last data byte 1 Transmit the CRC sequence after the last data byte
Offset + 0	9 ABC	Append bad CRC. Written by user (only valid if L is set). 0 No effect 1 Transmit the CRC sequence inverted after the last data byte (regardless of TC value)
Offset + 0	8–0	Reserved, must be cleared.
Offset + 2	15–0 Data Length	Data length, written by user. Data length is the number of octets the FEC should transmit from this BD's data buffer. It is never modified by the FEC.
Offset + 4	15–0 A[31:16]	Tx data buffer pointer, bits [31:16][1]
Offset + 6	15–0 A[15:0]	Tx data buffer pointer, bits [15:0]

[1] The transmit buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 4. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller.

NOTE

After the software driver has set up the buffers for a frame, it should set up the corresponding BDs. The last step in setting up the BDs for a transmit frame is setting the R bit in the first BD for the frame. The driver must follow that with a write to TDAR that triggers the FEC to poll the next BD in the ring.

17.5.2 Initialization Sequence

This section describes which registers are reset due to hardware reset, which are reset by the FEC RISC, and what locations you must initialize prior to enabling the FEC.

17.5.2.1 Hardware Controlled Initialization

In the FEC, hardware resets registers and control logic that generate interrupts. A hardware reset negates output signals and resets general configuration bits.

Other registers reset when the ECR[ETHER_EN] bit is cleared (which is accomplished by a hard reset or software to halt operation). By clearing ECR[ETHER_EN], configuration control registers such as the TCR and RCR are not reset, but the entire data path is reset.

Table 17-31. ECR[ETHER_EN] De-Assertion Effect on FEC

Register/Machine	Reset Value
XMIT block	Transmission is aborted (bad CRC appended)
RECV block	Receive activity is aborted
DMA block	All DMA activity is terminated
RDAR	Cleared
TDAR	Cleared
Descriptor Controller block	Halt operation

17.5.3 User Initialization (Prior to Setting ECR[ETHER_EN])

You need to initialize portions the FEC prior to setting the ECR[ETHER_EN] bit. The exact values depend on the particular application. The sequence is not important.

Table 17-32 defines Ethernet MAC registers requiring initialization.

Table 17-32. User Initialization (Before ECR[ETHER_EN])

Description
Initialize EIMR
Clear EIR (write 0xFFFF_FFFF)
TFWR (optional)
IALR / IAUR
GAUR / GALR
PALR / PAUR (only needed for full duplex flow control)
OPD (only needed for full duplex flow control)
RCR
TCR
MSCR (optional)
Clear MIB_RAM

Table 17-33 defines FEC FIFO/DMA registers that require initialization.

Table 17-33. FEC User Initialization (Before ECR[ETHER_EN])

Description
Initialize FRSR (optional)
Initialize EMRBR
Initialize ERDSR
Initialize ETDSR

Table 17-33. FEC User Initialization (Before ECR[ETHER_EN]) (continued)

Description
Initialize (Empty) Transmit Descriptor ring
Initialize (Empty) Receive Descriptor ring

17.5.4 Microcontroller Initialization

In the FEC, the descriptor control RISC initializes some registers after ECR[ETHER_EN] is asserted. After the microcontroller initialization sequence is complete, hardware is ready for operation.

Table 17-34 shows microcontroller initialization operations.

Table 17-34. Microcontroller Initialization

Description
Initialize BackOff Random Number Seed
Activate Receiver
Activate Transmitter
Clear Transmit FIFO
Clear Receive FIFO
Initialize Transmit Ring Pointer
Initialize Receive Ring Pointer
Initialize FIFO Count Registers

17.5.5 User Initialization (After Setting ECR[ETHER_EN])

After setting ECR[ETHER_EN], you can set up the buffer/frame descriptors and write to TDAR and RDAR. Refer to Section 17.5.1, “Buffer Descriptors,” for more details.

17.5.6 Network Interface Options

The FEC supports an MII interface for 10/100 Mbps Ethernet and a 7-wire serial interface for 10 Mbps Ethernet. The RCR[MII_MODE] bit select the interface mode. In MII mode (RCR[MII_MODE] set), there are 18 signals defined by the IEEE 802.3 standard and supported by the EMAC. Table 17-35 shows these signals.

Table 17-35. MII Mode

Signal Description	EMAC pin
Transmit Clock	FEC_TXCLK
Transmit Enable	FEC_TXEN
Transmit Data	FEC_TXD[3:0]
Transmit Error	FEC_TXER

Table 17-35. MII Mode (continued)

Signal Description	EMAC pin
Collision	FEC_COL
Carrier Sense	FEC_CRS
Receive Clock	FEC_RXCLK
Receive Data Valid	FEC_RXDV
Receive Data	FEC_RXD[3:0]
Receive Error	FEC_RXER
Management Data Clock	FEC_MDC
Management Data Input/Output	FEC_MDIO

The 7-wire serial mode interface (RCR[MII_MODE] cleared) is generally referred to as AMD mode. Table 17-36 shows the 7-wire mode connections to the external transceiver.

Table 17-36. 7-Wire Mode Configuration

Signal description	EMAC Pin
Transmit Clock	FEC_TXCLK
Transmit Enable	FEC_TXEN
Transmit Data	FEC_TXD[0]
Collision	FEC_COL
Receive Clock	FEC_RXCLK
Receive Data Valid	FEC_RXDV
Receive Data	FEC_RXD[0]

17.5.7 FEC Frame Transmission

The Ethernet transmitter is designed to work with almost no intervention from software. After ECR[ETHER_EN] is set and data appears in the transmit FIFO, the Ethernet MAC can transmit onto the network. The Ethernet controller transmits bytes least significant bit (lsb) first.

When the transmit FIFO fills to the watermark (defined by TFWR), MAC transmit logic asserts FEC_TXEN and starts transmitting the preamble (PA) sequence, the start frame delimiter (SFD), and then the frame information from the FIFO. However, the controller defers the transmission if the network is busy (FEC_CRS is asserted). Before transmitting, the controller waits for carrier sense to become inactive, then determines if carrier sense stays inactive for 60 bit times. If so, transmission begins after waiting an additional 36 bit times (96 bit times after carrier sense originally became inactive). See Section 17.5.15.1, “Transmission Errors,” for more details.

If a collision occurs during transmission of the frame (half duplex mode), the Ethernet controller follows the specified backoff procedures and attempts to retransmit the frame until the retry limit is reached. The transmit FIFO stores at least the first 64 bytes of the transmit frame, so they do not have to be retrieved from system memory in case of a collision. This improves bus utilization and latency in case immediate retransmission is necessary.

When all the frame data is transmitted, FCS (frame check sequence) or 32-bit cyclic redundancy check (CRC) bytes are appended if the TC bit is set in the transmit frame control word. If the ABC bit is set in the transmit frame control word, a bad CRC is appended to the frame data regardless of the TC bit value. Following the transmission of the CRC, the Ethernet controller writes the frame status information to the MIB block. Transmit logic automatically pads short frames (if the TC bit in the transmit buffer descriptor for the end of frame buffer is set).

Settings in the EIMR determine interrupts generated to the buffer (TXB) and frame (TFINT).

The transmit error interrupts are HBERR, BABT, LATE_COL, COL_RETRY_LIM, and XFIFO_UN. If the transmit frame length exceeds MAX_FL bytes, BABT interrupt is asserted. However, the entire frame is transmitted (no truncation).

To pause transmission, set TCR[GTS] (graceful transmit stop). The FEC transmitter stops immediately if transmission is not in progress; otherwise, it continues transmission until the current frame finishes or terminates with a collision. After the transmitter has stopped, the GRA (graceful stop complete) interrupt is asserted. If TCR[GTS] is cleared, the FEC resumes transmission with the next frame.

17.5.7.1 Duplicate Frame Transmission

The FEC fetches transmit buffer descriptors (TxBDs) and the corresponding transmit data continuously until the transmit FIFO is full. It does not determine whether the TxBD to be fetched is already being processed internally (as a result of a wrap). As the FEC nears the end of the transmission of one frame, it begins to DMA the data for the next frame. To remain one BD ahead of the DMA, it also fetches the TxBD for the next frame. It is possible that the FEC fetches from memory a BD that has already been processed but not yet written back (it is read a second time with the R bit remains set). In this case, the data is fetched and transmitted again.

Using at least three TxBDs fixes this problem for large frames, but not for small frames. To ensure correct operation for large or small frames, one of the following must be true:

- The FEC software driver ensures that there is always at least one TxBD with the ready bit cleared.
- Every frame uses more than one TxBD and every TxBD but the last is written back immediately after the data is fetched.
- The FEC software driver ensures a minimum frame size, *n*. The minimum number of TxBDs is then (Tx FIFO Size ÷ (*n* + 4)) rounded up to the nearest integer (though the result cannot be less than three). The default Tx FIFO size is 192 bytes; this size is programmable.

17.5.8 FEC Frame Reception

The FEC receiver works with almost no intervention from the host and can perform address recognition, CRC checking, short frame checking, and maximum frame length checking. The Ethernet controller receives serial data lsb first.

When the driver enables the FEC receiver by setting ECR[ETHER_EN], it immediately starts processing receive frames. When FEC_RXDV is asserted, the receiver first checks for a valid PA/SFD header. If the PA/SFD is valid, it is stripped and the receiver processes the frame. If a valid PA/SFD is not found, the frame is ignored.

In serial mode, the first 16 bit times of RX_D0 following assertion of FEC_RXDV are ignored. Following the first 16 bit times, the data sequence is checked for alternating 1/0s. If a 11 or 00 data sequence is detected during bit times 17 to 21, the remainder of the frame is ignored. After bit time 21, the data sequence is monitored for a valid SFD (11). If a 00 is detected, the frame is rejected. When a 11 is detected, the PA/SFD sequence is complete.

In MII mode, the receiver checks for at least one byte matching the SFD. Zero or more PA bytes may occur, but if a 00 bit sequence is detected prior to the SFD byte, the frame is ignored.

After the first 6 bytes of the frame are received, the FEC performs address recognition on the frame.

After a collision window (64 bytes) of data is received and if address recognition has not rejected the frame, the receive FIFO signals the frame is accepted and may be passed on to the DMA. If the frame is a runt (due to collision) or is rejected by address recognition, the receive FIFO is notified to reject the frame. Therefore, no collision fragments are presented to you except late collisions, which indicate serious LAN problems.

During reception, the Ethernet controller checks for various error conditions and after the entire frame is written into the FIFO, a 32-bit frame status word is written into the FIFO. This status word contains the M, BC, MC, LG, NO, CR, OV, and TR status bits, and the frame length. See Section 17.5.15.2, “Reception Errors,” for more details.

Receive buffer (RXB) and frame interrupts (RFINT) may be generated if enabled by the EIMR register. A receive error interrupt is a babbling receiver error (BABR). Receive frames are not truncated if they exceed the max frame length (MAX_FL); however, the BABR interrupt occurs and the LG bit in the receive buffer descriptor (RxBD) is set. See Section 17.5.1.2, “Ethernet Receive Buffer Descriptor (RxBD),” for more details.

When the receive frame is complete, the FEC sets the L-bit in the RxBD, writes the other frame status bits into the RxBD, and clears the E-bit. The Ethernet controller next generates a maskable interrupt (RFINT bit in EIR, maskable by RFIEN bit in EIMR), indicating that a frame is received and is in memory. The Ethernet controller then waits for a new frame.

17.5.9 Ethernet Address Recognition

The FEC filters the received frames based on destination address (DA) type — individual (unicast), group (multicast), or broadcast (all-ones group address). The difference between an individual address and a

group address is determined by the I/G bit in the destination address field. A flowchart for address recognition on received frames appears in the figures below.

Address recognition is accomplished through the use of the receive block and microcode running on the microcontroller. The flowchart shown in Figure 17-27 illustrates the address recognition decisions made by the receive block, while Figure 17-28 illustrates the decisions made by the microcontroller.

If the DA is a broadcast address and broadcast reject (RCR[BC_REJ]) is cleared, then the frame is accepted unconditionally, as shown in Figure 17-27. Otherwise, if the DA is not a broadcast address, then the microcontroller runs the address recognition subroutine, as shown in Figure 17-28.

If the DA is a group (multicast) address and flow control is disabled, then the microcontroller performs a group hash table lookup using the 64-entry hash table programmed in GAUR and GALR. If a hash match occurs, the receiver accepts the frame.

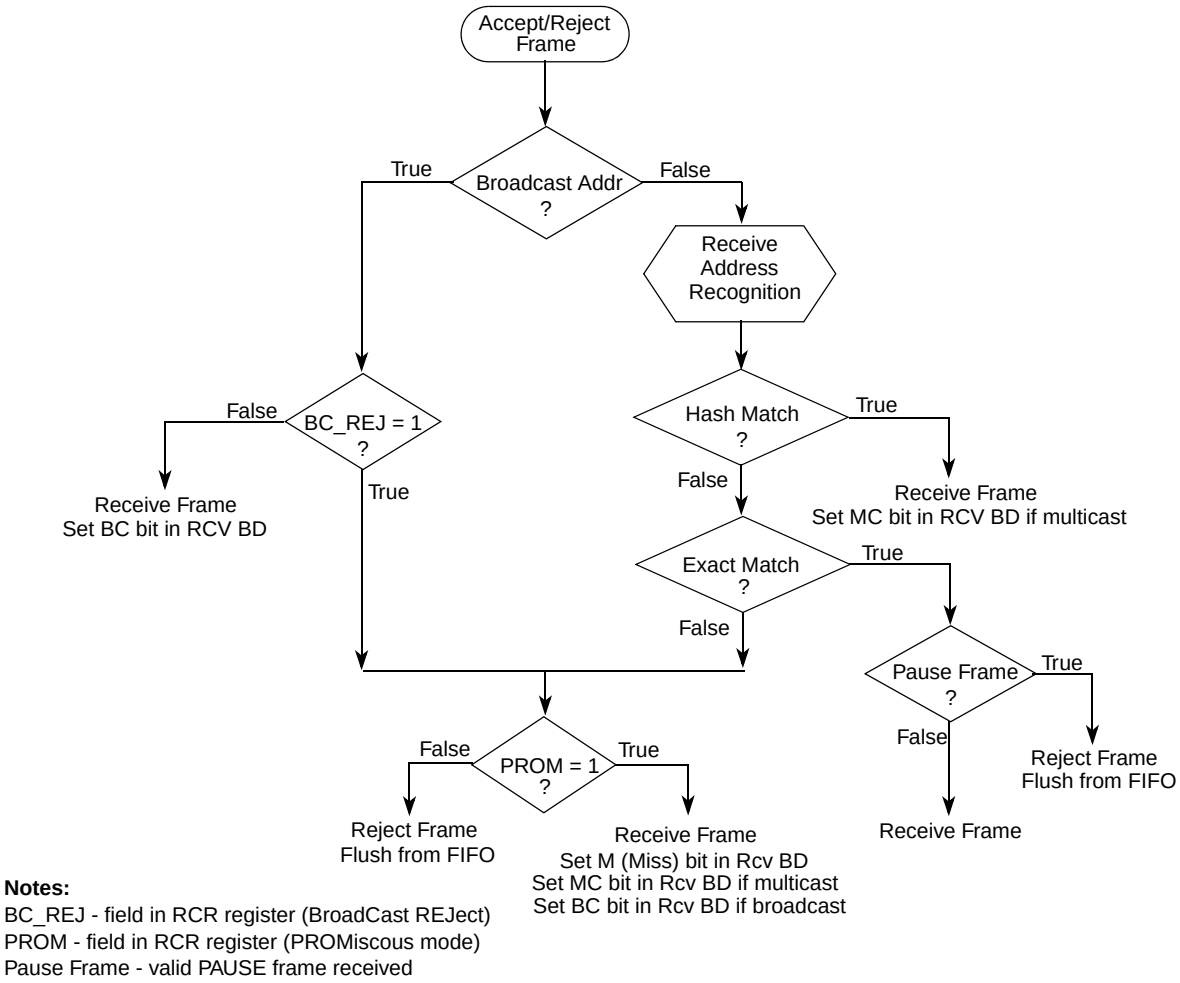
If flow control is enabled, the microcontroller does an exact address match check between the DA and the designated PAUSE DA (01:80:C2:00:00:01). If the receive block determines the received frame is a valid PAUSE frame, the frame is rejected. The receiver detects a PAUSE frame with the DA field set to the designated PAUSE DA or the unicast physical address.

If the DA is the individual (unicast) address, the microcontroller performs an individual exact match comparison between the DA and 48-bit physical address that you program in the PALR and PAUR registers. If an exact match occurs, the frame is accepted; otherwise, the microcontroller does an individual hash table lookup using the 64-entry hash table programmed in registers, IAUR and IALR. In the case of an individual hash match, the frame is accepted. Again, the receiver accepts or rejects the frame based on PAUSE frame detection, shown in Figure 17-27.

If neither a hash match (group or individual) nor an exact match (group or individual) occur, and if promiscuous mode is enabled (RCR[PROM] set), the frame is accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame is rejected.

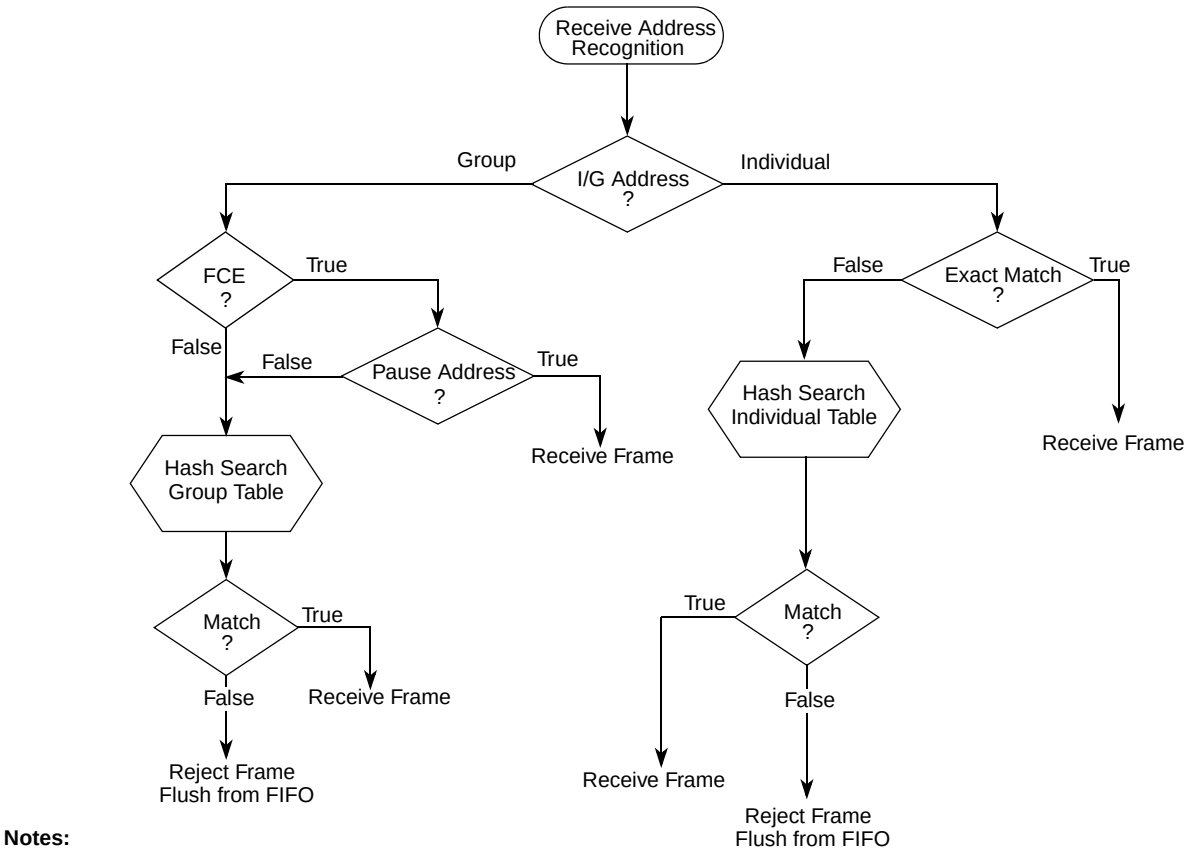
Similarly, if the DA is a broadcast address, broadcast reject (RCR[BC_REJ]) is asserted, and promiscuous mode is enabled, the frame is accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame is rejected.

In general, when a frame is rejected, it is flushed from the FIFO.



Notes:
BC_REJ - field in RCR register (BroadCast REJect)
PROM - field in RCR register (PROMiscous mode)
Pause Frame - valid PAUSE frame received

Figure 17-27. Ethernet Address Recognition—Receive Block Decisions



Notes:
FCE - field in RCR register (flow control enable)
I/G - Individual/Group bit in destination address (lsb in first byte received in MAC frame)

Figure 17-28. Ethernet Address Recognition—Microcode Decisions

17.5.10 Hash Algorithm

The hash table algorithm used in the group and individual hash filtering operates as follows. The 48-bit destination address is mapped into one of 64 bits, represented by 64 bits stored in GAUR, GALR (group address hash match), or IAUR, IALR (individual address hash match). This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the six most significant bits of the CRC-encoded result to generate a number between 0 and 63. The msb of the CRC result selects GAUR (msb = 1) or GALR (msb = 0). The five least significant bits of the hash result select the bit within the selected register. If the CRC generator selects a bit set in the hash table, the frame is accepted; otherwise, it is rejected.

For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The user must initialize the hash table registers. Use this CRC32 polynomial to compute the hash:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1 \quad \textit{Eqn. 17-2}$$

Table 17-37 contains example destination addresses and corresponding hash values.

Table 17-37. Destination Address to 6-Bit Hash

48-bit DA	6-bit Hash (in hex)	Hash Decimal Value
65FF_FFFF_FFFF	0x0	0
55FF_FFFF_FFFF	0x1	1
15FF_FFFF_FFFF	0x2	2
35FF_FFFF_FFFF	0x3	3
B5FF_FFFF_FFFF	0x4	4
95FF_FFFF_FFFF	0x5	5
D5FF_FFFF_FFFF	0x6	6
F5FF_FFFF_FFFF	0x7	7
DBFF_FFFF_FFFF	0x8	8
FBFF_FFFF_FFFF	0x9	9
BBFF_FFFF_FFFF	0xA	10
8BFF_FFFF_FFFF	0xB	11
0BFF_FFFF_FFFF	0xC	12
3BFF_FFFF_FFFF	0xD	13
7BFF_FFFF_FFFF	0xE	14
5BFF_FFFF_FFFF	0xF	15
27FF_FFFF_FFFF	0x10	16
07FF_FFFF_FFFF	0x11	17
57FF_FFFF_FFFF	0x12	18
77FF_FFFF_FFFF	0x13	19
F7FF_FFFF_FFFF	0x14	20
C7FF_FFFF_FFFF	0x15	21
97FF_FFFF_FFFF	0x16	22
A7FF_FFFF_FFFF	0x17	23
99FF_FFFF_FFFF	0x18	24
B9FF_FFFF_FFFF	0x19	25
F9FF_FFFF_FFFF	0x1A	26
C9FF_FFFF_FFFF	0x1B	27

Table 17-37. Destination Address to 6-Bit Hash (continued)

48-bit DA	6-bit Hash (in hex)	Hash Decimal Value
59FF_FFFF_FFFF	0x1C	28
79FF_FFFF_FFFF	0x1D	29
29FF_FFFF_FFFF	0x1E	30
19FF_FFFF_FFFF	0x1F	31
D1FF_FFFF_FFFF	0x20	32
F1FF_FFFF_FFFF	0x21	33
B1FF_FFFF_FFFF	0x22	34
91FF_FFFF_FFFF	0x23	35
11FF_FFFF_FFFF	0x24	36
31FF_FFFF_FFFF	0x25	37
71FF_FFFF_FFFF	0x26	38
51FF_FFFF_FFFF	0x27	39
7FFF_FFFF_FFFF	0x28	40
4FFF_FFFF_FFFF	0x29	41
1FFF_FFFF_FFFF	0x2A	42
3FFF_FFFF_FFFF	0x2B	43
BFFF_FFFF_FFFF	0x2C	44
9FFF_FFFF_FFFF	0x2D	45
DFFF_FFFF_FFFF	0x2E	46
EFFF_FFFF_FFFF	0x2F	47
93FF_FFFF_FFFF	0x30	48
B3FF_FFFF_FFFF	0x31	49
F3FF_FFFF_FFFF	0x32	50
D3FF_FFFF_FFFF	0x33	51
53FF_FFFF_FFFF	0x34	52
73FF_FFFF_FFFF	0x35	53
23FF_FFFF_FFFF	0x36	54
13FF_FFFF_FFFF	0x37	55
3DFF_FFFF_FFFF	0x38	56
0DFF_FFFF_FFFF	0x39	57
5DFF_FFFF_FFFF	0x3A	58
7DFF_FFFF_FFFF	0x3B	59

Table 17-37. Destination Address to 6-Bit Hash (continued)

48-bit DA	6-bit Hash (in hex)	Hash Decimal Value
FDFF_FFFF_FFFF	0x3C	60
DDFF_FFFF_FFFF	0x3D	61
9DFF_FFFF_FFFF	0x3E	62
BDFF_FFFF_FFFF	0x3F	63

17.5.11 Full Duplex Flow Control

Full-duplex flow control allows you to transmit pause frames and to detect received pause frames. Upon detection of a pause frame, MAC data frame transmission stops for a given pause duration.

To enable PAUSE frame detection, the FEC must operate in full-duplex mode (TCR[FDEN] set) with flow control (RCR[FCE] set). The FEC detects a pause frame when the fields of the incoming frame match the pause frame specifications, as shown in Table 17-38. In addition, the receive status associated with the frame should indicate that the frame is valid.

Table 17-38. PAUSE Frame Field Specification

48-bit Destination Address	0x0180_C200_0001 or Physical Address
48-bit Source Address	Any
16-bit Type	0x8808
16-bit Opcode	0x0001
16-bit PAUSE Duration	0x0000 – 0xFFFF

The receiver and microcontroller modules perform PAUSE frame detection. The microcontroller runs an address recognition subroutine to detect the specified pause frame destination address, while the receiver detects the type and opcode pause frame fields. On detection of a pause frame, TCR[GTS] is set by the FEC internally. When transmission has paused, the EIR[GRA] interrupt is asserted and the pause timer begins to increment. The pause timer uses the transmit backoff timer hardware for tracking the appropriate collision backoff time in half-duplex mode. The pause timer increments once every slot time, until OPD[PAUSE_DUR] slot times have expired. On OPD[PAUSE_DUR] expiration, TCR[GTS] is cleared allowing MAC data frame transmission to resume. The receive flow control pause status bit (TCR[RFC_PAUSE]) is set while the transmitter pauses due to reception of a pause frame.

To transmit a pause frame, the FEC must operate in full-duplex mode and you must set flow control pause (TCR[TFC_PAUSE]). After TCR[TFC_PAUSE] is set, the transmitter sets TCR[GTS] internally. When the transmission of data frames stops, the EIR[GRA] (graceful stop complete) interrupt asserts and the pause frame is transmitted. TCR[TFC_PAUSE,GTS] are then cleared internally.

You must specify the desired pause duration in the OPD register.

When the transmitter pauses due to receiver/microcontroller pause frame detection, TCR[TFC_PAUSE] may remain set and cause the transmission of a single pause frame. In this case, the EIR[GRA] interrupt is not asserted.

17.5.12 Inter-Packet Gap (IPG) Time

The minimum inter-packet gap time for back-to-back transmission is 96 bit times. After completing a transmission or after the backoff algorithm completes, the transmitter waits for carrier sense to be negated before starting its 96 bit time IPG counter. Frame transmission may begin 96 bit times after carrier sense is negated if it stays negated for at least 60 bit times. If carrier sense asserts during the last 36 bit times, it is ignored and a collision occurs.

The receiver accepts back-to-back frames with a minimum spacing of at least 28 bit times. If an inter-packet gap between receive frames is less than 28 bit times, the receiver may discard the following frame.

17.5.13 Collision Managing

If a collision occurs during frame transmission, the Ethernet controller continues the transmission for at least 32 bit times, transmitting a JAM pattern consisting of 32 ones. If the collision occurs during the preamble sequence, a JAM pattern is sent after the end of the preamble sequence.

If a collision occurs within 512 bit times (one slot time), the retry process is initiated. The transmitter waits a random number of slot times. If a collision occurs after 512 bit times, then no retransmission is performed and the end of frame buffer is closed with a Late Collision (LC) error indication.

17.5.14 MII Internal and External Loopback

Internal and external loopback are supported by the Ethernet controller. In loopback mode, both of the FIFOs are used and the FEC actually operates in a full-duplex fashion. Internal and external loopback are configured using combinations of the RCR[LOOP, DRT] and TCR[FDEN] bits.

Set FDEN for internal and external loopback.

For internal loopback, set RCR[LOOP] and clear RCR[DRT]. FEC_TXEN and FEC_TXER do not assert during internal loopback. During internal loopback, the transmit/receive data rate is higher than in normal operation because the transmit and receive blocks use the internal bus clock instead of the clocks from the external transceiver. This causes an increase in the required system bus bandwidth for transmit and receive data being DMA’d to/from external memory. It may be necessary to pace the frames on the transmit side and/or limit the size of the frames to prevent transmit FIFO underruns and receive FIFO overflows.

For external loopback, clear RCR[LOOP] and RCR[DRT], and configure the external transceiver for loopback.

17.5.15 Ethernet Error-Managing Procedure

The Ethernet controller reports frame reception and transmission error conditions using the MIB block counters, the FEC RxBDs, and the EIR register.

17.5.15.1 Transmission Errors

17.5.15.1.1 Transmitter Underrun

If this error occurs, the FEC sends 32 bits that ensure a CRC error and stops transmitting. All remaining buffers for that frame are then flushed and closed, and EIR[UN] is set. The FEC then continues to the next transmit buffer descriptor and begin transmitting the next frame. The UN interrupt is asserted if enabled in the EIMR register.

17.5.15.1.2 Retransmission Attempts Limit Expired

When this error occurs, the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and EIR[RL] is set. The FEC then continues to the next transmit buffer descriptor and begins transmitting the next frame. The RL interrupt is asserted if enabled in the EIMR register.

17.5.15.1.3 Late Collision

When a collision occurs after the slot time (512 bits starting at the Preamble), the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and EIR[LC] is set. The FEC then continues to the next transmit buffer descriptor and begin transmitting the next frame. The LC interrupt is asserted if enabled in the EIMR register.

17.5.15.1.4 Heartbeat

Some transceivers have a self-test feature called heartbeat or signal quality error. To signify a good self-test, the transceiver indicates a collision to the FEC within four microseconds after completion of a frame transmitted by the Ethernet controller. This indication of a collision does not imply a real collision error on the network, but is rather an indication that the transceiver continues to function properly. This is the heartbeat condition.

If TCR[HBC] is set and the heartbeat condition is not detected by the FEC after a frame transmission, a heartbeat error occurs. When this error occurs, the FEC closes the buffer, sets EIR[HB], and generates the HBERR interrupt if it is enabled.

17.5.15.2 Reception Errors

17.5.15.2.1 Overrun Error

If the receive block has data to put into the receive FIFO and the receive FIFO is full, FEC sets RxBD[OV]. All subsequent data in the frame is discarded and subsequent frames may also be discarded until the receive FIFO is serviced by the DMA and space is made available. At this point the receive frame/status word is written into the FIFO with the OV bit set. The driver must discard this frame.

17.5.15.2.2 Non-Octet Error (Dribbling Bits)

The Ethernet controller manages up to seven dribbling bits when the receive frame terminates past an non-octet aligned boundary. Dribbling bits are not used in the CRC calculation. If there is a CRC error, the frame non-octet aligned (NO) error is reported in the RxBD. If there is no CRC error, no error is reported.

17.5.15.2.3 CRC Error

When a CRC error occurs with no dribble bits, FEC closes the buffer and sets RxBD[CR]. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

17.5.15.2.4 Frame Length Violation

When the receive frame length exceeds MAX_FL bytes the BABR interrupt is generated, and RxBD[LG] is set. The frame is not truncated unless the frame length exceeds 2047 bytes.

17.5.15.2.5 Truncation

When the receive frame length exceeds 2047 bytes, frame is truncated and RxBD[TR] is set.

Chapter 18
Watchdog Timer Module

18.1 Introduction

The watchdog timer is a 16-bit timer used to help software recover from runaway code. The watchdog timer has a free-running down-counter (watchdog counter) that generates a reset on underflow. To prevent a reset, software must periodically restart the countdown by servicing the watchdog.

18.2 Low-Power Mode Operation

This subsection describes the operation of the watchdog module in low-power modes and halted mode of operation. Low-power modes are described in Chapter 7, "Power Management." Table 3-1 shows the watchdog 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 if WCR[WAIT] cleared, stopped otherwise	Upon Watchdog reset
Doze	Normal if WCR[DOZE] cleared, stopped otherwise	Upon Watchdog reset
Stop	Stopped	No
Halted	Normal if WCR[HALTED] cleared, stopped otherwise	Upon Watchdog reset

In wait mode with the watchdog control register's WAIT bit (WCR[WAIT]) set, watchdog timer operation stops. In wait mode with the WCR[WAIT] bit cleared, the watchdog timer continues to operate normally. In doze mode with the WCR[DOZE] bit set, the watchdog timer module operation stops. In doze mode with the WCR[DOZE] bit cleared, the watchdog timer continues to operate normally. Watchdog timer operation stops in stop mode. When stop mode is exited, the watchdog timer continues to operate in its pre-stop mode state.

In halted mode with the WCR[HALTED] bit set, watchdog timer module operation stops. In halted mode with the WCR[HALTED] bit cleared, the watchdog timer continues to operate normally. When halted mode is exited, watchdog timer operation continues from the state it was in before entering halted mode, but any updates made in halted mode remain.

18.3 Block Diagram

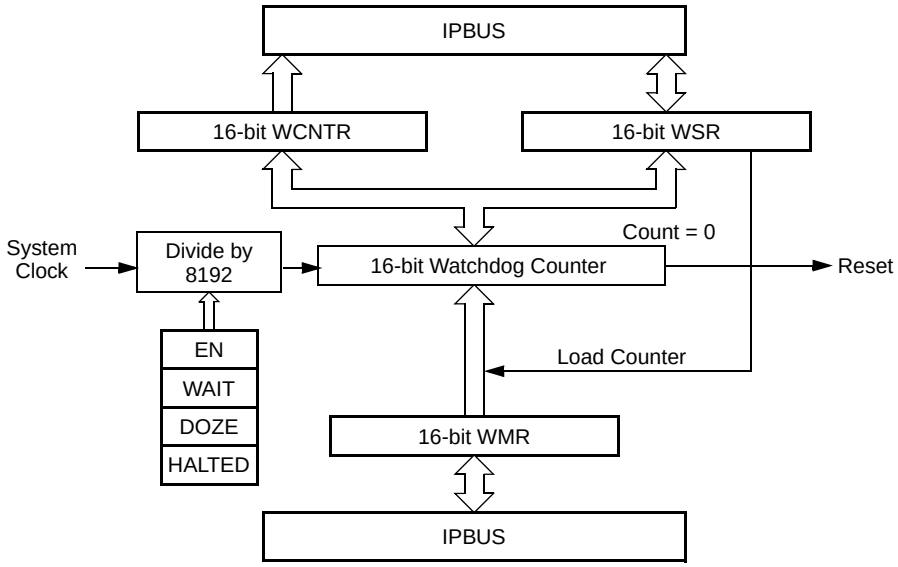


Figure 18-1. Watchdog Timer Block Diagram

18.4 Signals

The watchdog timer module has no off-chip signals.

18.5 Memory Map and Registers

This subsection describes the memory map and registers for the watchdog timer. The watchdog timer has a IPSBAR offset for base address of 0x0014_0000.

18.5.1 Memory Map

Refer to Table 18-2 for an overview of the watchdog memory map.

Table 18-2. Watchdog Timer Module Memory Map

IPSBAR Offset	Bits 15–8	Bits 7–0	Access[1]
0x0014_0000	Watchdog Control Register (WCR)		S
0x0014_0002	Watchdog Modulus Register (WMR)		S
0x0014_0004	Watchdog Count Register (WCNTR)		S/U
0x0014_0006	Watchdog Service Register (WSR)		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.

18.5.2 Registers

The watchdog timer programming model consists of these registers:

- Watchdog control register (WCR), which configures watchdog timer operation
- Watchdog modulus register (WMR), which determines the timer modulus reload value
- Watchdog count register (WCNTR), which provides visibility to the watchdog counter value
- Watchdog service register (WSR), which requires a service sequence to prevent reset

18.5.2.1 Watchdog Control Register (WCR)

The 16-bit WCR configures watchdog timer operation.

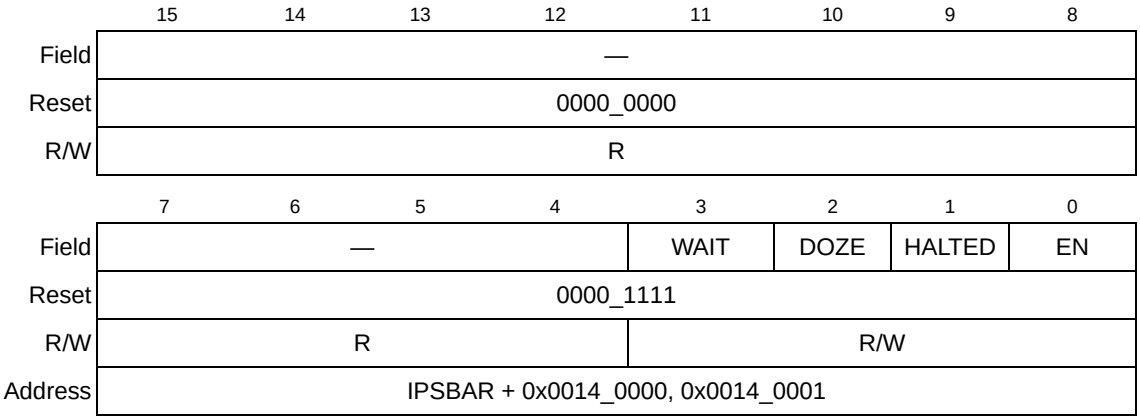


Figure 18-2. Watchdog Control Register (WCR)

Table 18-3. WCR Field Descriptions

Bit(s)	Name	Description
15–4	—	Reserved, should be cleared.
3	WAIT	Wait mode bit. Controls the function of the watchdog timer in wait mode. Once written, the WAIT bit is not affected by further writes except in halted mode. Reset sets WAIT. 1 Watchdog timer stopped in wait mode 0 Watchdog timer not affected in wait mode
2	DOZE	Doze mode bit. Controls the function of the watchdog timer in doze mode. Once written, the DOZE bit is not affected by further writes except in halted mode. Reset sets DOZE. 1 Watchdog timer stopped in doze mode 0 Watchdog timer not affected in doze mode
1	HALTED	Halted mode bit. Controls the function of the watchdog timer in halted mode. Once written, the HALTED bit is not affected by further writes except in halted mode. During halted mode, watchdog timer registers can be written and read normally. When halted mode is exited, timer operation continues from the state it was in before entering halted mode, but any updates made in halted mode remain. If a write-once register is written for the first time in halted mode, the register is still writable when halted mode is exited. 1 Watchdog timer stopped in halted mode 0 Watchdog timer not affected in halted mode Note: Changing the HALTED bit from 1 to 0 during halted mode starts the watchdog timer. Changing the HALTED bit from 0 to 1 during halted mode stops the watchdog timer.
0	EN	Watchdog enable bit. Enables the watchdog timer. Once written, the EN bit is not affected by further writes except in halted mode. When the watchdog timer is disabled, the watchdog counter and prescaler counter are held in a stopped state. 1 Watchdog timer enabled 0 Watchdog timer disabled

18.5.2.2 Watchdog Modulus Register (WMR)

	15	14	13	12	11	10	9	8
Field	WM15	WM14	WM13	WM12	WM11	WM10	WM9	WM8
Reset	1111_1111							
R/W	R/W							

	7	6	5	4	3	2	1	0
Field	WM7	WM6	WM5	WM4	WM3	WM2	WM1	WM0
Reset	1111_1111							
R/W	R/W							
Address	IPSBAR + 0x0014_0002, 0x0014_0003							

Figure 18-3. Watchdog Modulus Register (WMR)

Table 18-4. WMR Field Descriptions

Bit(s)	Name	Description
15–0	WM	Watchdog modulus. Contains the modulus that is reloaded into the watchdog counter by a service sequence. Once written, the WM[15:0] field is not affected by further writes except in halted mode. Writing to WMR immediately loads the new modulus value into the watchdog counter. The new value is also used at the next and all subsequent reloads. Reading WMR returns the value in the modulus register. Reset initializes the WM[15:0] field to 0xFFFF. Note: The prescaler counter is reset anytime a new value is loaded into the watchdog counter and also during reset.

18.5.2.3 Watchdog Count Register (WCNTR)

	15	14	13	12	11	10	9	8
Field	WC15	WC14	WC13	WC12	WC11	WC10	WC9	WC8
Reset	1111_1111							
R/W	R							

	7	6	5	4	3	2	1	0
Field	WC7	WC6	WC5	WC4	WC3	WC2	WC1	WC0
Reset	1111_1111							
R/W	R							
Address	IPSBAR + x0014_0004, 0x0014_0005							

Figure 18-4. Watchdog Count Register (WCNTR)

Table 18-5. WCNTR Field Descriptions

Bit(s)	Name	Description
15–0	WC	Watchdog count field. Reflects the current value in the watchdog counter. Reading the 16-bit WCNTR with two 8-bit reads is not guaranteed to return a coherent value. Writing to WCNTR has no effect, and write cycles are terminated normally.

18.5.2.4 Watchdog Service Register (WSR)

When the watchdog timer is enabled, writing 0x5555 and then 0xAAAA to WSR before the watchdog counter times out prevents a reset. If WSR is not serviced before the timeout, the watchdog timer sends a signal to the reset controller module that sets the RSR[WDR] bit and asserts a system reset.

Both writes must occur in the order listed before the timeout, but any number of instructions can be executed between the two writes. However, writing any value other than 0x5555 or 0xAAAA to WSR resets the servicing sequence, requiring both values to be written to keep the watchdog timer from causing a reset.

	15	14	13	12	11	10	9	8
Field	WS15	WS14	WS13	WS12	WS11	WS10	WS9	WS8
Reset	0000_0000							
R/W	R/W							

	7	6	5	4	3	2	1	0
Field	WS7	WS6	WS5	WS4	WS3	WS2	WS1	WS0
Reset	0000_0000							
R/W	R/W							
Address	IPSBAR + 0x0014_0006, 0x0014_0007							

Figure 18-5. Watchdog Service Register (WSR)

Chapter 19
Programmable Interrupt Timers (PIT0–PIT3)

19.1 Introduction

This chapter describes the operation of the four programmable interrupt timer modules: PIT0–PIT3.

19.1.1 Overview

Each PIT is a 16-bit timer that provides precise interrupts at regular intervals with minimal processor intervention. The timer can count down from the value written in the modulus register or it can be a free-running down-counter.

19.1.2 Block Diagram

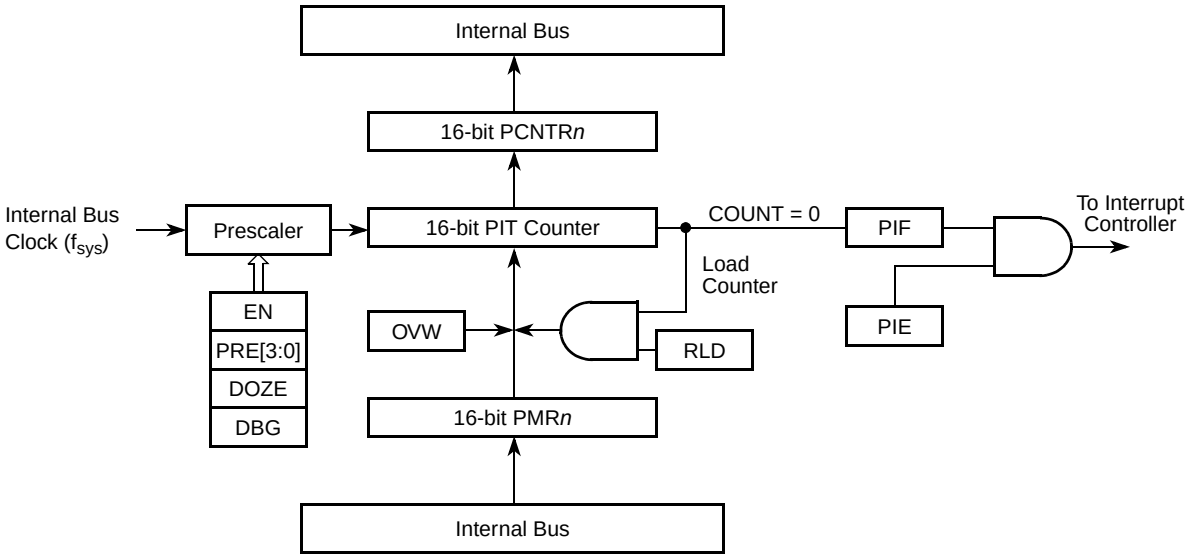


Figure 19-1. PIT Block Diagram

19.1.3 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 7, “Power Management.” Table 19-1 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 19-1. PIT Module Operation in Low-power Modes

Low-power Mode	PIT Operation	Mode Exit
Wait	Normal	N/A
Doze	Normal if PCSR*n*[DOZE] cleared, stopped otherwise	Any interrupt at or above level in LPICR, exit doze mode if PCSR*n*[DOZE] is set. Otherwise interrupt assertion has no effect.
Stop	Stopped	No
Debug	Normal if PCSR*n*[DBG] cleared, stopped otherwise	No. Any interrupt is 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 PCSR*n*[DOZE] bit set, PIT module operation stops. In doze mode with the PCSR*n*[DOZE] bit cleared, doze mode does not affect PIT operation. When doze mode is exited, PIT continues operating 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 PCSR*n*[DBG] bit set, PIT module operation stops. In debug mode with the PCSR*n*[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.

19.2 Memory Map/Register Definition

This section contains a memory map (see Table 19-2) and describes the register structure for PIT0–PIT3.

Table 19-2. Programmable Interrupt Timer Modules Memory Map

IPSBAR Offset PIT 0 PIT 1 PIT 2 PIT 3	Register	Width (bits)	Access[1]	Reset Value	Section/Page
Supervisor Access Only Registers[2]					
0x15_0000 0x16_0000 0x17_0000 0x18_0000	PIT Control and Status Register (PCSR*n*)	16	R/W	0x0000	19.2.1/19-3
0x15_0002 0x16_0002 0x17_0002 0x18_0002	PIT Modulus Register (PMR*n*)	16	R/W	0xFFFF	19.2.2/19-5

Table 19-2. Programmable Interrupt Timer Modules Memory Map (continued)

IPSBAR Offset PIT 0 PIT 1 PIT 2 PIT 3	Register	Width (bits)	Access[1]	Reset Value	Section/Page
User/Supervisor Access Registers					
0x15_0004 0x16_0004 0x17_0004 0x18_0004	PIT Count Register (PCNTR*n*)	16	R	0xFFFF	19.2.3/19-5

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.

19.2.1 PIT Control and Status Register (PCSR*n*)

The PCSR*n* registers configure the corresponding timer's operation.

IPSBAR Offset: 0x15_0000 (PCSR0), 0x16_0000 (PCSR1), 0x17_0000 (PCSR2), 0x18_0000 (PCSR3)

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 19-2. PCSR*n* Register

Table 19-3. PCSR*n* Field Descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>15–12</td><td>Reserved, must be cleared.</td></tr>
<tr><td>11–8
PRE</td><td>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></td></tr>
<tr><td>7</td><td>Reserved, must be cleared.</td></tr>
<tr><td>6
DOZE</td><td>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.</td></tr>
<tr><td>5
DBG</td><td>Debug mode bit. Controls the function of 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.</td></tr>
<tr><td>4
OVW</td><td>Overwrite. Enables writing to PMRn to immediately overwrite the value in the PIT counter.
0 Value in PMRn replaces value in PIT counter when count reaches 0x0000.
1 Writing PMRn immediately replaces value in PIT counter.</td></tr>
<tr><td>3
PIE</td><td>PIT interrupt enable. This read/write bit enables PIF flag to generate interrupt requests.
0 PIF interrupt requests disabled
1 PIF interrupt requests enabled</td></tr>
<tr><td>2
PIF</td><td>PIT interrupt flag. This read/write bit is set when 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.</td></tr>
</table>

Table 19-3. PCSR*n* Field Descriptions (continued)

Field	Description
1 RLD	Reload bit. The read/write reload bit enables loading the value of PMR*n* into 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 PIT is disabled, counter and prescaler are held in a stopped state. This bit is read anytime, write anytime. 0 PIT disabled 1 PIT enabled

19.2.2 PIT Modulus Register (PMR*n*)

The 16-bit read/write PMR*n* contains the timer modulus value 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)
Offset: 0x16_0002 (PMR1)
0x17_0002 (PMR2)
0x18_0002 (PMR3)

Access: Supervisor read/write

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

Figure 19-3. PIT Modulus Register (PMR*n*)

Table 19-4. PMR*n* Field Descriptions

Field	Description
15–0 PM	Timer modulus. The value of this register is loaded into the PIT counter when the count reaches zero and the PCSR*n*[RLD] bit is set. However, if PCSR*n*[OVW] is set, the value written to this field is immediately loaded into the counter. Reading this field returns the value written.

19.2.3 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 coherent. Writing to PCNTR*n* has no effect, and write cycles are terminated normally.

IPSBAR 0x15_0004 (PCNTR0)
Offset: 0x16_0004 (PCNTR1)
0x17_0004 (PCNTR2)
0x18_0004 (PCNTR3)

Access: User read only

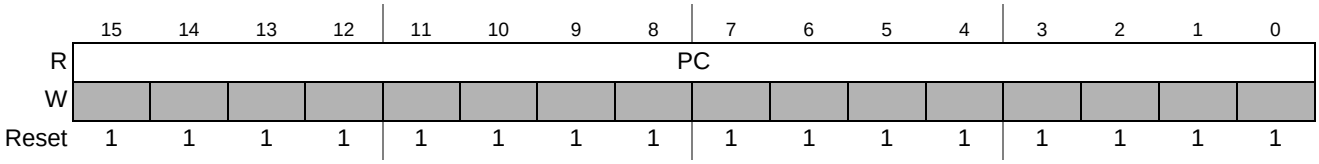


Figure 19-4. PIT Count Register (PCNTR*n*)

Table 19-5. PCNTR*n* Field Descriptions

Field	Description
15–0 PC	Counter value. Reading this field with two 8-bit reads is not guaranteed coherent. Writing to PCNTR*n* has no effect, and write cycles are terminated normally.

19.3 Functional Description

This section describes the PIT functional operation.

19.3.1 Set-and-Forget Timer Operation

This mode of operation is selected when the RLD bit in the PCSR register is set.

When PIT counter reaches a count of 0x0000, PIF flag is set in PCSR*n*. The value in the modulus register loads into the counter, and the counter begins decrementing toward 0x0000. If the PCSR*n*[PIE] bit is set, the PIF flag issues an interrupt request to the CPU.

When the PCSR*n*[OVW] bit is set, the counter can be directly initialized by writing to PMR*n* without having to wait for the count to reach 0x0000.

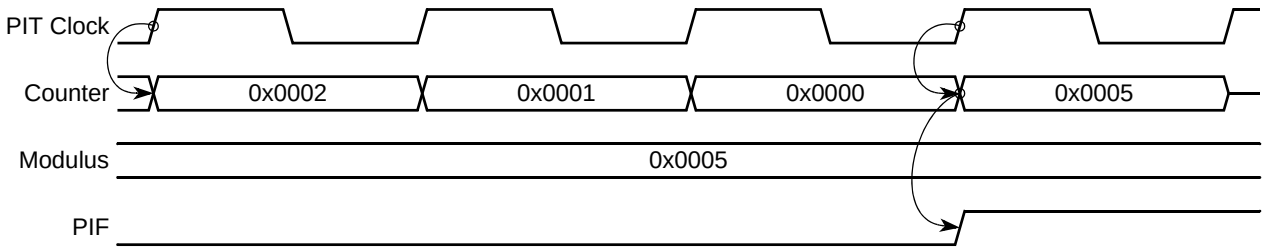


Figure 19-5. Counter Reloading from the Modulus Latch

19.3.2 Free-Running Timer Operation

This mode of operation is selected when the PCSR*n*[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, PCSR*n*[PIF] flag is set. If the PCSR*n*[PIE] bit is set, PIF flag issues an interrupt request to the CPU.

When the PCSR*n*[OVW] bit is set, counter can be directly initialized by writing to PMR*n* without having to wait for the count to reach 0x0000.

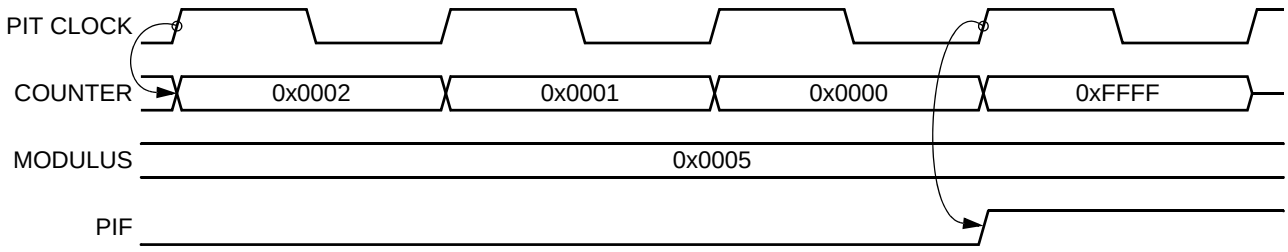


Figure 19-6. Counter in Free-Running Mode

19.3.3 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 PCSR*n*[PRE] bits. The PMR*n*[PM] bits select the timeout period.

$$\text{Timeout period} = \frac{2^{\text{PCSRn[PRE]}} \times (\text{PMRn[PM]} + 1)}{f_{sys}} \qquad \textit{Eqn. 19-1}$$

19.3.4 Interrupt Operation

Table 19-6 shows the interrupt request generated by the PIT.

Table 19-6. 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 20
General Purpose Timer Modules (GPTA and GPTB)

The processor has two 4-channel general purpose timer modules (GPTA and GPTB). Each 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.

20.1 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)

20.2 Block Diagram

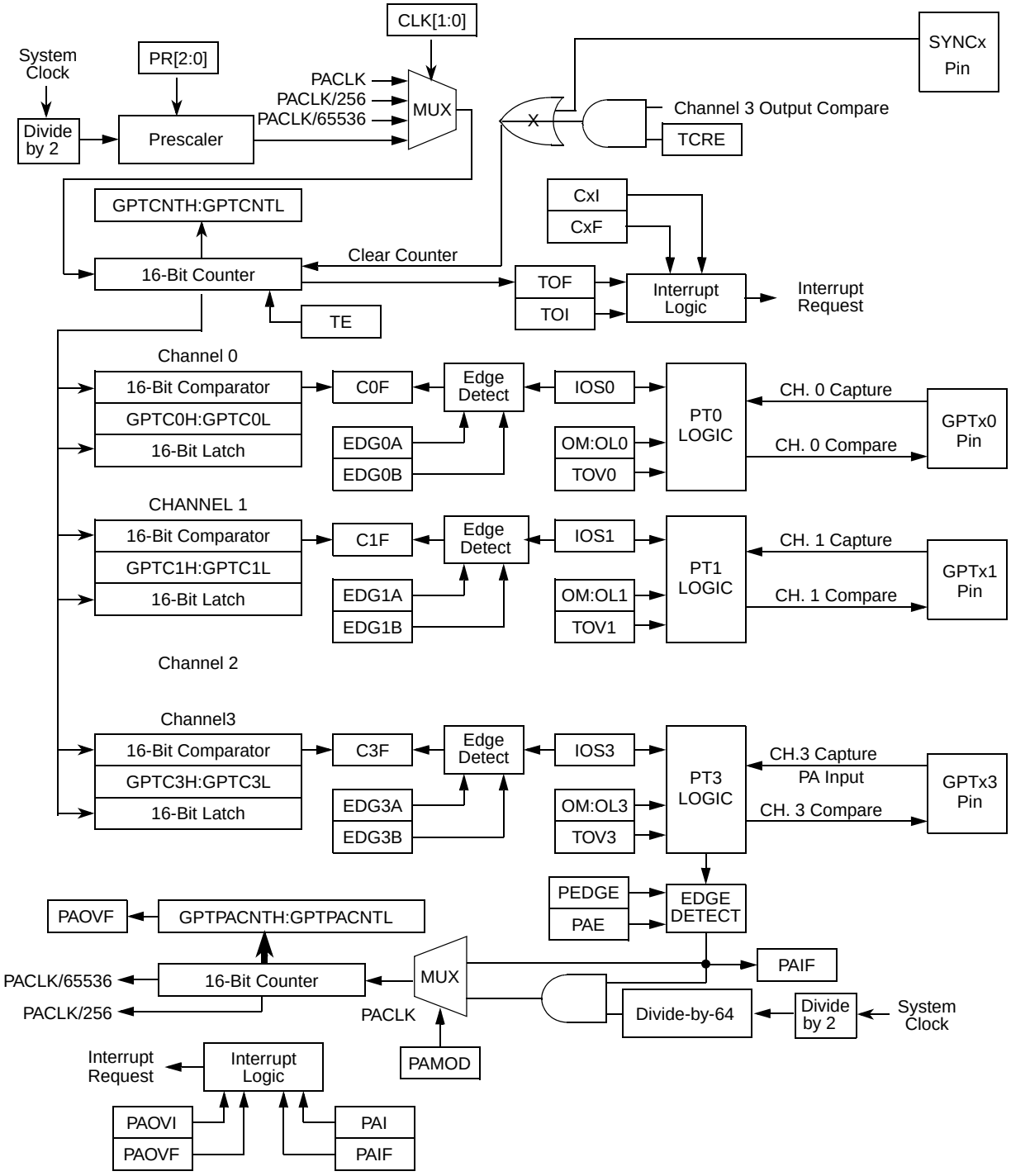


Figure 20-1. GPT Block Diagram

20.3 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 the Power Management Module. Table 3-1 shows the general purpose timer module operation in the low-power modes, and shows how this module may facilitate exit from each mode.

Table 20-1. 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.

20.4 Signal Description

Table 20-2 provides an overview of the signal properties.

NOTE

Throughout this section, an "*n*" in the pin name, as in "GPT*n*0," designates GPTA or GPTB.

Table 20-2. Signal Properties

Pin Name	GPTPORT Register Bit	Function	Reset State	Pull-up
GPT*n*0	PORTT*n*0	GPT*n* channel 0 IC/OC pin	Input	Active
GPT*n*1	PORTT*n*1	GPT*n* channel 1 IC/OC pin	Input	Active
GPT*n*2	PORTT*n*2	GPT*n* channel 2 IC/OC pin	Input	Active
GPT*n*3	PORTT*n*3	GPT*n* channel 3 IC/OC or PA pin	Input	Active
SYNC*n*	PORTE[3:0][1]	GPT*n* counter synchronization	Input	Active

[1] SYNCA is available on either PORTE3 or PORTE1; SYNCB is available on either PORTE2 or PORTE0.

20.4.1 GPT*n*[2:0]

The GPT*n*[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.

20.4.2 GPT*n*3

The GPT*n*3 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.

20.4.3 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.

20.5 Memory Map and Registers

See Table 20-3 for a memory map of the two GPT modules. GPTA has a base address of IPSBAR + 0x1A_0000. GPTB has a base address of IPSBAR + 0x1B_0000.

NOTE

Reading reserved or unimplemented locations returns zeroes. Writing to reserved or unimplemented locations has no effect.

Table 20-3. GPT Modules Memory Map

IPSBAR Offset		Bits 7–0	Access[1]
GPTA	GPTB		
0x1A_0000	0x1B_0000	GPT IC/OC Select Register (GPTIOS)	S
0x1A_0001	0x1B_0001	GPT Compare Force Register (GPTCFORC)	S
0x1A_0002	0x1B_0002	GPT Output Compare 3 Mask Register (GPTOC3M)	S
0x1A_0003	0x1B_0003	GPT Output Compare 3 Data Register (GPTOC3D)	S
0x1A_0004	0x1B_0004	GPT Counter Register (GPTCNT)	S
0x1A_0006	0x1B_0006	GPT System Control Register 1 (GPTSCR1)	S
0x1A_0007	0x1B_0007	Reserved[2]	—
0x1A_0008	0x1B_0008	GPT Toggle-on-Overflow Register (GPTTOV)	S
0x1A_0009	0x1B_0009	GPT Control Register 1 (GPTCTL1)	S
0x1A_000A	0x1B_000a	Reserved(2)	—
0x1A_000B	0x1B_000b	GPT Control Register 2 (GPTCTL2)	S
0x1A_000C	0x1B_000c	GPT Interrupt Enable Register (GPTIE)	S
0x1A_000D	0x1B_000d	GPT System Control Register 2 (GPTSCR2)	S
0x1A_000E	0x1B_000e	GPT Flag Register 1 (GPTFLG1)	S
0x1A_000F	0x1B_000f	GPT Flag Register 2 (GPTFLG2)	S
0x1A_0010	0x1B_0010	GPT Channel 0 Register High (GPTC0H)	S
0x1A_0011	0x1Bb_0011	GPT Channel 0 Register Low (GPTC0L)	S
0x1A_0012	0x1B_0012	GPT Channel 1 Register High (GPTC1H)	S
0x1A_0013	0x1B_0013	GPT Channel 1 Register Low (GPTC1L)	S
0x1A_0014	0x1B_0014	GPT Channel 2 Register High (GPTC2H)	S
0x1A_0015	0x1B_0015	GPT Channel 2 Register Low (GPTC2L)	S
0x1A_0016	0x1B_0016	GPT Channel 3 Register High (GPTC3H)	S

Table 20-3. GPT Modules Memory Map (continued)

IPSBAR Offset		Bits 7–0	Access[1]
GPTA	GPTB		
0x1A_0017	0x1B_0017	GPT Channel 3 Register Low (GPTC3L)	S
0x1A_0018	0x1B_0018	Pulse Accumulator Control Register (GPTPACTL)	S
0x1A_0019	0x1B_0019	Pulse Accumulator Flag Register (GPTPAFLG)	S
0x1A_001A	0x1B_001A	Pulse Accumulator Counter Register High (GPTPACNTH)	S
0x1A_001B	0x1B_001B	Pulse Accumulator Counter Register Low (GPTPACNTL)	S
0x1A_001C	0x1B_001C	Reserved[(2)]	—
0x1A_001D	0x1B_001D	GPT Port Data Register (GPTPORT)	S
0x1A_001E	0x1B_001E	GPT Port Data Direction Register (GPTDDR)	S
0x1A_001F	0x1B_001F	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.

20.5.1 GPT Input Capture/Output Compare Select Register (GPTIOS)

	7 4	3 0
Field	—	IOS
Reset	0000_0000	
R/W	R/W	
Address	IPSBAR + 0x401A_0000, 0x401B_0000	

Figure 20-2. GPT Input Capture/Output Compare Select Register (GPTIOS)

Table 20-4. 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

20.5.2 GPT Compare Force Register (GPCFORC)

	7–4	3–0
Field	—	FOC
Reset	0000_0000	
R/W	R/W	
Address	IPSBAR + 0x1A_00001, 0x1B_0001	

Figure 20-3. GPT Input Compare Force Register (GPCFORC)

Table 20-5. 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.

20.5.3 GPT Output Compare 3 Mask Register (GPTOC3M)

	7–4	3–0
Field	—	OC3M
Reset	0000_0000	
R/W	R/W	
Address	IPSBAR + 0x1A_0002, 0x1B_0002	

Figure 20-4. GPT Output Compare 3 Mask Register (GPTOC3M)

Table 20-6. 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

20.5.4 GPT Output Compare 3 Data Register (GPTOC3D)

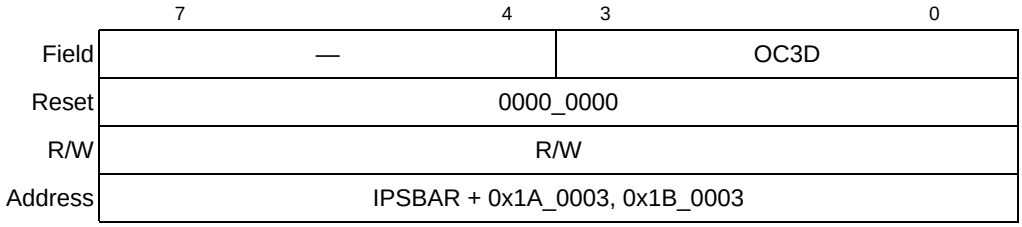


Figure 20-5. GPT Output Compare 3 Data Register (GPTOC3D)

Table 20-7. 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.

20.5.5 GPT Counter Register (GPTCNT)

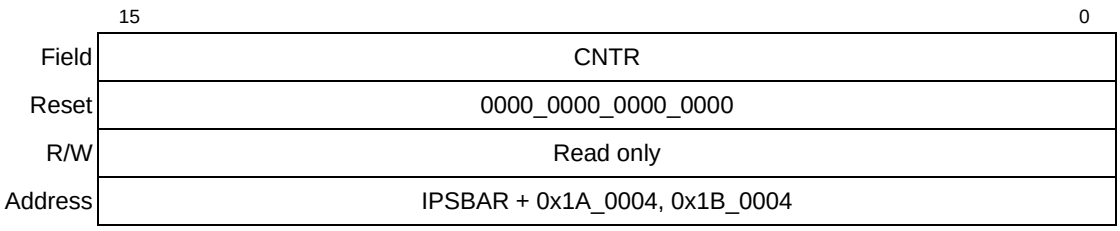


Figure 20-6. GPT Counter Register (GPTCNT)

Table 20-8. 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.

20.5.6 GPT System Control Register 1 (GPTSCR1)

	7	6	5	4	3	0
Field	GPTEN	—		TFFCA	—	
Reset	0000_0000					
R/W	R/W					
Address	IPSBAR + 0x1A_0006, 0x1B_0006					

Figure 20-7. GPT System Control Register 1 (GPTSCR1)

Table 20-9. 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 20-8. 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.

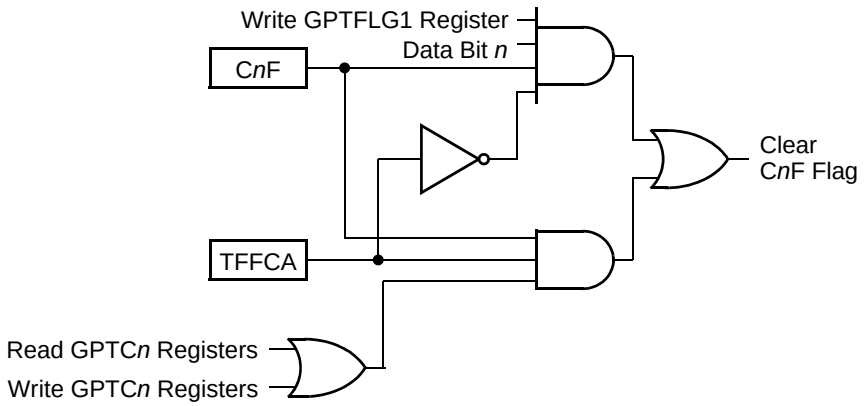


Figure 20-8. Fast Clear Flag Logic

20.5.7 GPT Toggle-On-Overflow Register (GPTTOV)

	7	6	5	4	3	2	1	0
Field	—				TOV			
Reset	0000_0000							
R/W	R/W							
Address	IPSBAR + 0x1A_0008, 0x1B_0008							

Figure 20-9. GPT Toggle-On-Overflow Register (GPTTOV)

Table 20-10. GPTTOV Field Descriptions

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

20.5.8 GPT Control Register 1 (GPTCTL1)

	7	6	5	4	3	2	1	0
Field	OM3	OL3	OM2	OL2	OM1	OL1	OM0	OL0
Reset	0000_0000							
R/W	R/W							
Address	IPSBAR + 0x1A_0009, 0x1B_0009							

Figure 20-10. GPT Control Register 1 (GPTCTL1)

Table 20-11. 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.

20.5.9 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 + 0x1A_000B, 0x1B_000B							

Figure 20-11. GPT Control Register 2 (GPTCTL2)

Table 20-12. 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)

20.5.10 GPT Interrupt Enable Register (GPTIE)

	7	6	5	4	3	0
Field	—				CI	
Reset	0000_0000					
R/W	R/W					
Address	IPSBAR + 0x1A_000C, 0x1B_000C					

Figure 20-12. GPT Interrupt Enable Register (GPTIE)

Table 20-13. 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

20.5.11 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 + 0x1A_000D, 0x1B_000D						

Figure 20-13. GPT System Control Register 2 (GPTSCR2)

Table 20-14. 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 20-14. 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.)

20.5.12 GPT Flag Register 1 (GPTFLG1)

	7	6	5	4	3	0
Field	—				CF	
Reset	0000_0000					
R/W	R/W					
Address	IPSBAR + 0x1A_000E, 0x1B_000E					

Figure 20-14. GPT Flag Register 1 (GPTFLG1)

Table 20-15. 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.

20.5.13 GPT Flag Register 2 (GPTFLG2)

	7	6	5	4	3	0
Field	TOF	—				
Reset	0000_0000					
R/W	R/W					
Address	IPSBAR + 0x1A_000F, 0x1B_000F					

Figure 20-15. GPT Flag Register 2 (GPTFLG2)

Table 20-16. 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.
6–0	—	Reserved, should be cleared.

Note: When the fast flag clear all bit, GPTSCR1[TFFCA], is set, any access to the GPT counter registers clears GPT flag register 2.

20.5.14 GPT Channel Registers (GPTC*n*)

	15 … 0
Field	CCNT
Reset	0000_0000_0000_0000
R/W	R/W
Address	IPSBAR + 0x1A_0010, 0x1A_0012, 0x1A_0014, 0x1A_0016, 0x1B_0010, 0x1B_0012, 0x1B_0014, 0x1B_0016

Figure 20-16. GPT Channel[0:3] Register (GPTC*n*)

Table 20-17. 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.

20.5.15 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 + 0x1A_0018, 0x1B_0018						

Figure 20-17. Pulse Accumulator Control Register (GPTPACTL)

Table 20-18. 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 $\overline{\text{RSTI}}$ 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 20-18. 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

20.5.16 Pulse Accumulator Flag Register (GPTPAFLG)

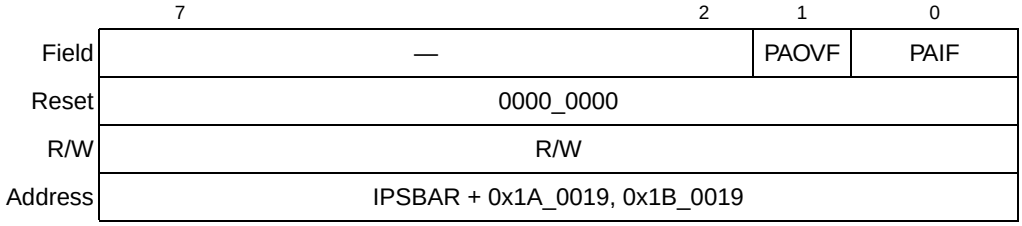


Figure 20-18. Pulse Accumulator Flag Register (GPTPAFLG)

Table 20-19. 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.

20.5.17 Pulse Accumulator Counter Register (GPTPACNT)

	15	0
Field	PACNT	
Reset	0000_0000_0000_0000	
R/W	R/W	
Address	IPSBAR + 0x1A_001A, 0x1B_001B	

Figure 20-19. Pulse Accumulator Counter Register (GPTPACNT)

Table 20-20. 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.

20.5.18 GPT Port Data Register (GPTPORT)

	7	6	5	4	3	0
Field	—				PORTT	
Reset	0000_0000					
R/W	R/W					
Address	IPSBAR + 0x1A_001D, 0x1B_001D					

Figure 20-20. GPT Port Data Register (GPTPORT)

Table 20-21. 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 PORTT*n* bit is 0, read pin driver state when corresponding GPTDDR bit is 1), write anytime.

20.5.19 GPT Port Data Direction Register (GPTDDR)

	7	6	5	4	3	2	1	0
Field	—				DDRT			
GPT Function	—				IC/OC			
Pulse Accumulator Function	—				PAI	—		
Reset	0000_0000							
R/W	R/W							
Address	IPSBAR + 0x1A_001E, 0x1B_001E							

Figure 20-21. GPT Port Data Direction Register (GPTDDR)

Table 20-22. 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

20.6 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.

20.6.1 Prescaler

The prescaler divides the module clock by 1, 2, 4, 8, 16, 32, 64, or 128. The GPTSCR2[PR] bits select the prescaler divisor.

20.6.2 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.

20.6.3 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.

20.6.4 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.

20.6.5 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.

20.6.6 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.

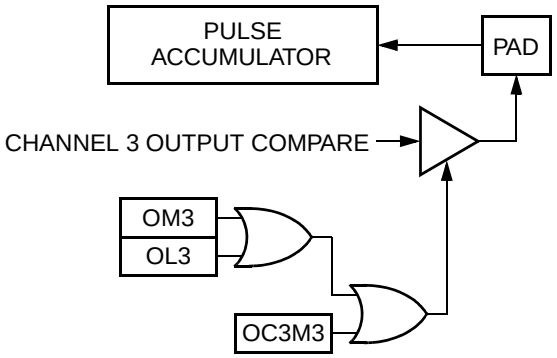


Figure 20-22. Channel 3 Output Compare/Pulse Accumulator Logic

20.6.7 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 20-23 shows how various timer settings affect pin functionality.

Table 20-23. 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
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)

Table 20-23. GPT Settings and Pin Functions (continued)

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.

20.7 Reset

Reset initializes the GPT registers to a known startup state as described in Section 20.5, "Memory Map and Registers."

20.8 Interrupts

Table 20-24 lists the interrupt requests generated by the timer.

Table 20-24. 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

20.8.1 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

20.8.2 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.

20.8.3 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.

20.8.4 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 21
DMA Timers (DTIM0–DTIM3)

21.1 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* appears throughout this section to refer to registers or signals associated with one of the four identical timer modules: DTIM0, DTIM1, DTIM2, or DTIM3.

21.1.1 Overview

Each DMA timer module has a separate register set for configuration and control. The timers can be configured to operate from the internal bus clock or from an external clocking source using the DTIN*n* signal. If the internal bus 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 request 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 26, “General Purpose I/O Module”) prior to configuring the DMA Timers.

Figure 21-1 is a block diagram of one of the four identical timer modules.

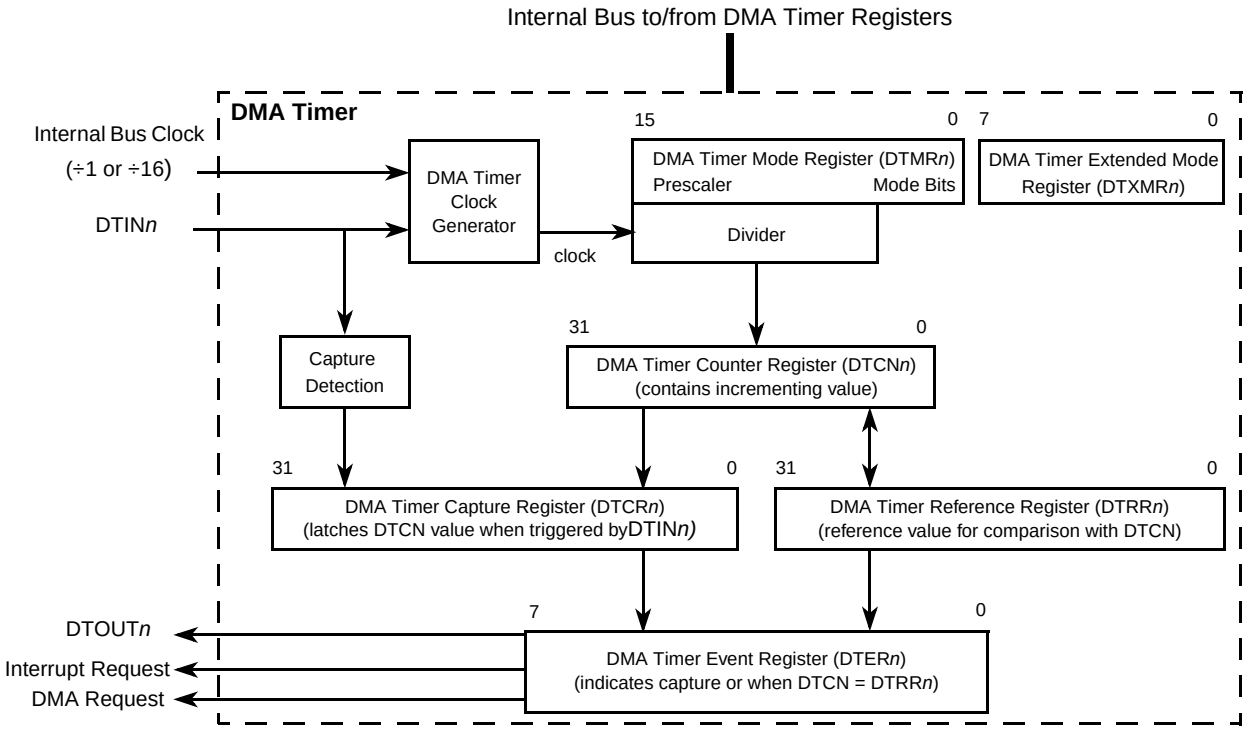


Figure 21-1. DMA Timer Block Diagram

21.1.2 Features

Each DMA timer module has:

- Maximum timeout period of 219,902 seconds at 80 MHz (~61 hours)
- 12.5-ns resolution at 80 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

21.2 Memory Map/Register Definition

The timer module registers, shown in Table 21-1, can be modified at any time.

Table 21-1. DMA Timer Module Memory Map

IPSBAR Offset DMA Timer 0 DMA Timer 1 DMA Timer 2 DMA Timer 3	Register	Width (bits)	Access	Reset Value	Section/Page
0x00_0400 0x00_0440 0x00_0480 0x00_04C0	DMA Timer *n* Mode Register (DTMR*n*)	16	R/W	0x0000	21.2.1/21-3
0x00_0402 0x00_0442 0x00_0482 0x00_04C2	DMA Timer *n* Extended Mode Register (DTXMR*n*)	8	R/W	0x00	21.2.2/21-5
0x00_0403 0x00_0443 0x00_0483 0x00_04C3	DMA Timer *n* Event Register (DTER*n*)	8	R/W	0x00	21.2.3/21-5
0x00_0404 0x00_0444 0x00_0484 0x00_04C4	DMA Timer *n* Reference Register (DTRR*n*)	32	R/W	0xFFFF_FFFF	21.2.4/21-7
0x00_0408 0x00_0448 0x00_0488 0x00_04C8	DMA Timer *n* Capture Register (DTCR*n*)	32	R/W	0x0000_0000	21.2.5/21-7
0x00_040C 0x00_044C 0x00_048C 0x00_04CC	DMA Timer *n* Counter Register (DTCN*n*)	32	R	0x0000_0000	21.2.6/21-8

21.2.1 DMA Timer Mode Registers (DTMR*n*)

The DTMR*n* registers program the prescaler and various timer modes.

IPSBAR Offset: 0x00_0400 (DTMR0)
0x00_0440 (DTMR1)
0x00_0480 (DTMR2)
0x00_04C0 (DTMR3)

Access: User read/write

	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 21-2. DTMR*n* Registers

Table 21-2. DTMR*n* Field Descriptions

Field	Description
15–8 PS	Prescaler value. Divides the clock input (internal bus clock/(16 or 1) or clock on DTIN*n*) 0x00 1 ... 0xFF 256
7–6 CE	Capture edge. 00 Disable capture event output. Timer in reference mode. 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 internal bus clock cycle (12.5-ns resolution at 80 MHz) 1 Toggle output.
4 ORRI	Output reference request, interrupt enable. If ORRI is set when DTER*n*[REF] is set, a DMA request or an interrupt occurs, depending on the value of DTXMR*n*[DMAEN] (DMA request if set, interrupt if cleared). 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.
3 FRR	Free run/restart 0 Free run. Timer count continues incrementing 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. Avoid setting CLK when RST is already set. Doing so causes CLK to zero (stop counting). 00 Stop count 01 Internal bus clock divided by 1 10 Internal bus clock divided by 16. This clock source is not synchronized with the timer; therefore, successive time-outs may vary slightly. 11 DTIN*n* pin (falling edge)
0 RST	Reset timer. Performs a software timer reset similar to an external reset, although other register values can be written while RST is cleared. 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

21.2.2 DMA Timer Extended Mode Registers (DTXMR*n*)

The DTXMR*n* registers program 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 21-3. DTXMR*n* Registers

Table 21-3. 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, must be cleared.
0 MODE16	Selects the increment mode for the timer. Setting MODE16 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 remain compared to the reference value. 0 Increment timer by 1 1 Increment timer by 65,537

21.2.3 DMA Timer Event Registers (DTER*n*)

DTER*n*, shown in Figure 21-4, reports capture or reference events by setting DTER*n*[CAP] or DTER*n*[REF]. This reporting happens regardless of the corresponding DMA request or interrupt enable values, DTXMR*n*[DMAEN] and DTMR*n*[ORRI,CE].

Writing a 1 to 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, clear REF and CAP 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, processing of the DMA data transfer automatically clears 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 21-4. DTER*n* Registers

Table 21-4. DTER*n* Field Descriptions

Field	Description
7–2	Reserved, must be cleared.
1 REF	Output reference event. The counter value (DTCN*n*) equals DTRR*n*. Writing a 1 to REF clears the event condition. Writing a 0 has no effect.
0 CAP	Capture event. The counter value has been latched into DTCR*n*. Writing a 1 to CAP clears the event condition. Writing a 0 has no effect.

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

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 and trigger interrupt
1	01	1	Capture on rising edge and trigger DMA
1	10	0	Capture on falling edge and trigger interrupt
1	10	1	Capture on falling edge and trigger DMA
1	11	0	Capture on any edge and trigger interrupt
1	11	1	Capture on any edge and trigger DMA

21.2.4 DMA Timer Reference Registers (DTRR*n*)

As part of the output-compare function, each DTRR*n* contains the reference value compared with the respective free-running timer counter (DTCN*n*).

The reference value is matched when DTCN*n* equals DTRR*n*. The prescaler indicates that DTCN*n* should be incremented again. Therefore, 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 (32-bit reference value)																															
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 21-5. DTRR*n* Registers

Table 21-5. DTRR*n* Field Descriptions

Field	Description
31–0 REF	Reference value compared with the respective free-running timer counter (DTCN*n*) as part of the output-compare function.

21.2.5 DMA Timer Capture Registers (DTCR*n*)

Each DTCR*n* latches the corresponding DTCN*n* value during a capture operation when an edge occurs on DTIN*n*, as programmed in DTMR*n*. The internal bus clock is assumed to be the clock source. DTIN*n* cannot simultaneously function as a clocking source and as an input capture pin. Indeterminate operation results if DTIN*n* 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 21-6. DTCR*n* Registers

Table 21-6. DTCR*n* Field Descriptions

Field	Description
31–0 CAP	Captures the corresponding DTCN*n* value during a capture operation when an edge occurs on DTIN*n*, as programmed in DTMR*n*.

21.2.6 DMA Timer Counters (DTCN*n*)

The current value of the 32-bit timer counter can be read at anytime without affecting counting. Writes to DTCN*n* clear the timer counter. The timer counter increments on the clock source rising edge (internal bus clock divided by 1, internal bus clock divided by 16, or DTIN*n*).

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 W	CNT (32-bit timer counter value count)																															
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 21-7. DMA Timer Counters (DTCN*n*)

Table 21-7. DTCN*n* Field Descriptions

Field	Description
31–0 CNT	Timer counter. Can be read at anytime without affecting counting and any write to this field clears it.

21.3 Functional Description

21.3.1 Prescaler

The prescaler clock input is selected from the internal bus clock (f_{sys} divided by 1 or 16) or from the corresponding timer input, DTIN*n*. DTIN*n* is synchronized to the internal bus clock, and the synchronization delay is between two and three internal bus clocks. The corresponding DTMR*n*[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, DTCN*n*.

21.3.2 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 DTIN*n* 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] and DTXMR*n*[DMAEN] are set, a DMA request is asserted. If DTER*n*[CAP] is set and DTXMR*n*[DMAEN] is cleared, an interrupt is asserted.

21.3.3 Reference Compare

Each DMA timer can be configured to count up to a reference value. If the reference value is met, DTER*n*[REF] is set.

- If DTMR*n*[ORRI] is set and DTXMR*n*[DMAEN] is cleared, an interrupt is asserted.
- If DTMR*n*[ORRI] and DTXMR*n*[DMAEN] are set, 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.

21.3.4 Output Mode

When a timer reaches the reference value selected by DTRR, it can send an output signal on DTOUT*n*. DTOUT*n* can be an active-low pulse or a toggle of the current output, as selected by the DTMR*n*[OM] bit.

21.4 Initialization/Application Information

The general-purpose timer modules typically, but not necessarily, follow this program order:

- 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 DTIN*n*
  - Configure DTOUT*n* 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.
  - Internal bus clock (can be divided by 1 or 16)
  - DTIN*n*, the maximum value of DTIN*n* is 1/5 of the internal bus clock, as described in the device's electrical characteristics

NOTE

DTIN*n* may not be configured as a clock source when the timer capture mode is selected or indeterminate operation results.

- The 8-bit DTMR*n*[PS] prescaler value is set.
- Using DTMR*n*[RST], counter is cleared and started.
- Timer events are managed with an interrupt service routine, a DMA request, or by a software polling mechanism.

21.4.1 Code Example

The following code provides an example of how to initialize and use DMA Timer0 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,     internal bus 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
```

21.4.2 Calculating Time-Out Values

Equation 21-1 determines time-out periods for various reference values:

$$\text{Timeout period} = (1/\text{clock frequency}) \times (1 \text{ or } 16) \times (\text{DTMR}n[\text{PS}] + 1) \times (\text{DTRR}n[\text{REF}] + 1) \qquad \textit{Eqn. 21-1}$$

When calculating time-out periods, add one to the prescaler to simplify calculating, because DTMR*n*[PS] equal to 0x00 yields a prescaler of one, and DTMR*n*[PS] equal to 0xFF yields a prescaler of 256.

For example, if a 80-MHz timer clock is divided by 16, DTMR*n*[PS] equals 0x7F, and the timer is referenced at 0x1312C (78,124 decimal), the time-out period is:

$$\text{Timeout period} = \frac{1}{80 \times 10^6} \times 16 \times (127 + 1) \times (78124 + 1) = 2.00 \text{ seconds} \qquad \textit{Eqn. 21-2}$$

Chapter 22
Queued Serial Peripheral Interface (QSPI)

22.1 Introduction

This chapter describes the queued serial peripheral interface (QSPI) module.

22.1.1 Block Diagram

Figure 22-1 illustrates the QSPI module.

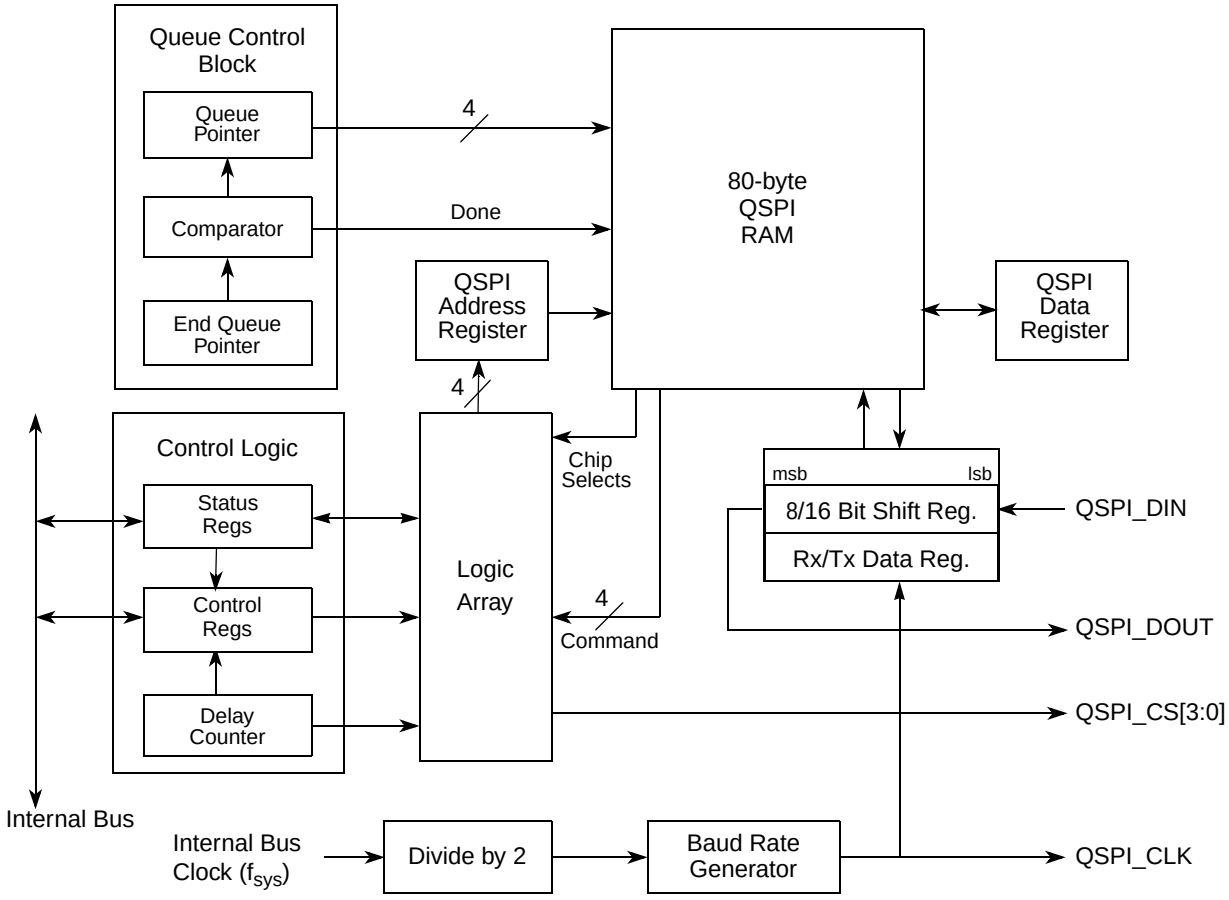


Figure 22-1. QSPI Block Diagram

22.1.2 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 26, “General Purpose I/O Module”) prior to configuring the QSPI module.

22.1.3 Features

Features include:

- Programmable queue to support up to 16 transfers without user intervention
  - 80 bytes of data storage provided
- Supports transfer sizes of 8 to 16 bits in 1-bit increments
- Four peripheral chip-select lines for control of up to 15 devices (All chip selects may not be available on all devices. See Chapter 14, “Signal Descriptions,” for details on which chip-selects are pinned-out.)
- Baud rates from 156.9 Kbps to 20 Mbps at 80 MHz internal bus frequency
- Programmable delays before and after transfers
- Programmable QSPI clock phase and polarity
- Supports wraparound mode for continuous transfers

22.1.4 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. If the master bit is not set, QSPI activity is indeterminate. The QSPI can initiate serial transfers but cannot respond to transfers initiated by other QSPI masters.

22.2 External Signal Description

The module provides access to as many as 15 devices with a total of seven signals: QSPI_DOUT, QSPI_DIN, QSPI_CLK, QSPI_CS[3:0].

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 when a command in the queue is executed. More than one chip-select signal can be asserted simultaneously.

Although QSPI_CS*n* signals 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 22-1. QSPI Input and Output Signals and Functions

Signal Name	Hi-Z or Actively Driven	Function
Data output (QSPI_DOUT)	Configurable	Serial data output from 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 from QSPI

22.3 Memory Map/Register Definition

Table 22-2 is the QSPI register memory map. Reading reserved locations returns zeros.

Table 22-2. QSPI Memory Map

IPSBAR Offset[1]	Register	Width (bits)	Access	Reset Value	Section/Page
0x00_0340	QSPI Mode Register (QMR)	16	R/W	0x0104	22.3.1/22-3
0x00_0344	QSPI Delay Register (QDLYR)	16	R/W	0x0404	22.3.2/22-5
0x00_0348	QSPI Wrap Register (QWR)	16	R/W[2]	0x0000	22.3.3/22-6
0x00_034C	QSPI Interrupt Register (QIR)	16	R/W[2]	0x0000	22.3.4/22-6
0x00_0350	QSPI Address Register (QAR)	16	R/W[2]	0x0000	22.3.5/22-7
0x00_0354	QSPI Data Register (QDR)	16	R/W	0x0000	22.3.6/22-8

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.

22.3.1 QSPI Mode Register (QMR)

The QMR, shown in Figure 22-2, 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.

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 Offset: 0x00_0340 (QMR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MSTR	0	BITS				CPOL	CPHA	BAUD							
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0

Figure 22-2. QSPI Mode Register (QMR)

Table 22-3. QMR Field Descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>15
MSTR</td><td>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.</td></tr>
<tr><td>14</td><td>Reserved, must be cleared.</td></tr>
<tr><td>13–10
BITS</td><td>Transfer size. Determines the number of bits to be transferred for each entry in the queue.
<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></td></tr>
<tr><td>9
CPOL</td><td>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.</td></tr>
<tr><td>8
CPHA</td><td>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.</td></tr>
<tr><td>7–0
BAUD</td><td>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 × [desired QSPI_CLK baud rate])</td></tr>
</table>

Figure 22-3 shows an example of a QSPI clocking and data transfer.

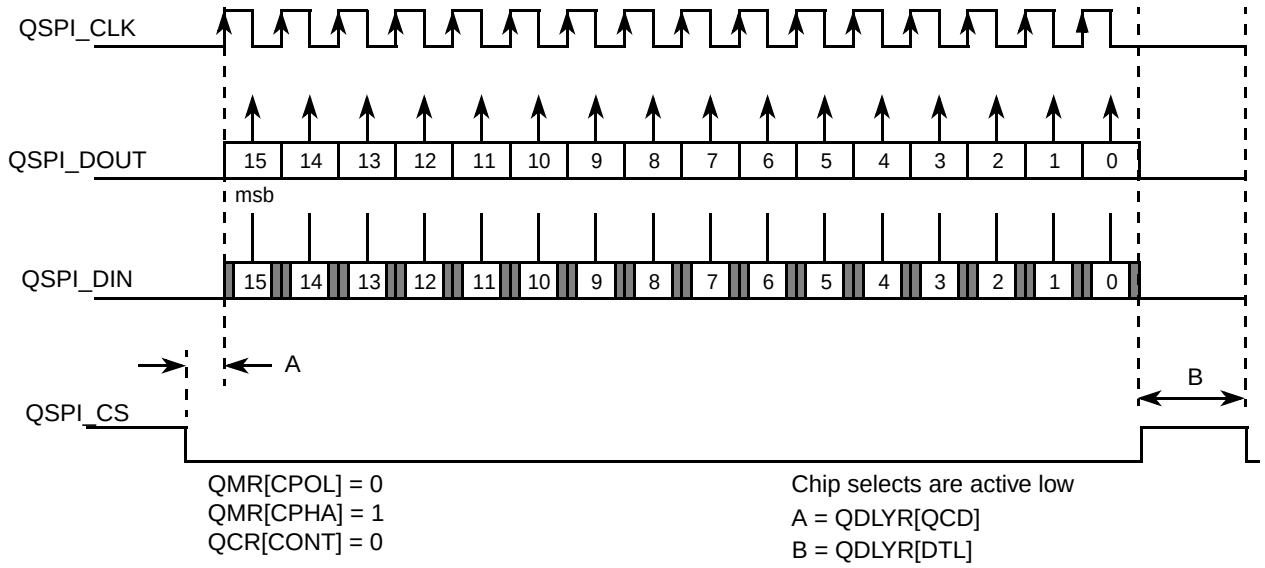


Figure 22-3. QSPI Clocking and Data Transfer Example

22.3.2 QSPI Delay Register (QDLYR)

The QDLYR is used to initiate master mode transfers and to set various delay parameters.

IPSBAR Offset: 0x00_0344 (QDLYR) Access: User read/write

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

Figure 22-4. QSPI Delay Register (QDLYR)

Table 22-4. 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. The QSPI clears this bit automatically 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 22.4.3, “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.

22.3.3 QSPI Wrap Register (QWR)

The QSPI wrap register provides halt transfer control, wraparound settings, and queue pointer locations.

IPSBAR Offset: 0x00_0348 (QWR) Access: User read/write

	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 22-5. QSPI Wrap Register (QWR)

Table 22-5. QWR Field Descriptions

Field	Description
15 HALT	Halt transfers. Assertion of this bit causes the QSPI to stop execution of commands after 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.

22.3.4 QSPI Interrupt Register (QIR)

The QIR contains QSPI interrupt enables and status flags.

IPSBAR Offset: 0x00_034C (QIR) Access: User read/write

	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 22-6. QSPI Interrupt Register (QIR)

Table 22-6. 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, must 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, must be cleared.
8 SPIFE	QSPI finished (SPIF) interrupt enable. 0 SPIF interrupt disabled 1 SPIF interrupt enabled
7–4	Reserved, must 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 (w1c) 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 completion of the command queue by the QSPI. Writing a 1 to this bit (w1c) clears it and writing 0 has no effect.
1	Reserved, must 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 (w1c) clears it and writing 0 has no effect.

22.3.5 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 22.4.1, “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. (These addresses refer to the QSPI RAM space, not the device memory map.)

NOTE

A read or write to the QSPI RAM causes QAR to increment. However, the QAR does not wrap after the last queue entry within each section of the RAM. The application software must manage address range errors.

IPSBAR Offset: 0x00_0350 (QAR) Access: User read/write

	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 22-7. QSPI Address Register (QAR)

Table 22-7. QAR Field Descriptions

Field	Description
15–6	Reserved, must be cleared.
5–0 ADDR	Address used to read/write the QSPI RAM. Ranges are as follows: 0x00–0x0F Transmit RAM 0x10–0x1F Receive RAM 0x20–0x2F Command RAM 0x30–0x3F Reserved

22.3.6 QSPI Data Register (QDR)

The QDR is used to access QSPI RAM indirectly. The CPU reads and writes all data from and to the QSPI RAM through this register.

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.

IPSBAR Offset: 0x00_0354 (QDR) Access: User read/write

	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 22-8. QSPI Data Register (QDR)

Table 22-8. QDR Field Descriptions

Field	Description
15–0 DATA	A write to this field causes data to be written to the QSPI RAM entry specified by QAR[ADDR]. Similarly, a read of this field returns the data in the QSPI RAM at the address specified by QAR[ADDR]. During command RAM accesses (QAR[ADDR] = 0x20–0x2F), only the most significant byte of this field is used.

22.3.7 Command RAM Registers (QCR0–QCR15)

The command RAM is accessed using the upper byte of the QDR; the QSPI cannot modify information in command RAM. 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].

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 22-9. Command RAM Registers (QCR0–QCR15)

Table 22-9. 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:** To keep the chip selects asserted for transfers 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.
11–8 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. Bits 11-8 map directly to the corresponding QSPI_CS*n* pins. If more than four chip selects are needed, then an external demultiplexor can be used with the QSPI_CS*n* pins. 0 Enable chip select. 1 Mask chip select. **Note:** Not all chip selects may be available on all device packages. See Chapter 14, “Signal Descriptions,” for details on which chip selects are pinned-out.
7–0	Reserved, must be cleared.

22.4 Functional Description

The QSPI uses a dedicated 80-byte block of static RAM accessible to the module and CPU to perform queued operations. The RAM is divided into three segments:

- 16 command control bytes (command RAM)
- 32 transmit data bytes (transmit data RAM)

- 32 receive data bytes (receive data 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, the term 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):

- New queue pointer (QWR[NEWQP])—points to the first command in the queue
- Internal queue pointer—points to the command currently being executed
- Completed queue pointer (QWR[CPTQP])—points to the last command executed
- 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]).

22.4.1 QSPI RAM

The QSPI contains an 80-byte block of static RAM that can be accessed by 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 data and command RAM are user write-only. The receive RAM is user read-only. Figure 22-10 shows the RAM configuration. The RAM contents are undefined immediately after a reset.

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] and causes the value in QAR to increment. Correspondingly, a read from 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 22-10. QSPI RAM Model

22.4.1.1 Receive RAM

Data received by the QSPI is stored in the receive RAM segment located at 0x10 to 0x1F in the QSPI RAM space. Read 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. Receive RAM is not writeable.

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.

22.4.1.2 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.

22.4.1.3 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, each 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 that sequence the following actions:

- Chip-select pins are activated.
- Data is transmitted from the 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 the 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.

22.4.2 Baud Rate Selection

The maximum QSPI clock frequency is one-fourth the clock frequency of the internal bus clock (f_{sys}). 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. Table 22-10 shows the QSPI_CLK frequency as a function of internal bus clock and baud rate.

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:

$$\text{QMR[BAUD]} = \frac{f_{sys}}{2 \times [\text{desired QSPI\_CLK baud rate}]} \qquad \textit{Eqn. 22-1}$$

Table 22-10. QSPI_CLK Frequency as Function of Internal Bus Clock and Baud Rate

Internal Bus Clock = 80 MHz	
QMR [BAUD]	**QSPI_CLK**
2	20 MHz
4	10 MHz
8	5 MHz
16	2.5 MHz
32	1.25 Hz
255	156.9 kHz

22.4.3 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 the 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:

$$\text{QSPI\_CS-to-QSPI\_CLK delay} = \frac{\text{QDLYR[QCD]}}{f_{sys}} \qquad \textit{Eqn. 22-2}$$

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 equals 1:

$$\text{Delay after transfer} = \frac{32 \times \text{QDLYR[DTL]}}{f_{sys}} \quad (\text{DT} = 1) \qquad \textit{Eqn. 22-3}$$

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:

$$\text{Standard delay after transfer} = \frac{17}{f_{sys}} \quad (DT = 0) \qquad \textit{Eqn. 22-4}$$

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.

22.4.4 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] indicates the required number of bits to be transferred, with the default value of 16 bits.

22.4.5 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*n* signals are asserted between transfers. When CONT is cleared, QSPI_CS*n* are negated between transfers. The QSPI_CS*n* 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.

22.5 Initialization/Application Information

The following steps are necessary to set up the QSPI 12-bit data transfers and a QSPI_CLK of 5 MHz. The QSPI RAM is set up for a queue of 16 transfers. All four 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 5 MHz (assuming a 80-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 23
UART Modules

23.1 Introduction

This chapter describes the use of the three universal asynchronous receiver/transmitters (UARTs) and includes programming examples.

NOTE

The designation *n* appears throughout this section to refer to registers or signals associated with one of the three identical UART modules: UART0, UART1, or UART2.

23.1.1 Overview

The internal bus clock can clock each of the three independent UARTs, eliminating the need for an external UART clock. As Figure 23-1 shows, each UART module interfaces directly to the CPU and consists of:

- Serial communication channel
- Programmable clock generation
- Interrupt control logic and DMA request logic
- Internal channel control logic

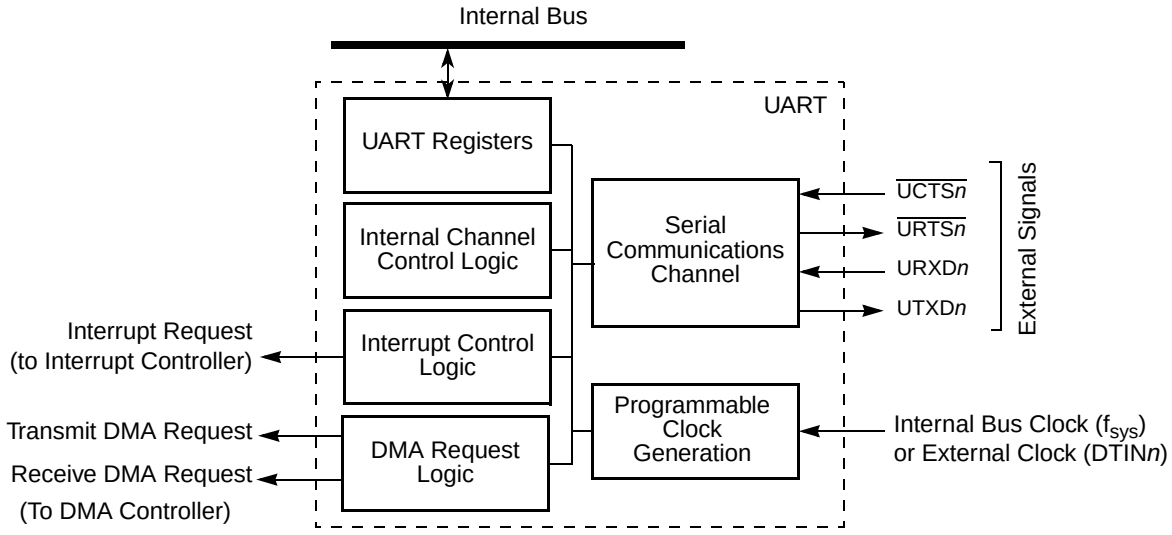


Figure 23-1. UART Block Diagram

NOTE

The DTIN*n* pin can clock UART*n*. However, if the timers are operating and the UART uses DTIN*n* as a clock source, 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 transmitter serial data output (UTXD*n*). See Section 23.4.2.1, "Transmitter."

The receiver converts serial data from the receiver serial data input (URXD*n*) 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 23.4.2.2, "Receiver."

NOTE

The GPIO module must be configured to enable the peripheral function of the appropriate pins (refer to Chapter 26, "General Purpose I/O Module") prior to configuring the UART module.

23.1.2 Features

The device contains three independent UART modules with:

- Each clocked by external clock or internal bus clock (eliminates need for an external UART clock)
- Full-duplex asynchronous/synchronous receiver/transmitter
- 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 serial channel programmable to normal (full-duplex), automatic echo, local loopback, or remote loopback 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

23.2 External Signal Description

Table 23-1 briefly describes the UART module signals.

Table 23-1. UART Module External Signals

Signal	Description
UTXD*n*	Transmitter Serial Data Output. UTXD*n* is held high (mark condition) when the transmitter is disabled, idle, or operating in the local loopback mode. Data is shifted out on UTXD*n* on the falling edge of the clock source, with the least significant bit (lsb) sent first.
URXD*n*	Receiver Serial Data Input. Data received on URXD*n* is sampled on the rising edge of the clock source, with the lsb received first.
$\overline{\text{UCTS}n}$	Clear-to- Send. This input can generate an interrupt on a change of state.
$\overline{\text{URTS}n}$	Request-to-Send. This output can be programmed to be negated or asserted automatically by the receiver or the transmitter. When connected to a transmitter's $\overline{\text{UCTS}n}$, $\overline{\text{URTS}n}$ can control serial data flow.

Figure 23-2 shows a signal configuration for a UART/RS-232 interface.

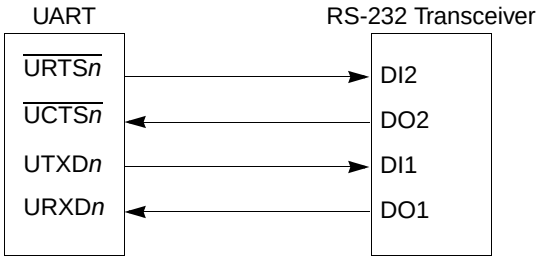


Figure 23-2. UART/RS-232 Interface

23.3 Memory Map/Register Definition

This section contains a detailed description of each register and its specific function. Flowcharts in Section 23.5, "Initialization/Application Information," describe basic UART module programming. Writing control bytes into the appropriate registers controls the operation of the UART module.

NOTE

UART registers are accessible only as bytes.

NOTE

Interrupt can mean an interrupt request asserted to the CPU or a DMA request.

Table 23-2. UART Module Memory Map

UART0 UART1 UART2	Register	Width (bit)	Access	Reset Value	Section/Page
0x00 0x0 0x0	UART Mode Registers[1] (UMR1*n*), (UMR2*n*)	8	R/W	0x00	23.3.1/23-5 23.3.2/23-6
0x04 0x4 0x4	UART Status Register (USR*n*)	8	R	0x00	23.3.3/23-8
	UART Clock Select Register[1](UCSR*n*)	8	W	See Section	23.3.4/23-9
0x08 0x8 0x8	UART Command Registers (UCR*n*)	8	W	0x00	23.3.5/23-9
0x0C 0xC 0xC	UART Receive Buffers (URB*n*)	8	R	0xFF	23.3.6/23-11
	UART Transmit Buffers (UTB*n*)	8	W	0x00	23.3.7/23-12
0x10 0x0 0x0	UART Input Port Change Register (UIPCR*n*)	8	R	See Section	23.3.8/23-12
	UART Auxiliary Control Register (UACR*n*)	8	W	0x00	23.3.9/23-13
0x14 0x4 0x4	UART Interrupt Status Register (UISR*n*)	8	R	0x00	23.3.10/23-13
	UART Interrupt Mask Register (UIMR*n*)	8	W	0x00	
0x18 0x8 0x8	UART Baud Rate Generator Register (UBG1*n*)	8	W[2]	0x00	23.3.11/23-15
0x1C 0xC 0xC	UART Baud Rate Generator Register (UBG2*n*)	8	W[2]	0x00	23.3.11/23-15
0x34 0x4 0x4	UART Input Port Register (UIP*n*)	8	R	0xFF	23.3.12/23-15
0x38 0x8 0x8	UART Output Port Bit Set Command Register (UOP1*n*)	8	W[2]	0x00	23.3.13/23-16
0x3C 0xC 0xC	UART Output Port Bit Reset Command Register (UOP0*n*)	8	W[2]	0x00	23.3.13/23-16

1 UMR1*n*, UMR2*n*, and UCSR*n* must be changed only after the receiver/transmitter is issued a software reset command. If 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.

23.3.1 UART Mode Registers 1 (UMR1*n*)

The UMR1*n* registers control UART module 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*.

IPSBAR 0x00_0200 (UMR10)
Offset: 0x00_0240 (UMR11)
0x00_0280 (UMR12)

Access: User read/write[1]

	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 23-3. UART Mode Registers 1 (UMR1*n*)

Table 23-3. UMR1*n* Field Descriptions

Field	Description
7 RXRTS	Receiver request-to-send. Allows the $\overline{\text{URTS}}n$ output to control the $\overline{\text{UCTS}}n$ input of the transmitting device to prevent receiver overrun. If the receiver and transmitter are incorrectly programmed for $\overline{\text{URTS}}n$ control, $\overline{\text{URTS}}n$ control is disabled for both. Transmitter RTS control is configured in UMR2*n*[TXRTS]. 0 The receiver has no effect on $\overline{\text{URTS}}n$. 1 When a valid start bit is received, $\overline{\text{URTS}}n$ is negated if the UART's FIFO is full. $\overline{\text{URTS}}n$ is reasserted when the FIFO has an empty position available.
6 RXIRQ/ FFULL	Receiver interrupt select. 0 RXRDY is the source generating interrupt or DMA requests. 1 FFULL is the source generating 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 UART was issued. See Section 23.3.5, "UART Command Registers (UCRn)."
4–3 PM	Parity mode. Selects the parity or multidrop mode for the UART. 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.

Table 23-3. UMR1*n* Field Descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>2
PT</td><td>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></td></tr>
<tr><td>1–0
B/C</td><td>Bits per character. Selects 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</td></tr>
</table>

23.3.2 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.

IPSBAR 0x00_0200 (UMR20)
Offset: 0x00_0240 (UMR21)
0x00_0280 (UMR22)

Access: User read/write[1]

	7	6	5	4	3	2	1	0
R W	CM		TXRTS	TXCTS	SB			
Reset:	0	0	0	0	0	0	0	0

1 After UMR1*n* is read or written, the pointer points to UMR2*n*

Figure 23-4. UART Mode Registers 2 (UMR2*n*)

Table 23-4. UMR2*n* Field Descriptions

Field	Description
7–6 CM	Channel mode. Selects a channel mode. Section 23.4.3, “Looping Modes,” describes individual modes. 00 Normal 01 Automatic echo 10 Local loopback 11 Remote loopback
5 TXRTS	Transmitter ready-to-send. Controls negation of $\overline{\text{URTS}n}$ to automatically terminate a message transmission. Attempting to program a receiver and transmitter in the same UART for $\overline{\text{URTS}n}$ control is not permitted and disables $\overline{\text{URTS}n}$ control for both. 0 The transmitter has no effect on $\overline{\text{URTS}n}$. 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 transmitter shift and holding registers are completely sent, including the programmed number of stop bits.
4 TXCTS	Transmitter clear-to-send. If TXCTS and TXRTS are set, TXCTS controls the operation of the transmitter. 0 $\overline{\text{UCTS}n}$ has no effect on the transmitter. 1 Enables clear-to-send operation. The transmitter checks the state of $\overline{\text{UCTS}n}$ each time it is ready to send a character. If $\overline{\text{UCTS}n}$ is asserted, the character is sent; if it is deasserted, the signal UTXD*n* remains in the high state and transmission is delayed until $\overline{\text{UCTS}n}$ is asserted. Changes in $\overline{\text{UCTS}n}$ as a character is being sent do not affect its transmission.
3–0 SB	Stop-bit length control. Selects length of 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, 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

23.3.3 UART Status Registers (USR*n*)

The USR*n* registers show the status of the transmitter, the receiver, and the FIFO.

IPSBAR 0x00_0204 (USR0) Access: User read-only
Offset: 0x00_0244 (USR1)
0x00_0284 (USR2)

	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 23-5. UART Status Registers (USR*n*)

Table 23-5. USR*n* Field Descriptions

Field	Description
7 RB	Received break. The received break circuit detects breaks originating 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 URXD*n* returns to the high state for at least one-half bit time, which equals two successive edges of the UART clock. RB is valid only when RXRDY is set.
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 is set.
5 PE	Parity error. Valid only if RXRDY is set. 0 No parity error occurred. 1 If UMR1*n*[PM] equals 0*x* (with parity or force parity), the corresponding character in the FIFO was received with incorrect parity. If UMR1*n*[PM] equals 11 (multidrop), PE stores the received address or data (A/D) bit. PE is valid only when RXRDY is set.
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. The RESET ERROR STATUS command in UCR*n* clears OE.
3 TEMP	Transmitter empty. 0 The transmit buffer is not empty. A character is shifted out, or the transmitter is disabled. The transmitter is enabled/disabled by programming UCR*n*[TC]. 1 The transmitter has underrun (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.

Table 23-5. USR*n* Field Descriptions (continued)

Field	Description
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.

23.3.4 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 23.4.1, “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.

IPSBAR 0x00_0204 (UCSR0) Access: User write-only
Offset: 0x00_0244 (UCSR1)
0x00_0284 (UCSR2)

	7	6	5	4	3	2	1	0
R								
W	RCS				TCS			
Reset:	See Note				See Note			

Note: The RCS and TCS reset values are set so the receiver and transmiter use the prescaled internal bus clock as their clock source.

Figure 23-6. UART Clock Select Registers (UCSR*n*)

Table 23-6. UCSR*n* Field Descriptions

Field	Description
7–4 RCS	Receiver clock select. Selects the clock source for the receiver. 1101 Prescaled internal bus clock (f_{sys}) 1110 DTIN*n* divided by 16 1111 DTIN*n*
3–0 TCS	Transmitter clock select. Selects the clock source for the transmitter. 1101 Prescaled internal bus clock (f_{sys}) 1110 DTIN*n* divided by 16 1111 DTIN*n*

23.3.5 UART Command Registers (UCR*n*)

The UCRs 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.

IPSBAR 0x00_0208 (UCR0)
Offset: 0x00_0248 (UCR1)
0x00_0288 (UCR2)

Access: User write-only

	7	6	5	4	3	2	1	0
R								
W	0	MISC			TC		RC	
Reset:	0	0	0	0	0	0	0	0

Figure 23-7. UART Command Registers (UCR*n*)

Table 23-7 describes UCR*n* fields and commands. Examples in Section 23.4.2, "Transmitter and Receiver Operating Modes," show how these commands are used.

Table 23-7. UCR*n* Field Descriptions

Field	Description
7	Reserved, must be cleared.
6–4 MISC	MISC Field (this field selects a single command)

	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 UTXD*n* low. If the transmitter is empty, break may be delayed up to one bit time. If the transmitter is active, break starts when character transmission completes. 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. Transmitter must be enabled for the command to be accepted. This command ignores the state of $\overline{\text{UCTS}}n$.
111	STOP BREAK	Causes UTXD*n* to go high (mark) within two bit times. Any characters in the transmit buffer are sent.

Table 23-7. UCR*n* Field Descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>3–2
TC</td><td>Transmit command field. Selects a single transmit command.
<table>
<tr><th></th><th>Command</th><th>Description</th></tr>
<tr><td>00</td><td>NO ACTION TAKEN</td><td>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.</td></tr>
<tr><td>01</td><td>TRANSMITTER ENABLE</td><td>Enables operation of the UART’s transmitter. USRn[TXEMP,TXRDY] are set. If the transmitter is already enabled, this command has no effect.</td></tr>
<tr><td>10</td><td>TRANSMITTER DISABLE</td><td>Terminates transmitter operation and clears USRn[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.</td></tr>
<tr><td>11</td><td>—</td><td>Reserved, do not use.</td></tr>
</table></td></tr>
<tr><td>1–0
RC</td><td>Receive command field. Selects a single receive command.
<table>
<tr><th></th><th>Command</th><th>Description</th></tr>
<tr><td>00</td><td>NO ACTION TAKEN</td><td>Causes the receiver to stay in its current mode. If the receiver is enabled, it remains enabled; if disabled, it remains disabled.</td></tr>
<tr><td>01</td><td>RECEIVER ENABLE</td><td>If the UART module is not in multidrop mode (UMR1n[PM] ≠ 11), RECEIVER ENABLE enables the UART's receiver and forces it into search-for-start-bit state. If the receiver is already enabled, this command has no effect.</td></tr>
<tr><td>10</td><td>RECEIVER DISABLE</td><td>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 loopback or multidrop mode, the receiver operates even though this command is selected. If the receiver is already disabled, the command has no effect.</td></tr>
<tr><td>11</td><td>—</td><td>Reserved, do not use.</td></tr>
</table></td></tr>
</table>

23.3.6 UART Receive Buffers (URB*n*)

The receive buffers contain one serial shift register and three receiver holding registers, which act as a FIFO. URXD*n* 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 23-18). RB contains the character in the receiver.

IPSBAR 0x00_020C (URB0)
Offset: 0x00_024C (URB1)
0x00_028C (URB2)

Access: User read-only

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

Figure 23-8. UART Receive Buffer (URB*n*)

23.3.7 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 UART'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 UART's TXRDY is cleared and the transmitter is disabled have no effect on the transmit buffer.

Figure 23-9 shows UTB*n*. TB contains the character in the transmit buffer.

IPSBAR 0x00_020C (UTB0)
Offset: 0x00_024C (UTB1)
0x00_028C (UTB2)

Access: User write-only

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

Figure 23-9. UART Transmit Buffer (UTB*n*)

23.3.8 UART Input Port Change Registers (UIPCR*n*)

The UIPCRs hold the current state and the change-of-state for $\overline{\text{UCTS}}n$.

IPSBAR 0x00_0210 (UIPCR0)
Offset: 0x00_0250 (UIPCR1)
0x00_0290 (UIPCR2)

Access: User read-only

	7	6	5	4	3	2	1	0
R	0	0	0	COS	1	1	1	CTS
W								
Reset:	0	0	0	0	1	1	1	$\overline{\text{UCTS}}n$

Figure 23-10. UART Input Port Changed Registers (UIPCR*n*)

Table 23-8. UIPCR*n* Field Descriptions

Field	Description
7–5	Reserved
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 $\overline{\text{UCTS}}$*n* input. UACR*n* can be programmed to generate an interrupt to the CPU when a change of state is detected.
3–1	Reserved
0 CTS	Current state of clear-to-send. Starting two serial clock periods after reset, CTS reflects the state of $\overline{\text{UCTS}}$*n*. If $\overline{\text{UCTS}}$*n* 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 $\overline{\text{UCTS}}$*n* input is asserted. 1 The current state of the $\overline{\text{UCTS}}$*n* input is deasserted.

23.3.9 UART Auxiliary Control Register (UACR*n*)

The UACRs control the input enable.

IPSBAR 0x00_0210 (UACR0) Access: User write-only
Offset: 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 23-11. UART Auxiliary Control Registers (UACR*n*)

Table 23-9. UACR*n* Field Descriptions

Field	Description
7–1	Reserved, must 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 $\overline{\text{UCTS}}$*n* input (if UIMR*n*[COS] = 1).

23.3.10 UART Interrupt Status/Mask Registers (UISR*n*/UIMR*n*)

The UISRs 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, internal interrupt output is asserted. If a UIMR*n* bit is cleared, 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.

IPSBAR 0x00_0214 (UISR0) Access: User read/write
Offset: 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 23-12. UART Interrupt Status/Mask Registers (UISR*n*/UIMR*n*)

Table 23-10. 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 $\overline{\text{UCTS}}$*n* and was programmed in UACR*n*[IEC] to cause an interrupt.
6–3	Reserved, must be cleared.
2 DB	Delta break. 0 No new break-change condition to report. Section 23.3.5, "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] and USR*n*[RXRDY]
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 is cleared 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

23.3.11 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 23.4.1.2.1, "Internal Bus Clock Baud Rates."

IPSBAR 0x00_0218 (UBG10) Access: User write-only
Offset: 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 23-13. UART Baud Rate Generator Registers (UBG1*n*)

IPSBAR 0x00_021C (UBG20) Access: User write-only
Offset: 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 23-14. UART Baud Rate Generator Registers (UBG2*n*)

NOTE

The minimum value 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. UBG1*n* and UBG2*n* are write-only and cannot be read by the CPU.

23.3.12 UART Input Port Register (UIP*n*)

The UIP*n* registers show the current state of the $\overline{\text{UCTS}n}$ input.

IPSBAR 0x00_0234 (UIP0) Access: User read-only
Offset: 0x00_0274 (UIP1)
0x00_02B4 (UIP2)

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

Figure 23-15. UART Input Port Registers (UIP*n*)

Table 23-11. UIP*n* Field Descriptions

Field	Description
7–1	Reserved
0 CTS	Current state of clear-to-send. The $\overline{\text{UCTS}n}$ 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*[CTS]. 0 The current state of the $\overline{\text{UCTS}n}$ input is logic 0. 1 The current state of the $\overline{\text{UCTS}n}$ input is logic 1.

23.3.13 UART Output Port Command Registers (UOP1*n*/UOP0*n*)

The $\overline{\text{URTS}n}$ output can be asserted by writing a 1 to UOP1*n*[RTS] and negated by writing a 1 to UOP0*n*[RTS].

IPSBAR 0x00_0238 (UOP10) Access: User write-only
Offset: 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	0	0	0	0	0	0	0	RTS
Reset:	0	0	0	0	0	0	0	0

Figure 23-16. UART Output Port Command Registers (UOP1*n*/UOP0*n*)

Table 23-12. UOP1*n*/UOP0*n* Field Descriptions

Field	Description
7–1	Reserved, must be cleared.
0 RTS	Output port output. Controls assertion (UOP1)/negation (UOP0) of $\overline{\text{URTS}n}$ output. 0 Not affected. 1 Asserts $\overline{\text{URTS}n}$ in UOP1. Negates $\overline{\text{URTS}n}$ in UOP0.

23.4 Functional Description

This section describes operation of the clock source generator, transmitter, and receiver.

23.4.1 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 16-bit divider is used to produce standard UART baud rates.

23.4.1.1 Programmable Divider

As Figure 23-17 shows, the UART*n* transmitter and receiver can use the following clock sources:

- An external clock signal on the DTIN*n* pin. When not divided, DTIN*n* provides a synchronous clock; when divided by 16, it is asynchronous.
- The internal bus clock supplies an asynchronous clock source divided by 32 and then divided by the 16-bit value programmed in UBG1*n* and UBG2*n*. See Section 23.3.11, "UART Baud Rate Generator Registers (UBG1n/UBG2n)."

The choice of DTIN or internal bus clock is programmed in the UCSR.

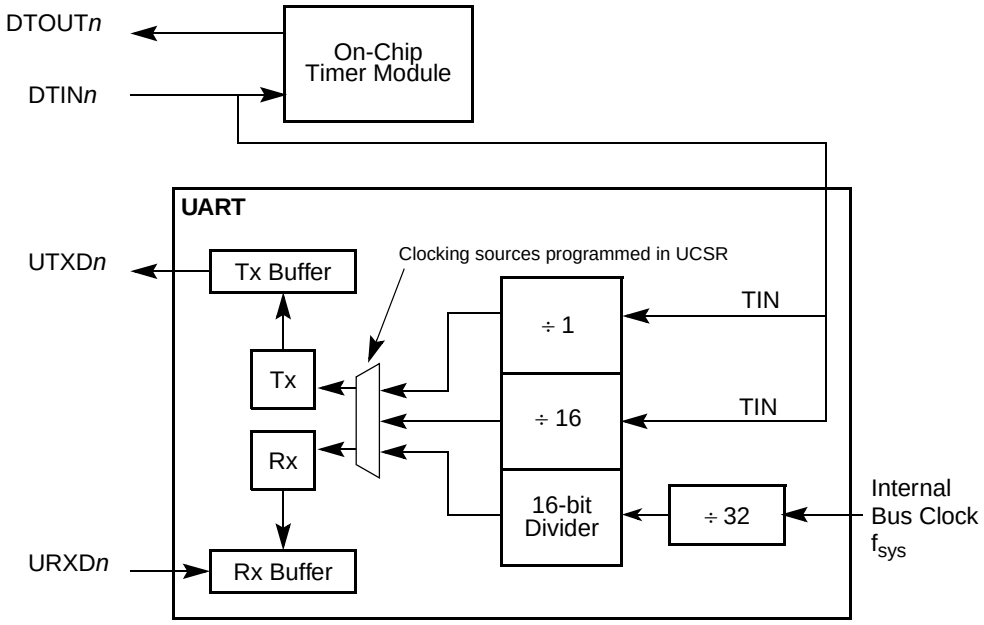


Figure 23-17. Clocking Source Diagram

NOTE

If DTIN*n* is a clocking source for the timer or UART, that timer module cannot use DTIN*n* for timer input capture.

23.4.1.2 Calculating Baud Rates

The following sections describe how to calculate baud rates.

23.4.1.2.1 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 UBG1*n* and UBG2*n* registers. The baud-rate calculation is:

$$\text{Baudrate} = \frac{f_{sys}}{[32 \times \text{Divider}]} \qquad \textit{Eqn. 23-1}$$

Using a 80-MHz internal bus clock and letting baud rate equal 9600, then

$$\text{Divider} = \frac{80\text{MHz}}{[32 \times 9600]} = 260(\text{decimal}) = 0\text{x}0104(\text{hexadecimal}) \qquad \textit{Eqn. 23-2}$$

Therefore, UBG1*n* equals 0x01 and UBG2*n* equals 0x04.

23.4.1.2.2 External Clock

An external source clock (DTIN*n*) passes through a divide-by-1 or 16 prescaler. If f_{extc} is the external clock frequency, baud rate can be described with this equation:

$$\text{Baudrate} = \frac{f_{extc}}{(16 \text{ or } 1)} \qquad \textit{Eqn. 23-3}$$

23.4.2 Transmitter and Receiver Operating Modes

Figure 23-18 is a functional block diagram of the transmitter and receiver showing the command and operating registers, which are described generally in the following sections. For detailed descriptions, refer to Section 23.3, "Memory Map/Register Definition."

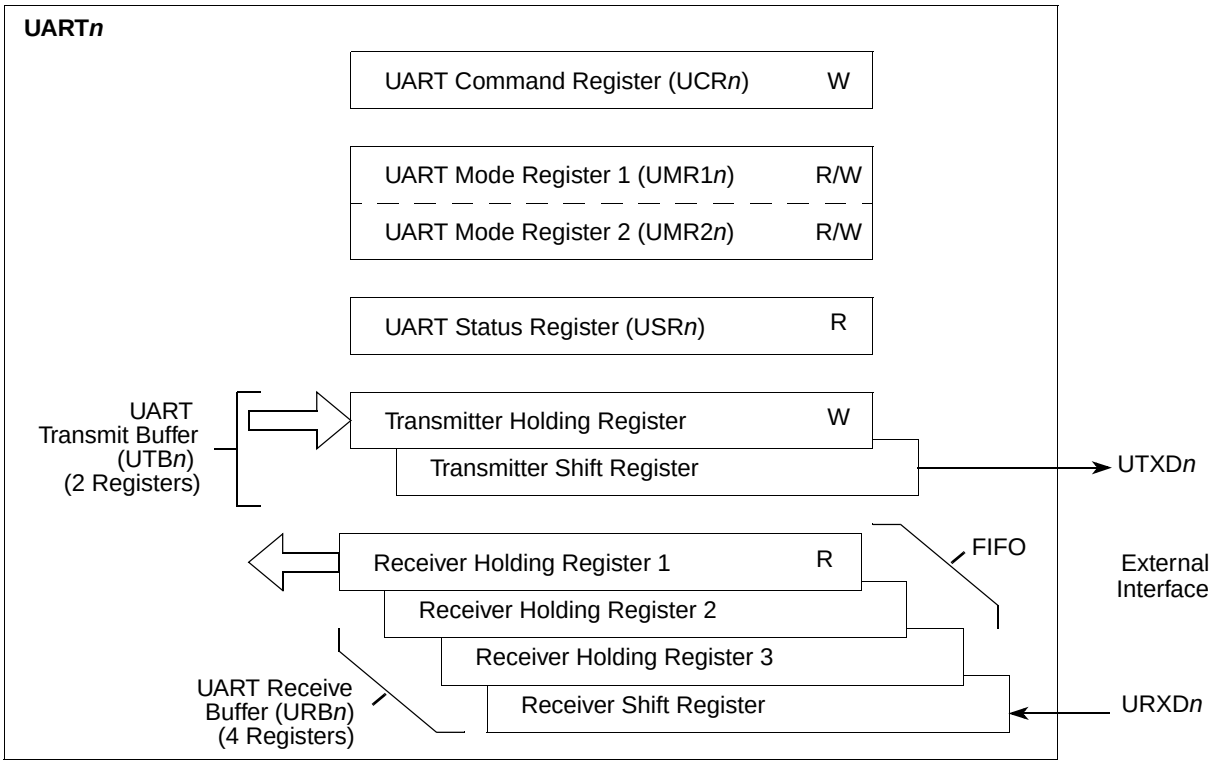


Figure 23-18. Transmitter and Receiver Functional Diagram

23.4.2.1 Transmitter

The transmitter is enabled through the UART command register (UCR*n*). When it is ready to accept a character, UART sets USR*n*[TXRDY]. The transmitter converts parallel data from the CPU to a serial bit stream on UTXD*n*. 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.

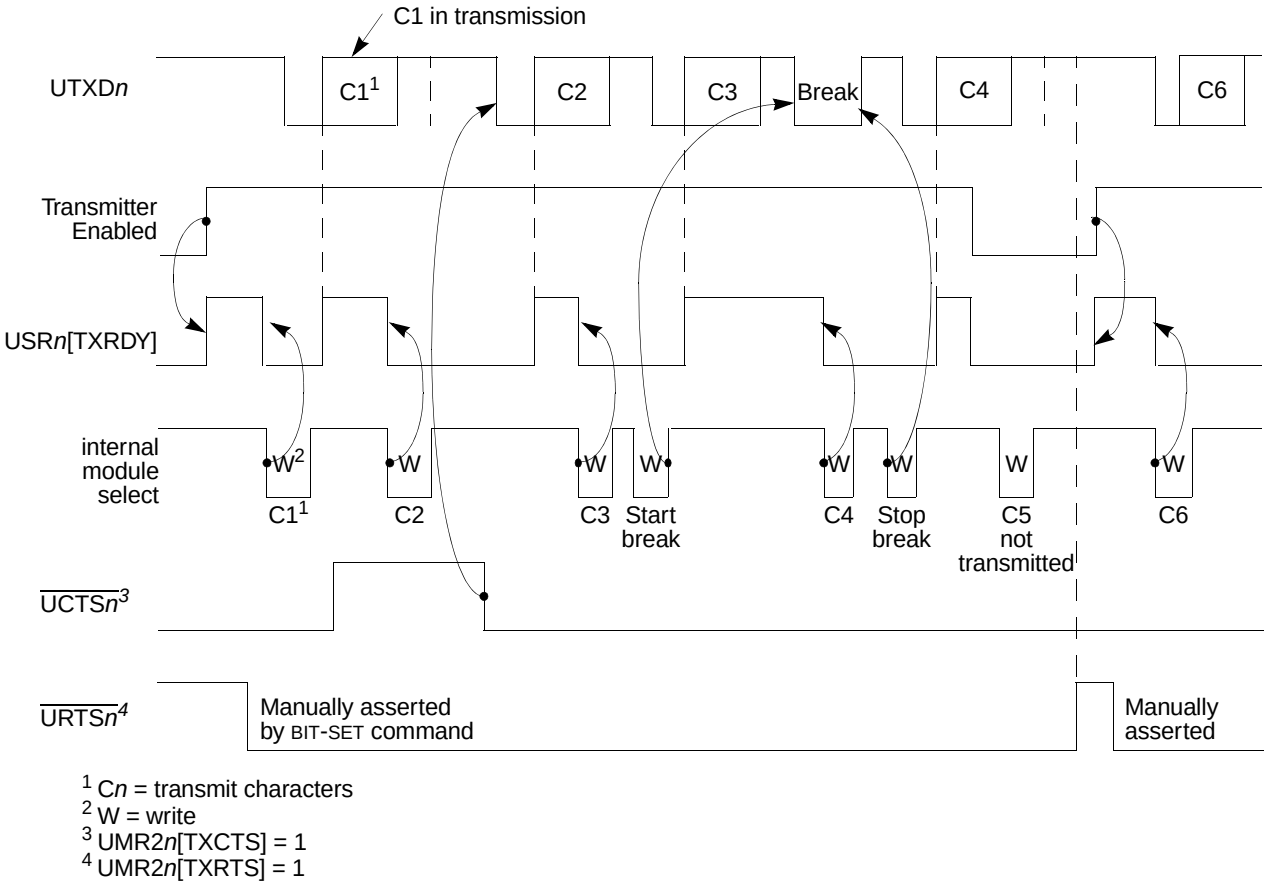
After the stop bits are sent, if no new character is in the transmitter holding register, the UTXD*n* output remains high (mark condition) and the transmitter empty bit (USR*n*[TXEMP]) is set. Transmission resumes and TXEMP is cleared when the CPU loads a new character into the UART transmit buffer (UTB*n*). 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 23.3.5, “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, $\overline{\text{UCTS}}$*n* must be asserted for the character to be transmitted. If $\overline{\text{UCTS}}$*n* is negated in the middle of a transmission, the character in the shift register is sent and UTXD*n* remains in mark state until $\overline{\text{UCTS}}$*n* is reasserted. If transmitter is forced to send a continuous low condition by issuing a SEND BREAK command, transmitter ignores the state of $\overline{\text{UCTS}}$*n*.

If the transmitter is programmed to automatically negate $\overline{\text{URTS}}$*n* when a message transmission completes, $\overline{\text{URTS}}$*n* must be asserted manually before a message is sent. In applications in which the transmitter is disabled after transmission is complete and $\overline{\text{URTS}}$*n* is appropriately programmed, $\overline{\text{URTS}}$*n* is negated one bit time after the character in the shift register is completely transmitted. The transmitter must be manually reenabled by reasserting $\overline{\text{URTS}}$*n* before the next message is sent.

Figure 23-19 shows the functional timing information for the transmitter.



[1] C*n* = transmit characters
[2] W = write
[3] UMR2*n*[TXCTS] = 1
[4] UMR2*n*[TXRTS] = 1

Figure 23-19. Transmitter Timing Diagram

23.4.2.2 Receiver

The receiver is enabled through its UCR*n*, as described in Section 23.3.5, "UART Command Registers (UCRn)."

When the receiver detects a high-to-low (mark-to-space) transition of the start bit on URXD*n*, the state of URXD*n* 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 URXD*n* is sampled high, start bit is invalid and the search for the valid start bit begins again.

If URXD*n* remains low, a valid start bit is assumed. 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 URXD*n* input is sampled on the rising edge of the programmed clock source. The lsb is received first. The data then transfers to a receiver holding register and USR*n*[RXRDY] is set. If the character is less than 8 bits, the most significant unused bits in the receiver holding register are cleared.

After the stop bit is detected, receiver immediately looks for the next start bit. However, if a non-zero character is received without a stop bit (framing error) and URXD*n* remains low for one-half of the bit period after the stop bit is sampled, 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, and RB error and break flags in the USR*n* at the received character boundary. They are valid only if USR*n*[RXRDY] is set.

If a break condition is detected (URXD*n* is low for the entire character including the stop bit), a character of all 0s loads into the receiver holding register and USR*n*[RB,RXRDY] are set. URXD*n* 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. The receiver places the damaged character in the Rx FIFO and sets the corresponding USR*n* error bits and USR*n*[RXRDY]. Then, if the break lasts until the next character time, the receiver places an all-zero character into the Rx FIFO and sets USR*n*[RB,RXRDY].

Figure 23-20 shows receiver functional timing.

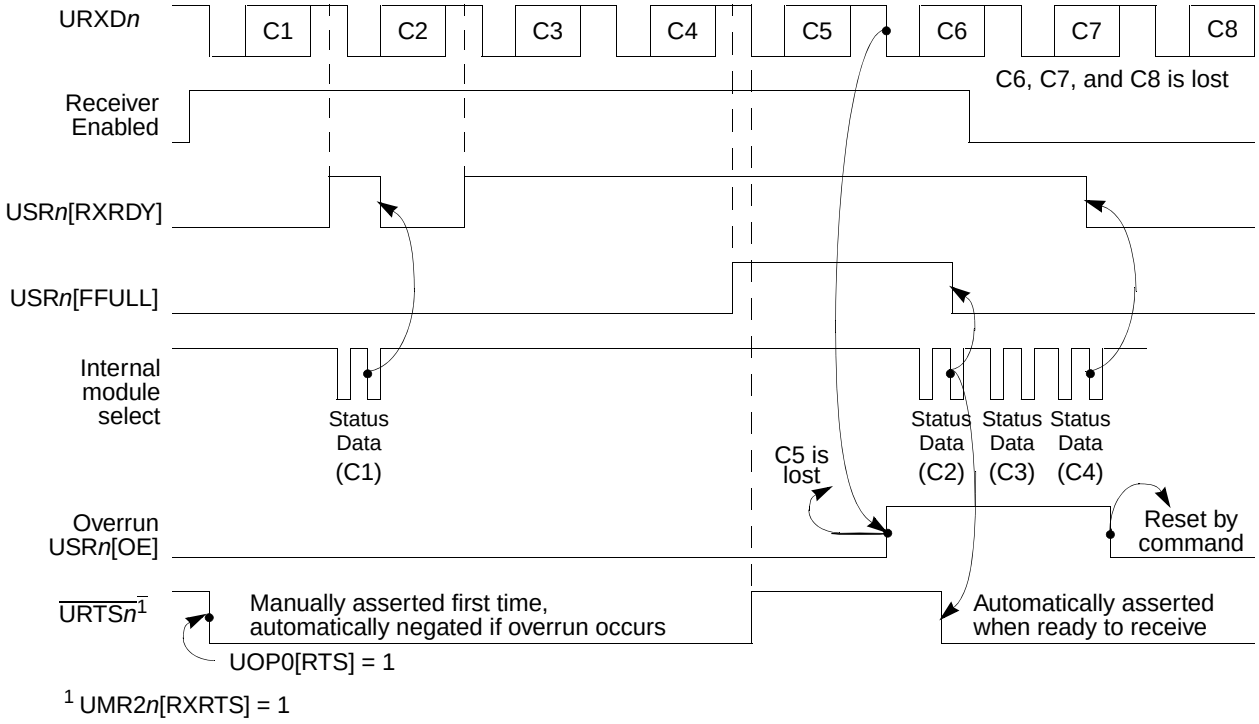


Figure 23-20. Receiver Timing Diagram

23.4.2.3 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 URXD*n* (see Figure 23-18). Data is assembled in the receiver shift register and loaded into the top empty receiver holding register position of the FIFO. Therefore, 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; overrun error (OE) is not appended. By

programming the ERR bit in the UART'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. The RXRDY or FFULL bit can be selected to cause an interrupt and TXRDY or RXRDY can be used to generate a DMA request.

The two error modes are selected by UMR1*n*[ERR]:

- 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 $\overline{\text{URTS}}n$, in which case the receiver automatically negates $\overline{\text{URTS}}n$ when a valid start bit is detected and the FIFO is full. The receiver asserts $\overline{\text{URTS}}n$ when a FIFO position becomes available; therefore, connecting $\overline{\text{URTS}}n$ to the $\overline{\text{UCTS}}n$ input of the transmitting device can prevent overrun errors.

NOTE

The receiver continues reading characters in the FIFO if the receiver is disabled. If the receiver is reset, the FIFO, $\overline{\text{URTS}}n$ control, all receiver status bits, interrupts, and DMA requests are reset. No more characters are received until the receiver is reenabled.

23.4.3 Looping Modes

The UART can be configured to operate in various looping modes. These modes are useful for local and remote system diagnostic functions. The modes are described in the following paragraphs and in Section 23.3, "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.

23.4.3.1 Automatic Echo Mode

In automatic echo mode, shown in Figure 23-21, 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 UTXD*n*. The receiver must be enabled, but the transmitter need not be.

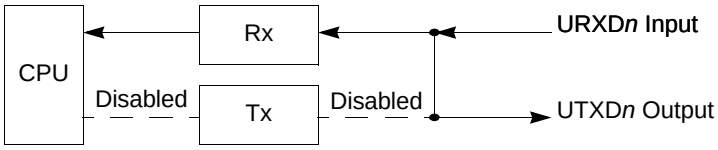


Figure 23-21. Automatic Echo

Because the transmitter is inactive, USR*n*[TXEMP,TXRDY] is 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.

23.4.3.2 Local Loopback Mode

Figure 23-22 shows how UTXD*n* and URXD*n* are internally connected in local loopback mode. This mode is for testing the operation of a UART by sending data to the transmitter and checking data assembled by the receiver to ensure proper operations.

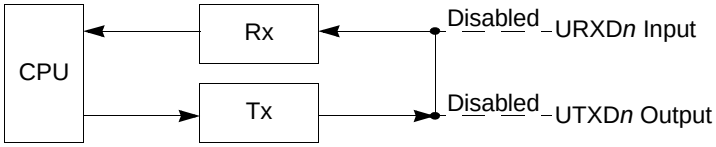


Figure 23-22. Local Loopback

Features of this local loopback mode are:

- Transmitter and CPU-to-receiver communications continue normally in this mode.
- URXD*n* input data is ignored.
- UTXD*n* is held marking.
- The receiver is clocked by the transmitter clock. The transmitter must be enabled, but the receiver need not be.

23.4.3.3 Remote Loopback Mode

In remote loopback mode, shown in Figure 23-23, the UART automatically transmits received data bit by bit on the UTXD*n* output. The local CPU-to-transmitter link is disabled. This mode is useful in testing receiver and transmitter operation of a remote UART. For this mode, 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 next valid start bit is detected.

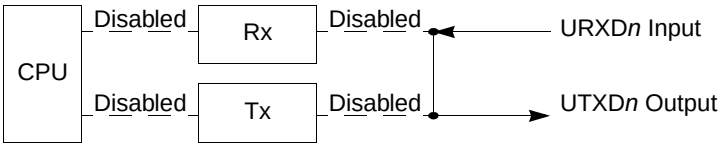


Figure 23-23. Remote Loopback

23.4.4 Multidrop Mode

Setting UMR1*n*[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 receivers disabled, they continuously monitor the master's data stream. When the master sends an address character, the slave receiver notifies its respective CPU by setting USR*n*[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. Unaddressed slave stations 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 23-24.

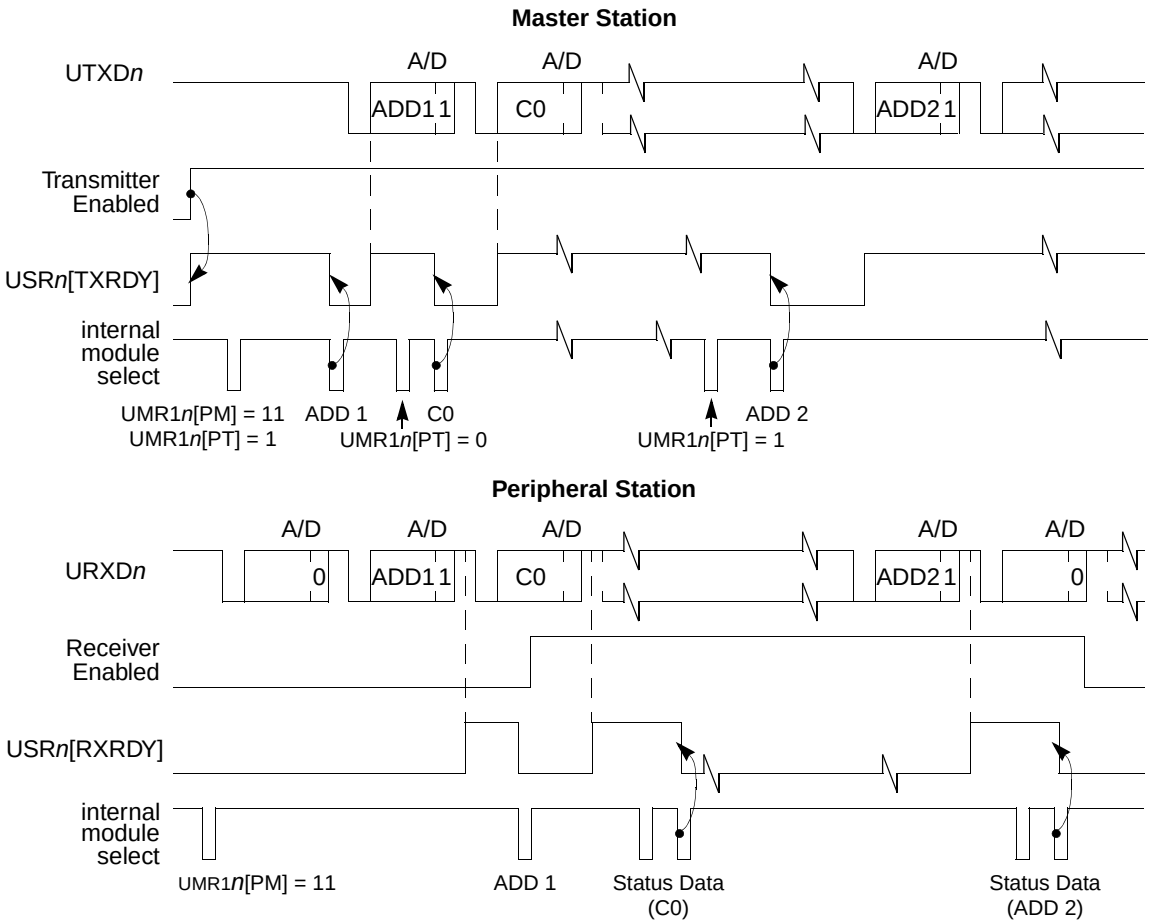


Figure 23-24. Multidrop Mode Timing Diagram

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 equals 1 indicates an address character; A/D equals 0 indicates a data character. The polarity of A/D is selected through UMR1*n*[PT]. UMR1*n* 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 1 (address tag). The character is discarded if the received A/D bit is 0 (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, data bits load into the data portion of the FIFO while the A/D bit loads into the status portion of the FIFO normally used for a parity error (USR*n*[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 continues containing error detection and correction information. If 8-bit characters are not required, one way to provide error detection is to use software to calculate parity and append it to the 5-, 6-, or 7-bit character.

23.4.5 Bus Operation

This section describes bus operation during read, write, and interrupt acknowledge cycles to the UART module.

23.4.5.1 Read Cycles

The UART module responds to reads with byte data. Reserved registers return zeros.

23.4.5.2 Write Cycles

The UART module accepts write data as bytes only. Write cycles to read-only or reserved registers complete normally without an error termination, but data is ignored.

23.5 Initialization/Application Information

The software flowchart, Figure 23-25, consists of:

- UART module initialization—These routines consist of SINIT and CHCHK (See Sheet 1 p. 23-30 and Sheet 2 p. 23-31). 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 loopback 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. 23-33 and Sheet 5 p. 23-34) 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—This consists of SIRQ (See Sheet 4 p. 23-33), 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.

23.5.1 Interrupt and DMA Request Initialization

23.5.1.1 Setting up the UART to Generate Core Interrupts

The list below provides steps to properly initialize the UART to generate an interrupt request to the processor's interrupt controller. See Section 10.3.6.1, "Interrupt Sources," for details on interrupt assignments for the UART modules.

1. Initialize the appropriate ICR*x* 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 generates interrupt requests, 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 23-13.

Table 23-13. 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

23.5.1.2 Setting up the UART to Request DMA Service

The UART is capable of generating two internal DMA request signals: transmit and receive.

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 transmitted from memory and writing it into the UART transmit buffer (UTB*n*). This allows the DMA channel to stream data from memory to the UART for transmission without processor intervention. After 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 FIFO full or receive ready (FFULL/RXRDY) 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. After 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 continuing to support interrupt notification to the processor for $\overline{\text{CTS}}$ change-of-state and delta break error managing.

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, UIMR1[TXRDY] should be cleared. This prevents TXRDY from generating an interrupt request while a transmit DMA request is generated.
3. Enable DMA access to the UART*n* registers by setting the corresponding PACR register in the SCM for read/write in supervisor and user modes.
4. Enable DMA access to SRAM by setting the SPV bit in the core RAMBAR, and the BDE bit in the SCM RAMBAR
5. Initialize the DMA channel. The DMA should be configured for cycle steal mode and a source and destination size of one byte. This causes a single byte to be transferred for each UART DMA request. Set the disable request bit (DCR*n*[D_REQ] to disable external requests when the BCR reaches zero.
6. For a transmit process:
   — Set the DMA SAR register to the address of the source data
   — Set DCR*n*[SINC] to increment the source pointer
   — Set DAR to the address if the UART transmit buffer (UTB)
   — Clear DCR*n*[DINC]
   — Set BCR to the number of bytes to transmit.
7. For a receive process:
   — Set the DMA SAR register to the address of the UART receive buffer (URB)
   — Clear DCR*n*[SINC]
   — Set DAR to the address of the source data
   — Set DCR*n*[DINC] to increment the destination pointer
   — Set BCR to the number of bytes to transmit.
8. Start the data transfer by setting DCR*n*[EEXT], which enables the UART channel to issue DMA requests.

Table 23-14 shows the DMA requests.

Table 23-14. UART DMA Requests

Register	Bit	DMA Request
UISR*n*	1	Receive DMA request
UISR*n*	0	Transmit DMA request

23.5.2 UART Module Initialization Sequence

The following shows the UART module initialization sequence.

1. UCR*n*:
   a) Reset the receiver and transmitter.
   b) Reset the mode pointer (MISC[2–0] = 0b001).
2. UIMR*n*: Enable the desired interrupt sources.
3. UACR*n*: Initialize the input enable control (IEC bit).
4. UCSR*n*: Select the receiver and transmitter clock. Use timer as source if required.
5. UMR1*n*:
   a) If preferred, program operation of receiver ready-to-send (RXRTS bit).
   a) Select receiver-ready or FIFO-full notification (RXRDY/FFULL bit).
   b) Select character or block error mode (ERR bit).
   c) Select parity mode and type (PM and PT bits).
   d) Select number of bits per character (B/Cx bits).
6. UMR2*n*:
   a) Select the mode of operation (CM bits).
   b) If preferred, program operation of transmitter ready-to-send (TXRTS).
   c) If preferred, program operation of clear-to-send (TXCTS bit).
   d) Select stop-bit length (SB bits).
7. UCR*n*: Enable transmitter and/or receiver.

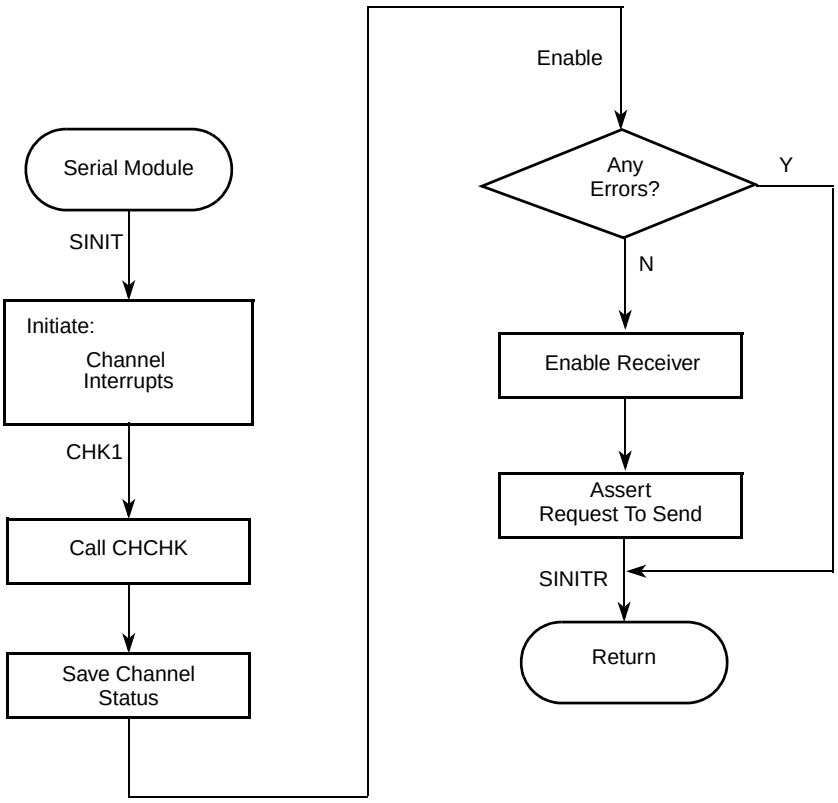


Figure 23-25. UART Mode Programming Flowchart (Sheet 1 of 5)

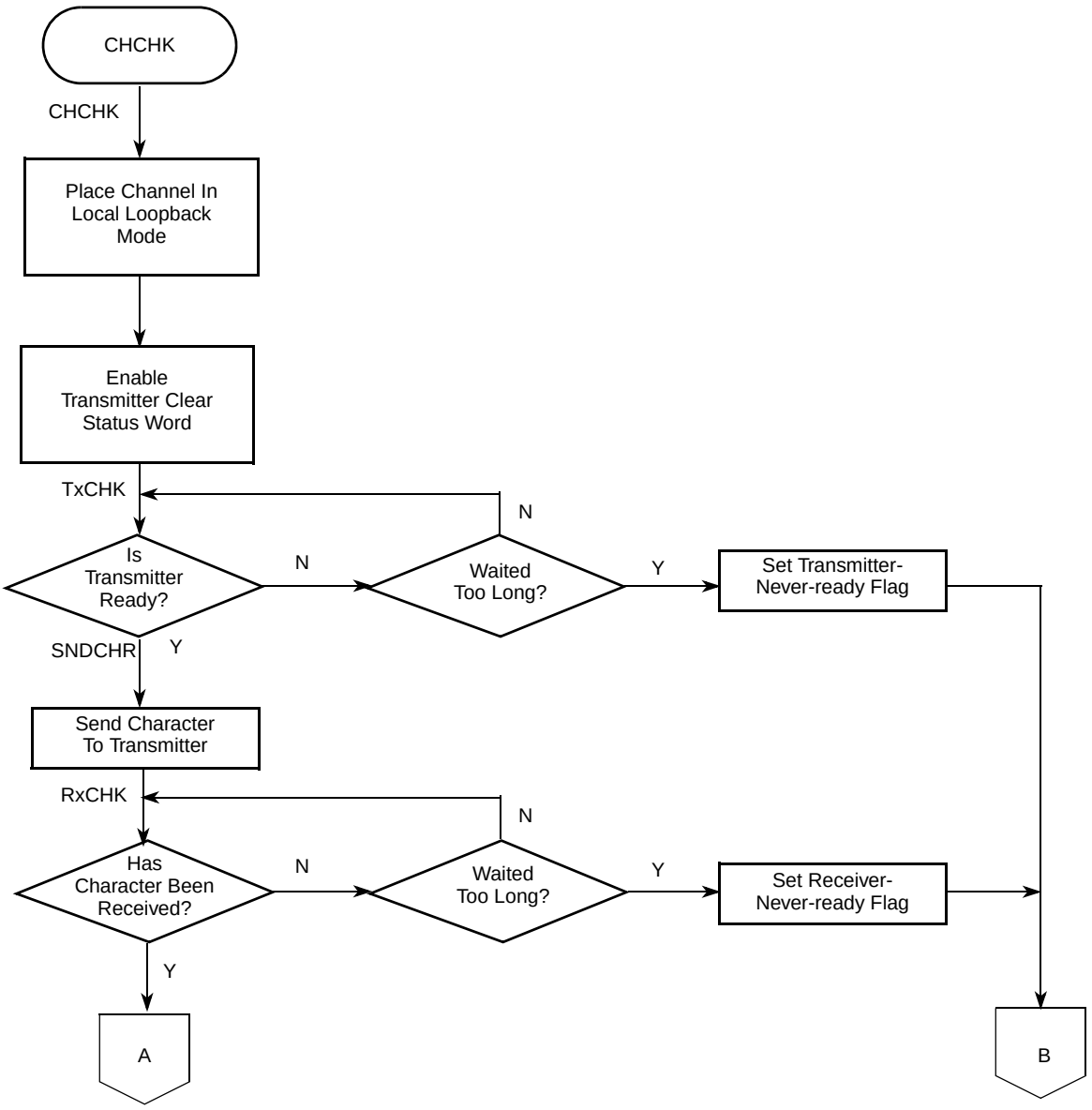


Figure 23-25. UART Mode Programming Flowchart (Sheet 2 of 5)

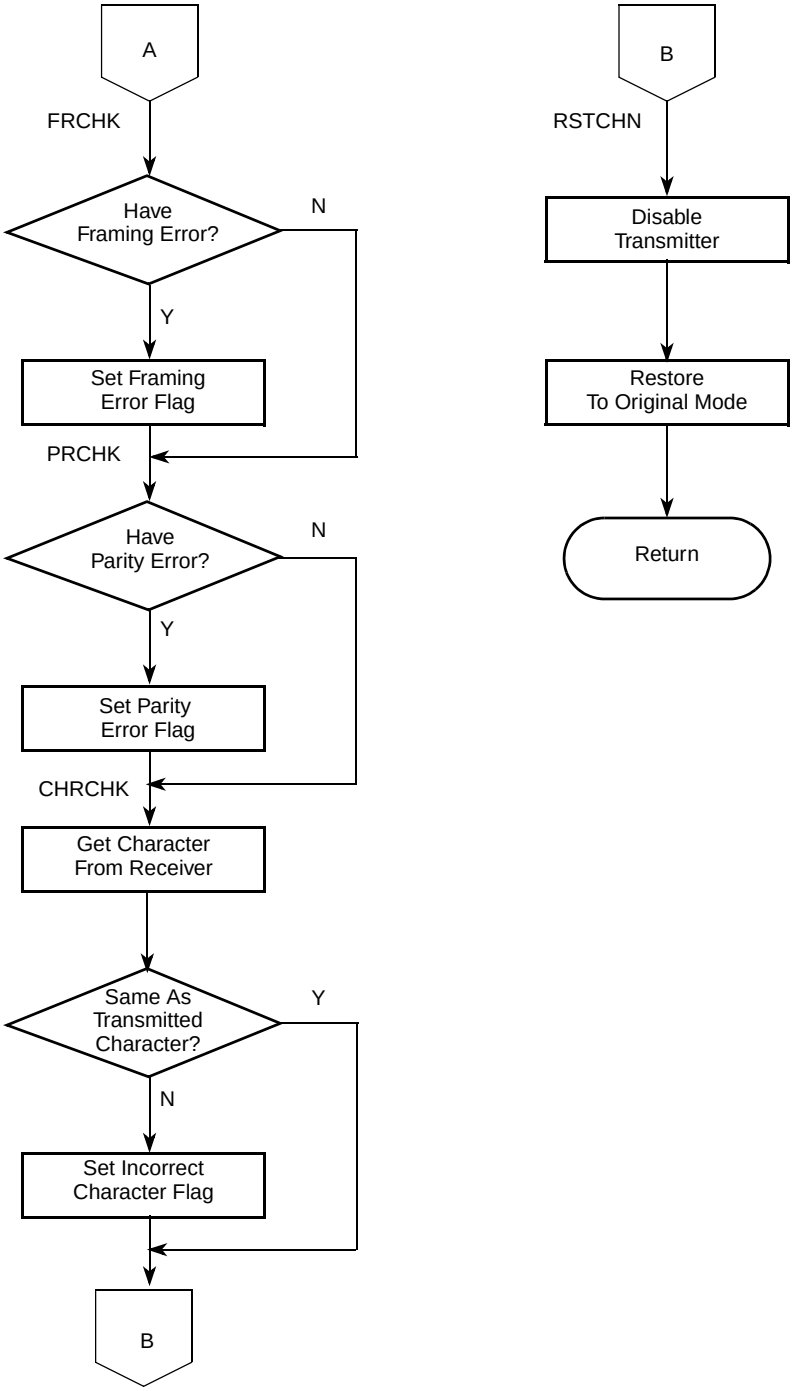


Figure 23-25. UART Mode Programming Flowchart (Sheet 3 of 5)

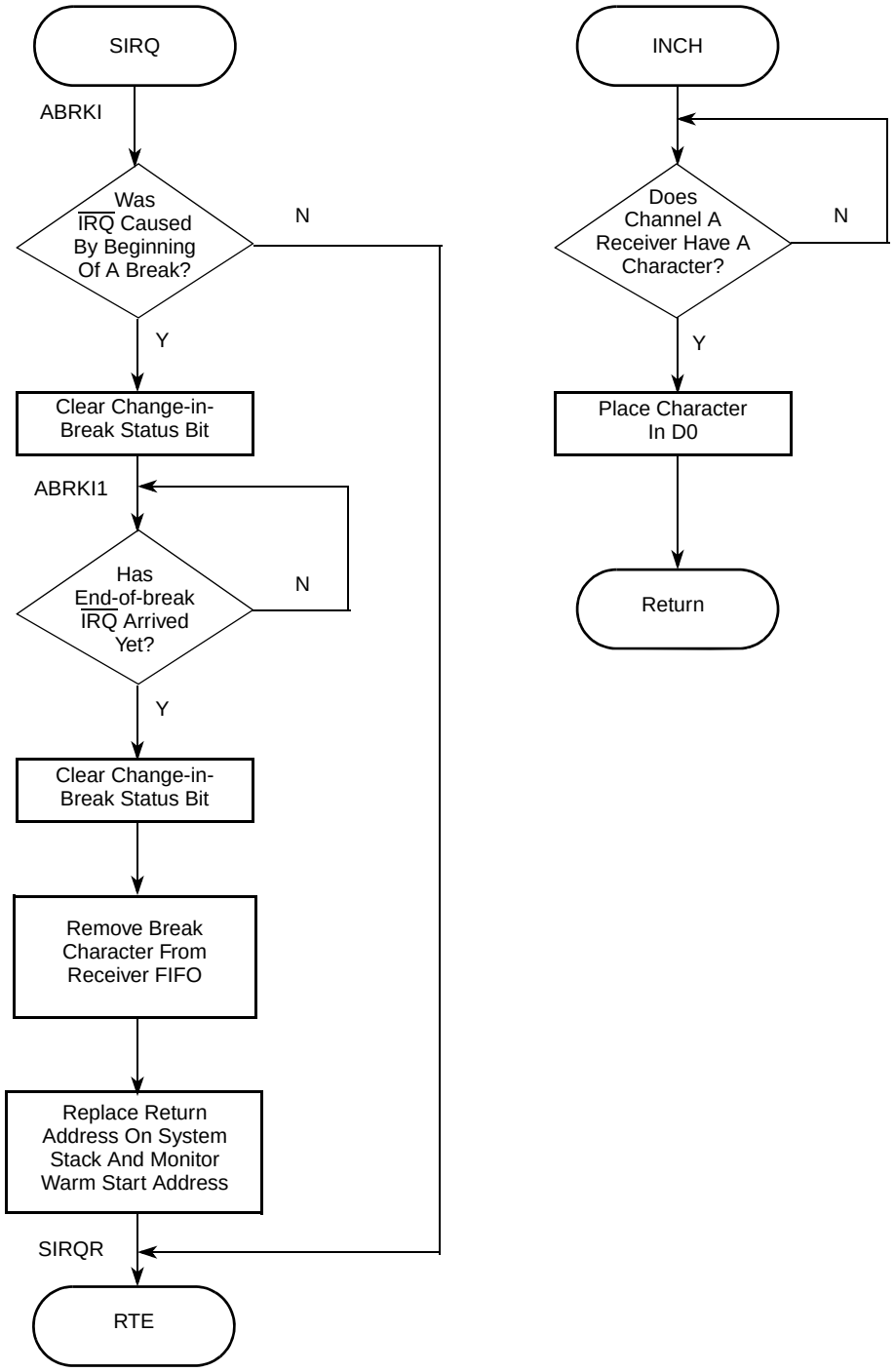


Figure 23-25. UART Mode Programming Flowchart (Sheet 4 of 5)

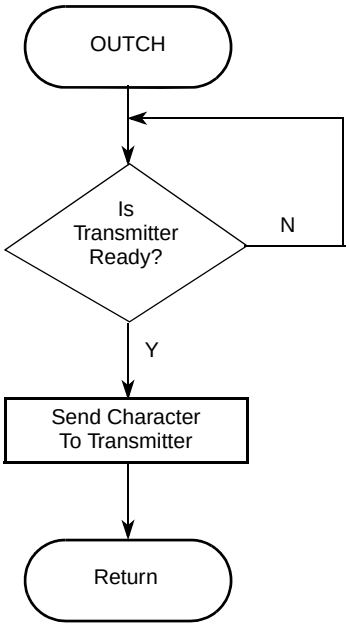


Figure 23-25. UART Mode Programming Flowchart (Sheet 5 of 5)

Chapter 24
I²C Interface

24.1 Introduction

This chapter describes the I²C module, clock synchronization, and I²C programming model registers. It also provides extensive programming examples.

24.1.1 Block Diagram

Figure 24-1 is a I²C module block diagram, illustrating the interaction of the registers described in Section 24.2, “Memory Map/Register Definition”.

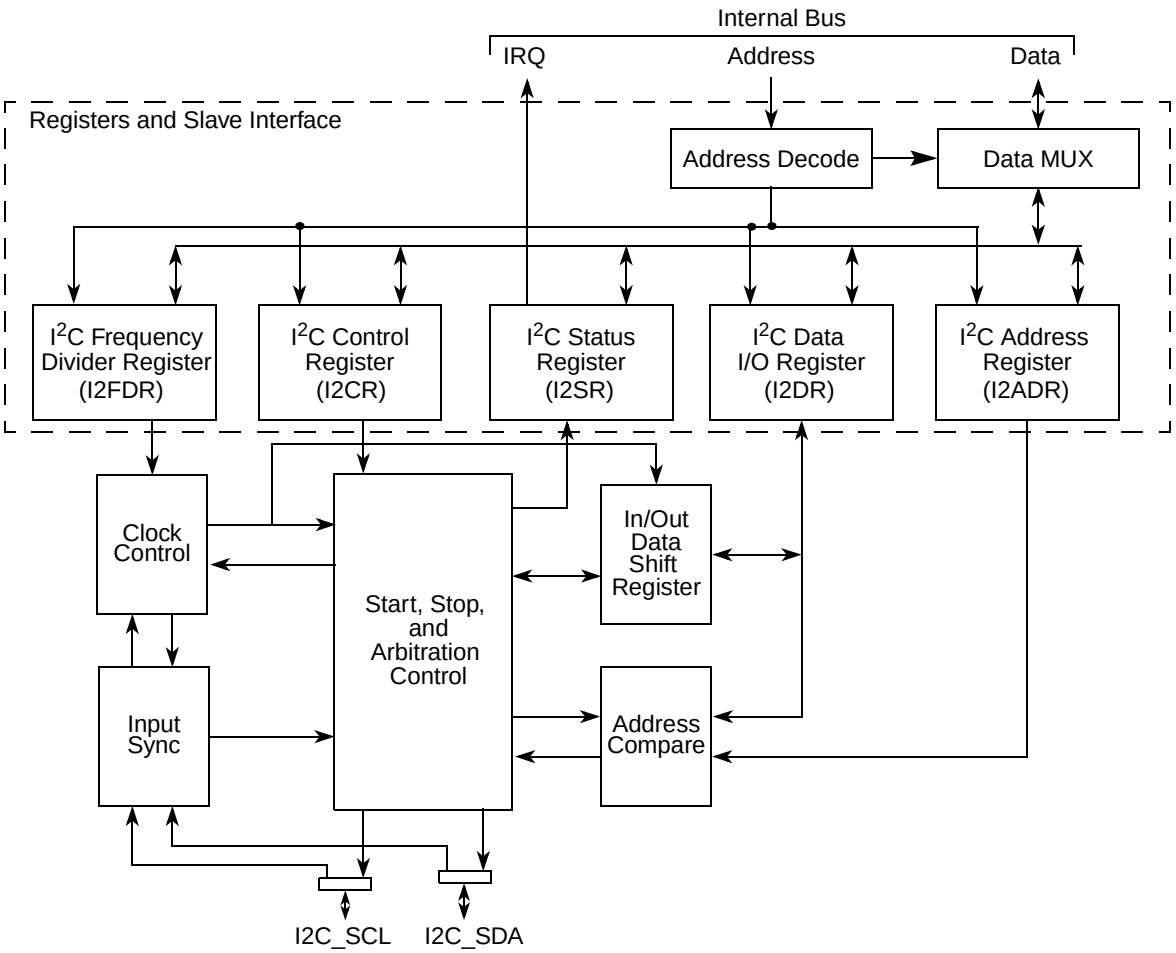


Figure 24-1. I²C Module Block Diagram

24.1.2 Overview

I²C 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 I²C bus allows additional devices to connect to the bus for expansion and system development.

The interface operates 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 connected are limited by a maximum bus capacitance of 400 pF.

The I²C 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 I²C module is compatible with the Philips I²C bus protocol. For information on system configuration, protocol, and restrictions, see *The I²C Bus Specification, Version 2.1*.

NOTE

The GPIO module must be configured to enable the peripheral function of the appropriate pins (refer to Chapter 26, “General Purpose I/O Module”) prior to configuring the I²C module.

24.1.3 Features

The I²C module has these key features:

- Compatibility with I²C bus standard version 2.1
- 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

24.2 Memory Map/Register Definition

The below table lists the configuration registers used in the I²C interface.

Table 24-1. I²C Module Memory Map

IPSBAR Offset	Register	Access	Reset Value	Section/Page
0x00_0300	I²C Address Register (I2ADR)	R/W	0x00	24.2.1/24-3
0x00_0304	I²C Frequency Divider Register (I2FDR)	R/W	0x00	24.2.2/24-3
0x00_0308	I²C Control Register (I2CR)	R/W	0x00	24.2.3/24-4
0x00_030C	I²C Status Register (I2SR)	R/W	0x81	24.2.4/24-5
0x00_0310	I²C Data I/O Register (I2DR)	R/W	0x00	24.2.5/24-6

24.2.1 I²C Address Register (I2ADR)

I2ADR holds the address the I²C responds to when addressed as a slave. It is not the address sent on the bus during the address transfer when the module is performing a master transfer.

IPSBAR Offset: 0x00_0300 (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 24-2. I²C Address Register (I2ADR)

Table 24-2. 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, must be cleared.

24.2.2 I²C Frequency Divider Register (I2FDR)

The I2FDR, shown in Figure 24-3, provides a programmable prescaler to configure the I²C clock for bit-rate selection.

IPSBAR Offset: 0x00_0304 (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 24-3. I²C Frequency Divider Register (I2FDR)

Table 24-3. I2FDR Field Descriptions

Field	Description
7–6	Reserved, must 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

24.2.3 I²C Control Register (I2CR)

I2CR enables the I²C module and the I²C interrupt. It also contains bits that govern operation as a slave or a master.

IPSBAR Offset: 0x00_0308 (I2CR) Access: User read/write

	7	6	5	4	3	2	1	0
Reset:	0	0	0	0	0	0	0	0

Figure 24-4. I²C Control Register (I2CR)

Table 24-4. 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; initiating a start cycle may corrupt the current bus cycle, ultimately causing 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 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 must set MTX according to I2SR[SRW]. In master mode, MTX must 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 master and slave receivers. 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 (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	Reserved, must be cleared.

24.2.4 I²C Status Register (I2SR)

I2SR contains bits that indicate transaction direction and status.

IPSBAR Offset: 0x00_030C (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 24-5. I²C Status Register (I2SR)

Table 24-5. 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 falling edge of 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, must 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 0 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.

24.2.5 I²C Data I/O Register (I2DR)

In master-receive mode, reading I2DR allows a read to occur and for the next data byte to be received. In slave mode, the same function is available after the I²C has received its slave address.

IPSBAR Offset: 0x00_0310 (I2DR) Access: User read/write

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

Figure 24-6. I²C Data I/O Register (I2DR)

Table 24-6. I2DR Field Description

Field	Description
7–0 DATA	I²C data. When data is written to this register in master transmit mode, 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). 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.

24.3 Functional Description

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. Therefore, 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 24.4.1, “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.

24.3.1 START Signal

When no other device is bus master (I2C_SCL and I2C_SDA lines are at logic high), a device can initiate communication by sending a START signal (see A in Figure 24-7). 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.

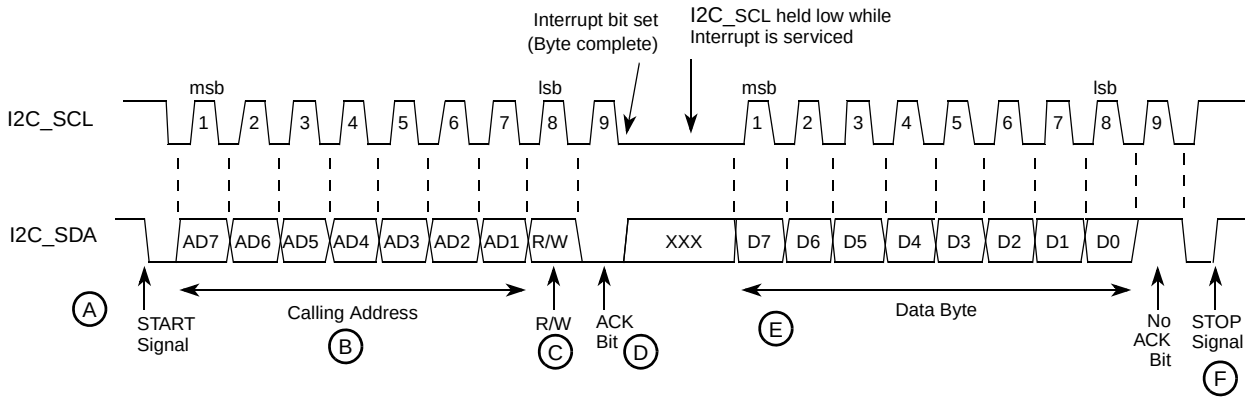


Figure 24-7. I²C Standard Communication Protocol

24.3.2 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 equals write transfer, 1 equals 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.

24.3.3 Data Transfer

When successful slave addressing is achieved, data transfer can proceed (see E in Figure 24-7) 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 24-7 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 24-8.

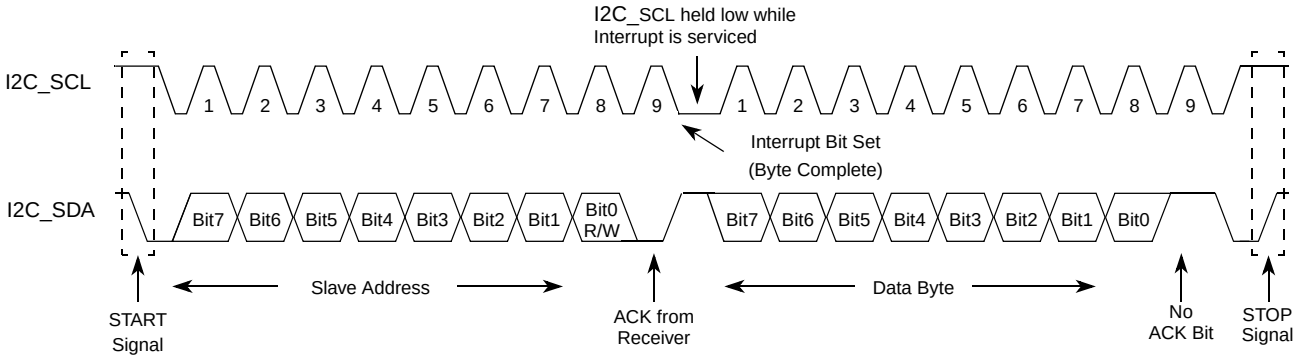


Figure 24-8. Data Transfer

24.3.4 Acknowledge

The transmitter releases the I2C_SDA line high during the acknowledge clock pulse as shown in Figure 24-9. 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 data transfer or generate a START signal (repeated start, shown in Figure 24-10 and discussed in Section 24.3.6, “Repeated START”) to start a new calling sequence.

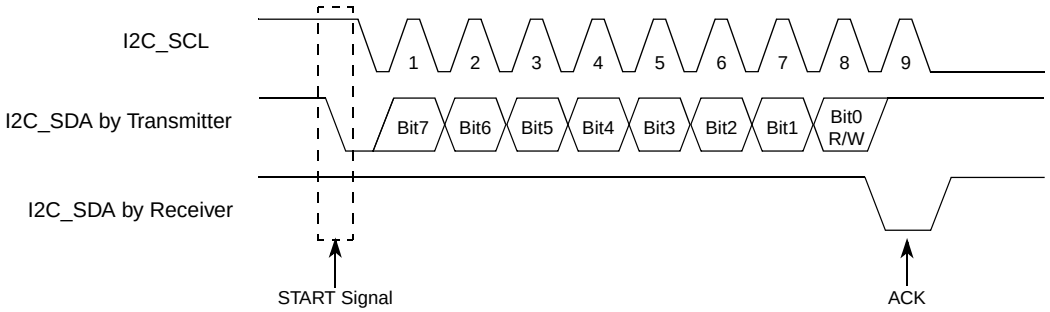


Figure 24-9. Acknowledgement by Receiver

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 24-9).

24.3.5 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 24-7). 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 24.3.6, “Repeated START.”

24.3.6 Repeated START

A repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication, as shown in Figure 24-10. 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.

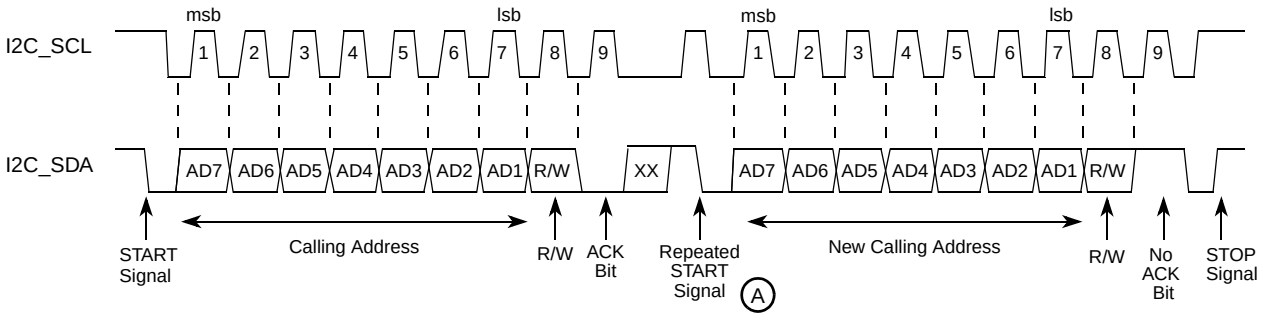


Figure 24-10. Repeated START

Various combinations of read/write formats are then possible:

- The first example in Figure 24-11 is the case of master-transmitter transmitting to slave-receiver. The transfer direction is not changed.
- The second example in Figure 24-11 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 24-11, START condition and slave address are 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/$\overline{W}$ bit.

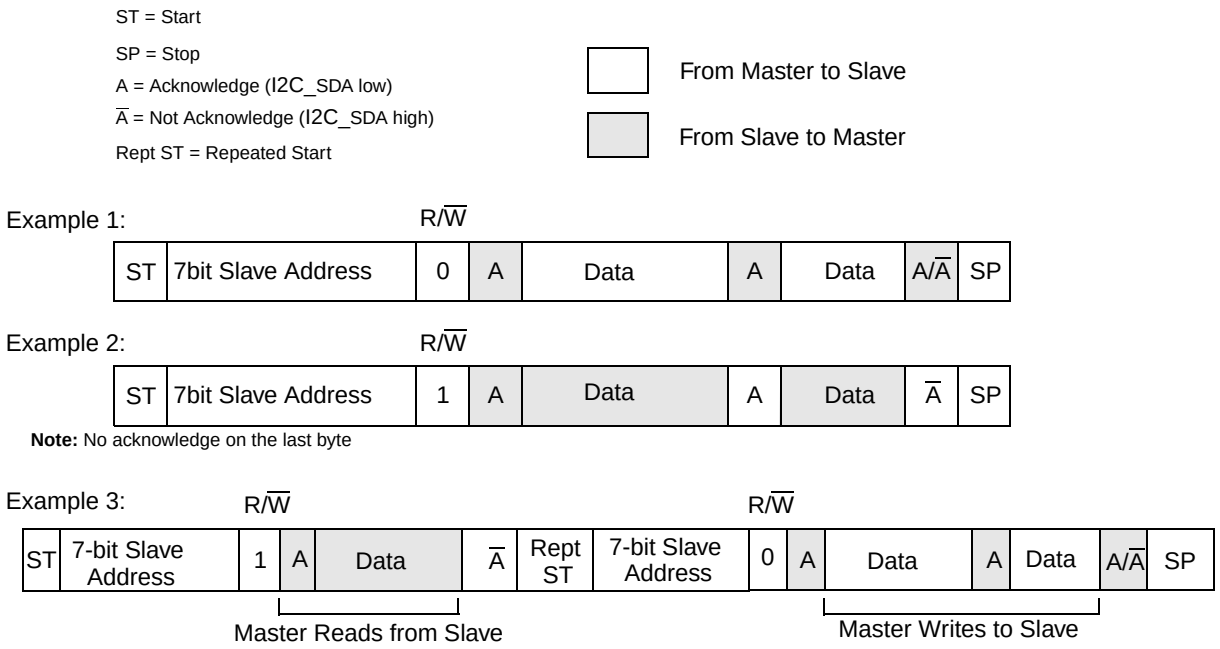


Figure 24-11. Data Transfer, Combined Format

24.3.7 Clock Synchronization and Arbitration

I²C is a true multi-master bus that allows more than one master 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 after a device's clock has gone low, it holds the I2C_SCL line low until the clock high state is reached. However, change of low to high in this device's clock may not change the state of the I2C_SCL line if another device clock remains 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 24-12). 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.

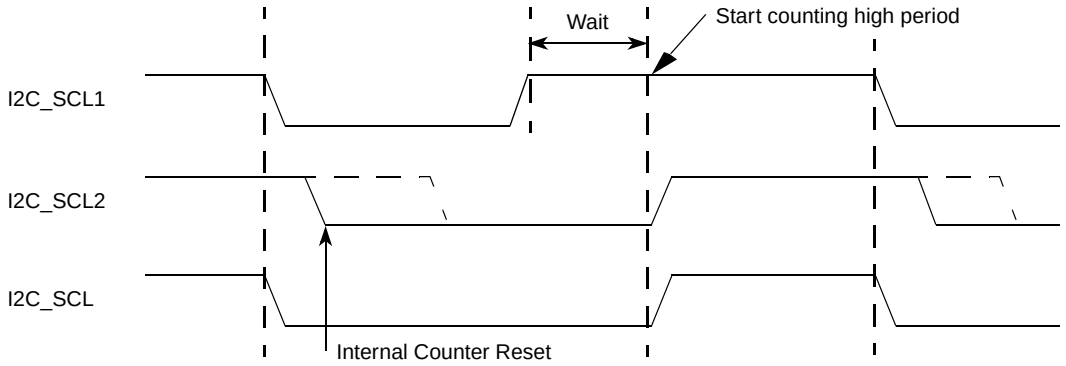


Figure 24-12. Clock Synchronization

A data arbitration procedure determines the relative priority of the contending masters. 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 24-13). In this case, transition from master to slave mode does not generate a STOP condition. Meanwhile, hardware sets I2SR[IAL] to indicate loss of arbitration.

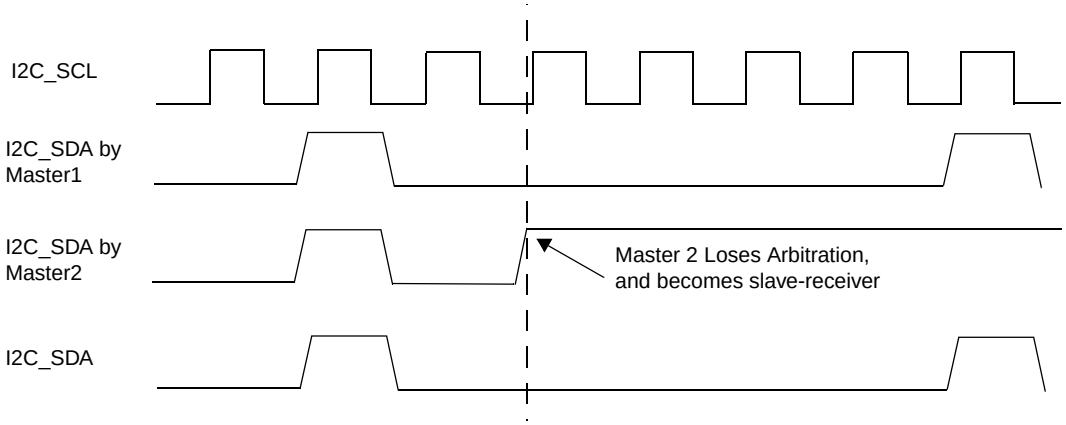


Figure 24-13. Arbitration Procedure

24.3.8 Handshaking and Clock Stretching

The clock synchronization mechanism can acts 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.

24.4 Initialization/Application Information

The following examples show programming for initialization, signaling START, post-transfer software response, signaling STOP, and generating a repeated START.

24.4.1 Initialization Sequence

Before the interface can transfer serial data, registers must be initialized:

1. Set I2FDR[IC] to obtain I2C_SCL frequency from the system bus clock. See Section 24.2.2, "I²C Frequency Divider Register (I2FDR)."
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 power-cycled on.

```
I2CR = 0x0
I2CR = 0xA0
dummy read of I2DR
I2SR = 0x0
I2CR = 0x0
I2CR = 0x80       ; re-enable
```

24.4.2 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 is cleared), 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, the

processor may need 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):

1. Check I2SR[IBB]. If it is set, wait until it is clear.
2. After cleared, set to transmit mode by setting I2CR[MTX].
3. Set master mode by setting I2CR[MSTA]. This generates a START condition.
4. Transmit the calling address via the I2DR.
5. Check I2SR[IBB]. If it is clear, wait until it is set and go to step #1.

24.4.3 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. Reading from I2DR in receive mode or writing to I2DR in transmit mode can clear I2SR[ICF].

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; 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 24-14).

1. Clear the I2CR[IIF] flag.
2. Check if acknowledge has been received, I2SR[RXAK].
3. If no ACK, end transmission. Else, transmit next byte of data via I2DR.

24.4.4 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.

1. Check if acknowledge has been received, I2SR[RXAK]. If no ACK, end transmission and go to step #5.
2. Get value from transmitting counter, TXCNT. If no more data, go to step #5.
3. Transmit next byte of data via I2DR.
4. Decrement TXCNT and go to step #1
5. Generate a stop condition by clearing I2CR[MSTA].

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.

1. Decrement RXCNT.
2. If last byte (RXCNT = 0) go to step #4.
3. If next to last byte (RXCNT = 1), set I2CR[TXAK] to disable ACK and go to step #5.
4. This is last byte, so clear I2CR[MSTA] to generate a STOP signal.
5. Read data from I2DR.
6. If there is more data to be read (RXCNT ≠ 0), go to step #1 if desired.

24.4.5 Generation of Repeated START

If the master wants the bus after the data transfer, it can signal another START followed by another slave address without signaling a STOP, as in the following example.

1. Generate a repeated START by setting I2CR[RSTA].
2. Transmit the calling address via I2DR.

24.4.6 Slave Mode

In the slave interrupt service routine, software must poll the I2SR[IAAS] bit to determine if the controller has received its slave address. If IAAS is set, software must set the transmit/receive mode select bit (I2CR[MTX]) according to the I2SR[SRW]. Writing to I2CR clears 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 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 releases I2C_SCL so the master can generate a STOP signal.

24.4.7 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 continues generating 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] set and I2CR[MSTA] cleared.

If a non-master device tries to transmit or execute a START, hardware inhibits the transmission, clears MSTA without signaling a STOP, generates an interrupt to the CPU, and sets IAL to indicate a failed attempt to engage the bus. When considering these cases, slave service routine should first test IAL and software should clear it if it is set.

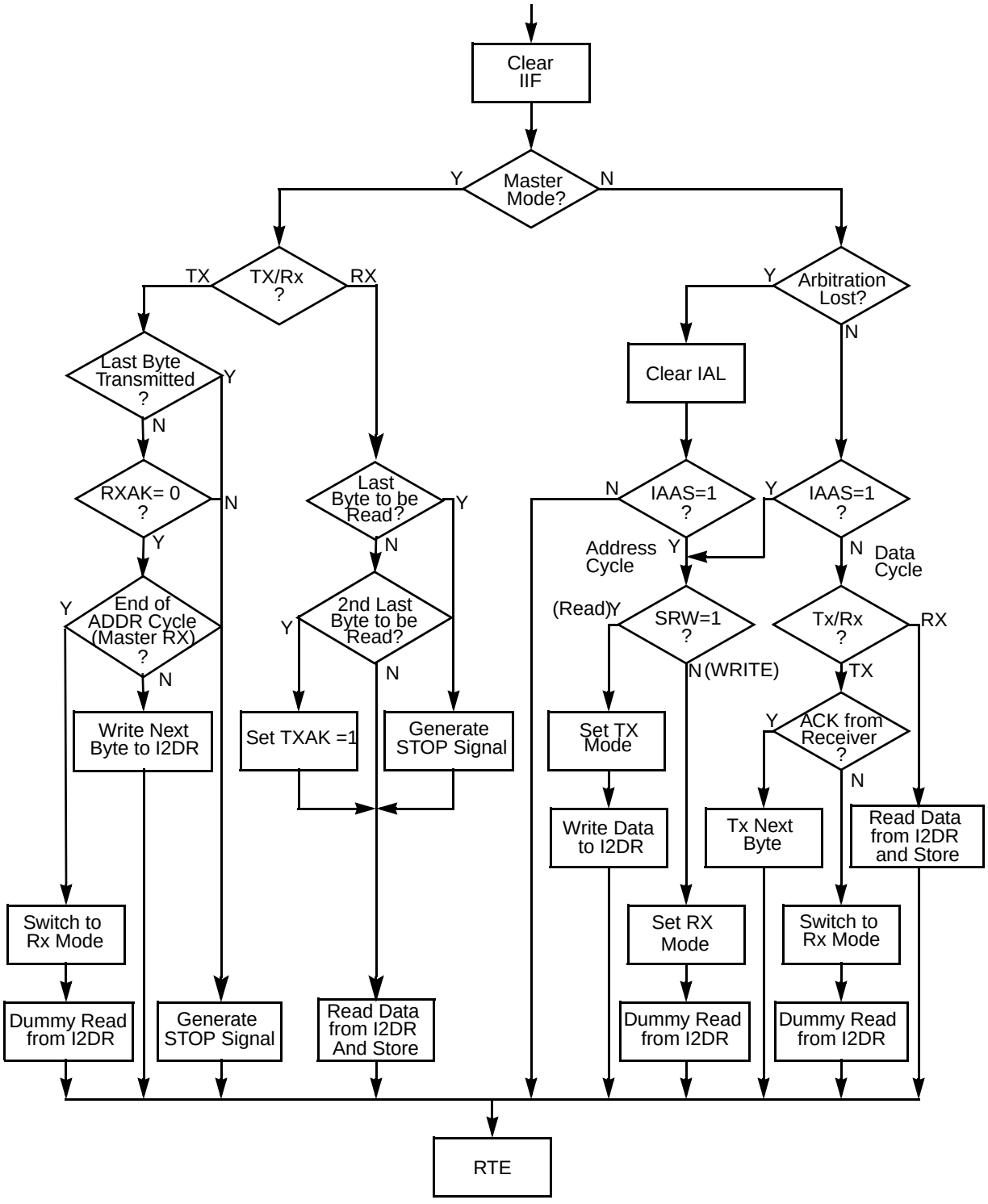


Figure 24-14. Flow-Chart of Typical I²C Interrupt Routine

Chapter 25
FlexCAN

The FlexCAN module 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 Mbit/sec), short distance, priority based protocol which 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 Features

- Based on and includes all existing Freescale TouCAN module features
- Freescale IP interface architecture
- Full implementation of the CAN protocol specification version 2.0
  - 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 1Mbit/sec
- Up to 16 flexible message buffers of 0–8 bytes data length, each configurable as Rx or Tx, all supporting standard and extended messages
- 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
- Programmable I/O modes
- Maskable interrupts
- Independent of the transmission medium (external transceiver is assumed)
- Open network architecture
- Multimaster bus
- High immunity to EMI
- Short latency time for high-priority messages
- Low-power “sleep” mode, with programmable “wake up” on bus activity

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.

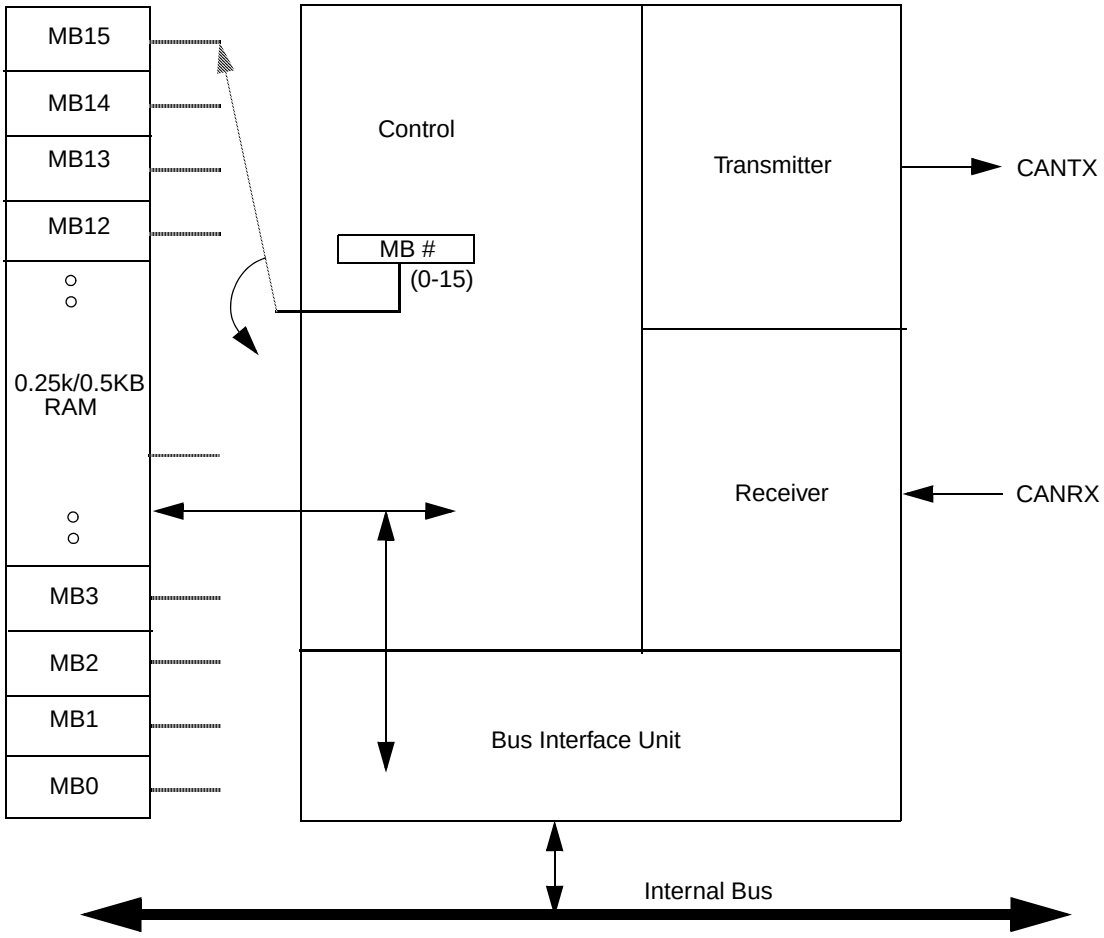


Figure 25-1. FlexCAN Block Diagram and Pinout

25.1.1 FlexCAN Memory Map

The FlexCAN module address space is split into 128 bytes starting at the base address, and then an extra 256 bytes starting at the base address +128. The upper 256 are fully used for the message buffer structures, as described in Section 25.3.2, "Message Buffer Memory Map." Out of the lower 128 bytes, only part is occupied by various registers.

Table 25-1. FlexCAN Memory Map

IPSBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x1C_0000	Module Configuration Register (MCR)		Reserved	
0x1C_0004	Reserved		Control Register 0 (CANCTRL0)	Control Register 1 (CANCTRL1)

Table 25-1. FlexCAN Memory Map (continued)

IPSBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x1C_0008	Prescaler Divider (PRESDIV)	Control Register 2 (CANCTRL2)	Free Running Timer (TIMER)	
0x1C_000C	Reserved		Reserved	
0x1C_0010	Rx Global Mask (RXGMASK)			
0x1C_0014	Rx Buffer 14 Mask (RX14MASK)			
0x1C_0018	Rx Buffer 15 Mask (RX15MASK)			
0x1C_0020	Error and Status (ESTAT)		Interrupt Masks (IMASK)	
0x1C_0024	Interrupt Flags (IFLAG)		Rx Error Counter (RXECTR)	Tx Error Counter (TXECTR)
0x1C_0034–0x1C_007F	Reserved		Reserved	
0x1C_0080–0x1C_017F	Message Buffers 0–15			

25.1.2 External Signals

The FlexCAN module/CAN transceiver is composed of two signals: CANTX, which is the serial transmitted data, and CANRX, which is the serial received data.

25.2 The CAN System

A typical CAN system is shown below in Figure 25-2.

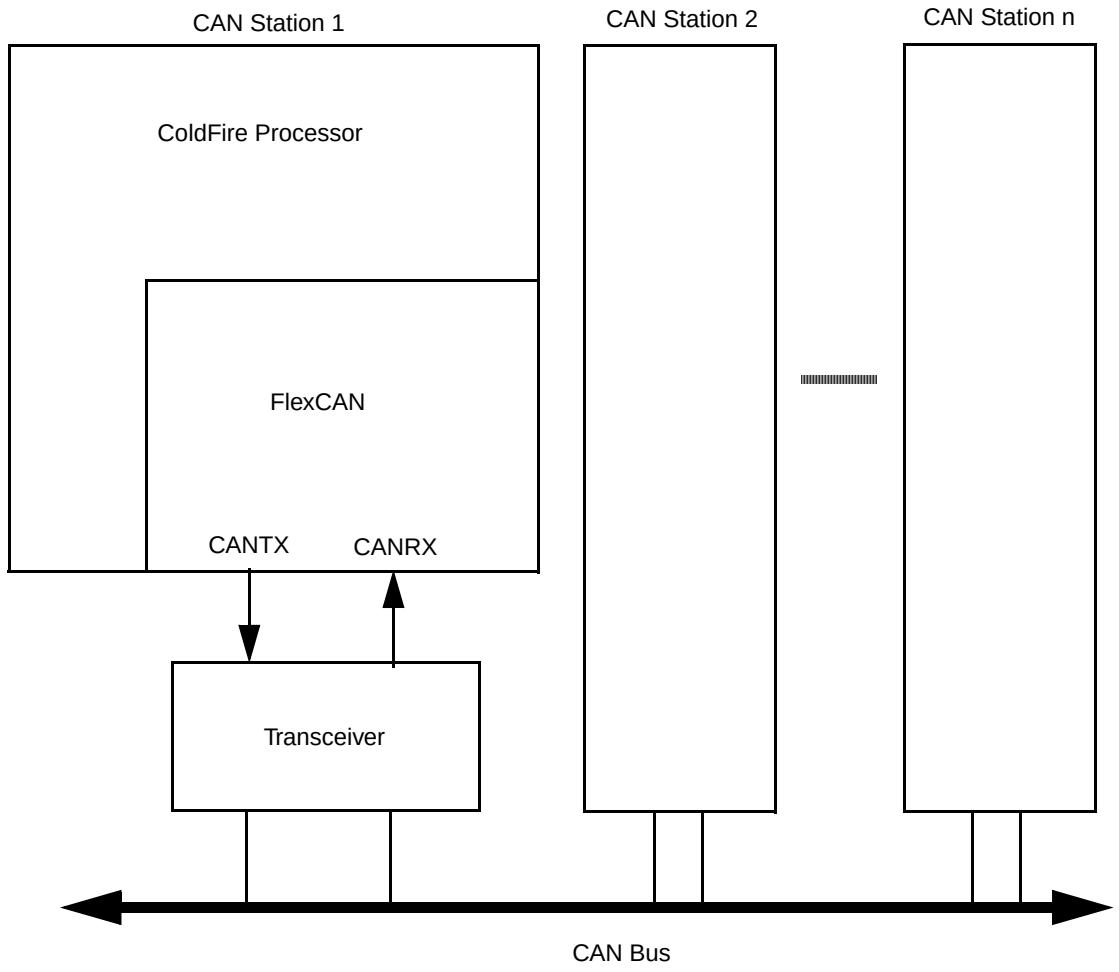


Figure 25-2. Typical CAN system

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.

25.3 Message Buffers

25.3.1 Message Buffer Structure

Figure 25-3 shows the extended (29 bit) ID message buffer structure. Figure 25-4 displays the standard (11 bit) ID message buffer structure.

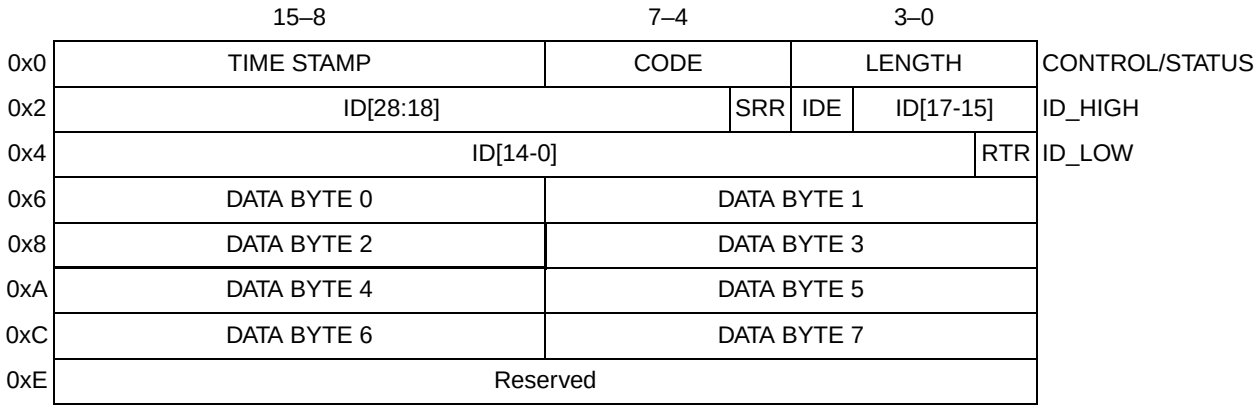


Figure 25-3. Extended ID Message Buffer Structure

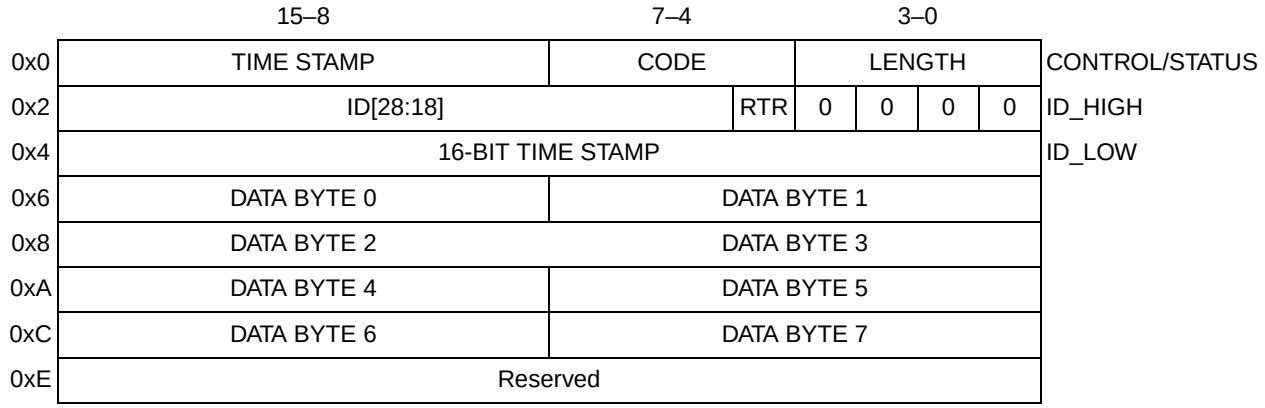


Figure 25-4. Standard ID Message Buffer Structure

25.3.1.1 Common Fields for Extended and Standard Format Frames

Table 25-2 describes the message buffer fields that are common to both extended and standard identifier format frames.

Table 25-2. Common Extended/Standard Format Frames

Field	Description
Time Stamp	Contains a copy of the high byte of the free running timer, which is captured at the beginning of the identifier field of the frame on the CAN bus.
Code	Refer to Table 25-3 and Table 25-4.
Rx Length	Length (in bytes) of the Rx data stored in offset 0x6 through 0xD of the buffer. This field is written by the FlexCAN module, copied from the data length code (DLC) field of the received frame.
Tx Length	Length (in bytes) of the data to be transmitted, located in offset 0x6 through 0xD of the buffer. This field is written by the CPU, and is used as the DLC field value. If remote transmission request (RTR)) = 1, the frame is a remote frame and will be transmitted without the data field, regardless of the value in Tx length.
Data	This field can store up to eight data bytes for a frame. For Rx frames, the data is stored as it is received from the bus. For Tx frames, the CPU provides the data to be transmitted within the frame.
Reserved	This word entry field (16 bits) should not be accessed by the CPU.

Table 25-3. Message Buffer Codes for Receive Buffers

Rx Code Before Rx New Frame	Description	Rx Code After Rx New Frame	Comment
0000	NOT ACTIVE — message buffer is not active.	—	—
0100	EMPTY — message buffer is active and empty.	0010	—
0010	FULL — message buffer is full.	0110	If a CPU read occurs before the new frame, new receive code is 0010.
0110	OVERRUN — second frame was received into a full buffer before the CPU read the first one.		
0101[1]	BUSY — message buffer is now being filled with a new receive frame. This condition will be cleared within 20 cycles.	0010	An empty buffer was filled.
0011[1]		0110	A full buffer was filled.
0111[1]		0110	An overrun buffer was filled.

[1]For transmit message buffers, upon read, the BUSY bit should be ignored.

Table 25-4. Message Buffer Codes for Transmit Buffers

RTR	Initial Tx Code	Description	Code After Successful Transmission
X	1000	Message buffer not ready for transmit.	—
0	1100	Data frame to be transmitted once, unconditionally.	1000
1	1100	Remote frame to be transmitted once, and message buffer becomes an Rx message buffer for data frames.	0100
0	1010[1]	Data frame to be transmitted only as a response to a remote frame.	1010
0	1110	Data frame to be transmitted only once, unconditionally, and then only as a response to remote frame.	1010

[1]When a matching remote request frame is detected, the code for such a message buffer is changed to be 1110.

25.3.1.2 Fields for Extended Format Frames

Table 25-5 describes the message buffer fields used only for extended identifier format frames.

Table 25-5. Extended Format Frames

Field	Description
ID[28:18]/[17:15]	Contains the 14 most significant bits of the extended identifier, located in the ID HIGH word of the message buffer.
Substitute Remote Request (SRR)	Contains a fixed recessive bit, used only in extended format. Should be set to one by the user for Tx buffers. It will be stored as received on the CAN bus for Rx buffers. This is a bit in the ID HIGH word of the message buffer.
ID Extended (IDE)	If extended format frame is used, this field should be set to one. If zero, standard format frame should be used. This is a bit in the ID HIGH word of the message buffer.
ID[14:0]	Bits [14:0] of the extended identifier, located in the ID LOW word of the message buffer.
Remote Transmission Request (RTR)	This bit is located in the least significant bit of the ID LOW word of the message buffer; 0 Data Frame 1 Remote Frame.

25.3.1.3 Fields for Standard Format Frames

Table 25-6 describes the message buffer fields used only for standard identifier format frames.

Table 25-6. Standard Format Frames

Field	Description
ID[28:18]	Contains bits [28:18] of the identifier, located in the ID HIGH word of the message buffer. The four least significant bits in this register (corresponding to the IDE bit and ID[17:15] for an extended identifier message) must all be written as logic zeros to ensure proper operation of the FlexCAN.
RTR	Remote Transmission Request. This bit is located in the ID HIGH word of the message buffer. 0 data frame 1 remote frame. If the FlexCAN transmits this bit as a one and receives it as a zero, an “arbitration loss” is indicated. If the FlexCAN transmits this bit as a zero and receives it as a one, a bit error is indicated. If the FlexCAN transmits a value and receives a matching response, a successful bit transmission is indicated.
16-Bit Time Stamp	The 16-bit time stamp, located in the ID LOW word of the message buffer, is not needed for standard format, and is used in a standard format buffer to store the 16-bit value of the free-running timer. The timer value is captured at the beginning of the identifier field of the frame on the CAN bus.

25.3.2 Message Buffer Memory Map

The message buffer memory map starts at an offset of 0x80 from the FlexCAN’s base address (0x1C_0000). The 256-byte message buffer space is fully used by the 16 message buffer structures.

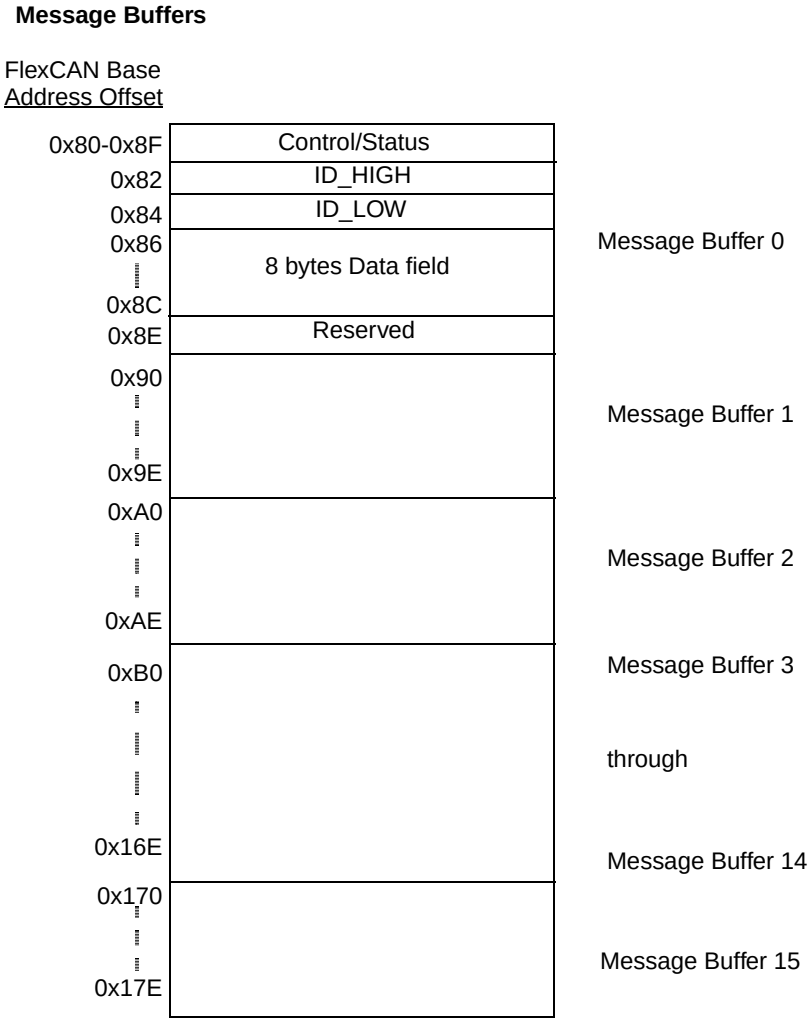


Figure 25-5. FlexCAN Memory Map

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.

NOTE

Note that for both processes, the first CPU action in preparing a MB should be to deactivate it by setting its code field to the proper value. This requirement is mandatory to assure proper operation.

25.4.1 Transmit Process

The CPU prepares or changes an MB for transmission by executing the following steps:

- Writing the Control/Status word to hold Tx MB inactive (code = 1000).
- Writing the ID_HIGH and ID_LOW words.

- Writing the Data bytes
- Writing the Control/Status word (active Code, Length)

NOTE

The first and last steps are mandatory!

Starting from the last step, this MB will participate in the internal arbitration process, which takes place every time the CAN bus is sensed as free by the receiver or at the inter-frame space, and there is at least one MB ready for transmission. This internal arbitration process is intended to select the MB from which the next frame is transmitted.

When this process is over, and there is a ‘winner’ MB for transmission, the frame is transferred to the serial message buffer (SMB) for transmission (Move Out).

While transmitting, the FlexCAN transmits up to eight data bytes, even if the DLC is bigger in value.

At the end of the successful transmission, the value of the free-running timer (which was captured at the beginning of the Identifier field on the CAN bus), is written into the “Time Stamp” field in the MB, the Code field in the Control/Status word of the MB is updated and a status flag is set in the IFLAG register.

25.4.2 Receive Process

The CPU prepares or changes an MB for frame reception by executing the following steps:

- Writing the control/status word to hold Rx MB inactive (code = 0000).
- Writing the ID_HIGH and ID_LOW words.
- Writing the control/status word to mark the Rx MB as active and empty.

NOTE

The first and last steps are mandatory!

Starting from the last step, this MB is an active receive buffer and will participate in the internal matching process, which takes place every time the receiver receives an error-free frame. In this process, all active receive buffers compare their ID value to the newly received one, and if a match occurs, the frame is transferred (Move In) to the first (that is, lowest entry) matching MB. The value of the free-running timer (which was captured at the beginning of the Identifier field on the CAN bus) is written into the “Time Stamp” field in the MB, the ID field, data field (8 bytes at most) and the LENGTH field are stored, the Code field is updated and a status flag is set in the IFLAG register.

The CPU should read a receive frame from its MB in the following way:

- Control/status word (mandatory—activates internal lock for this buffer).
- ID (Optional—needed only if a mask was used).
- Data field word(s).
- Free-running timer (Releases internal lock —optional).

The read of the free-running timer is not mandatory. If not executed, the MB remains locked, unless the CPU starts the read process for another MB. Note that only a single MB is locked at a time. The only mandatory CPU read operation is of the Control/Status word, to assure data coherency. If the BUSY bit is set in the MB code, then the CPU should defer until this bit is negated.

The CPU should synchronize to frame reception by the status flag for the specific MB (see Section 25.5.10, “Interrupt Flag Register (IFLAG)”), and not by the control/status word code field for that MB. This is because polling the control/status word may lock the MB (see above), and the Code may change before the full frame is received into the MB.

Note that the received identifier field is always stored in the matching MB, thus the contents of the identifier field in a MB may change if the match was due to mask.

25.4.2.1 Self-Received Frames

The FlexCAN receives self-transmitted frames if there exists a matching receive MB.

25.4.3 Message Buffer Handling

To maintain data coherency and proper FlexCAN operation, the CPU must obey the rules listed in Section 25.4.1, “Transmit Process” and in Section 25.4.2, “Receive Process.” Deactivation of a message buffer (MB) is a host action that causes that message buffer to be excluded from FlexCAN transmit or receive processes. Any CPU write access to a control/status word of MB structure deactivates that MB, thus excluding it from Rx/Tx processes. Any form of CPU MB structure access within the FlexCAN (other than those specified in Section 25.4.1, “Transmit Process” and in Section 25.4.2, “Receive Process”) may cause the FlexCAN to behave in an unpredictable manner.

The match/arbitration processes are performed only during one period by the FlexCAN. Once a winner or match is determined, there is no re-evaluation whatsoever, in order to ensure that a receive frame is not lost. Two receive MBs or more that hold a matching ID to a received frame do not assure reception in the FlexCAN if the user has deactivated the matching MB after FlexCAN has scanned the second.

25.4.3.1 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.

25.4.3.2 Transmit Message Buffer Deactivation

Any write access to the control/status word of a transmit message buffer during the process of selecting a message buffer for transmission immediately deactivates that message buffer, removing it from the transmission process.

If the user deactivates the transmit MB while a message is being transferred from a transmit message buffer to a SMB the message will not be transmitted.

If the user deactivates the transmit message buffer after the message is transferred to the SMB, the message will be transmitted, but no interrupt will be requested and the transmit code will not be updated.

If a message buffer containing the lowest ID is deactivated while that message is undergoing the internal arbitration process to determine which message should be sent, then that message may not be transmitted.

25.4.3.3 Receive Message Buffer Deactivation

Any write access to the control/status word of a receive message buffer during the process of selecting a message buffer for reception immediately deactivates that message buffer, removing it from the reception process.

If a receive message buffer is deactivated while a message is being transferred into it, the transfer is halted and no interrupt is requested. If this occurs, that receive message buffer may contain mixed data from two different frames.

Data should never be written into a receive message buffer. If this is done while a message is being transferred from an SMB, the control/status word will reflect a full or overrun condition, but no interrupt will be requested.

25.4.3.4 Locking and Releasing Message Buffers

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 which 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 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.

25.4.4 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 to one. 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 which 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.

25.4.5 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.

25.4.6 Time Stamp

The value of the free-running 16-bit timer 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 time stamp entry of the receive message buffer at the time the message is written into that buffer. For a message being transmitted, the time stamp 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.

25.4.7 Listen-Only Mode

In listen-only mode, the FlexCAN module is able to receive messages without giving an acknowledgment. Whenever the module enters this mode the status of the Error Counters is frozen and the FlexCAN module operates like in error passive mode. Since the module does not influence the CAN bus in this mode the host device is capable of functioning like a monitor or for automatic bit-rate detection.

25.4.8 Bit Timing

The FlexCAN module uses three 8-bit registers to set up the bit timing parameters required by the CAN protocol. Control registers 1 and 2 (CANCTRL1, CANCTRL2) contain the PROPSEG, PSEG1, PSEG2, and the RJW fields which allow the user to configure the bit timing parameters. The prescaler divide register (PRESDIV) allows the user to select the ratio used to derive the S-clock from the system clock. The time quanta clock operates at the S-clock frequency. Table 25-7 provides examples of system clock, CAN bit rate, and S-clock bit timing parameters.

Table 25-7. Examples of System Clock/CAN Bit-Rate/S-Clock

System Clock Freq (Mhz)	Can bit-rate (Mhz)	Possible S-Clock Freq (Mhz)	Possible number of time-quanta/bit	Pre-Scaler programed value + 1	Comments
48	1	8,12,24	8,12,24	3,2,1	Min 8 time-quanta Max 25 time-quanta
40	1	10,20	10,20	2,1	
32	1	8,16	8,16	2,1	
48	0.125	1,1.5,2,3	8,12,16,24	24,16,12,8	
40	0.125	1,2,2.5	8,16,20	20,10,8	
32	0.125	1,2	8,16	16,8	

25.4.8.1 Configuring the FlexCAN Bit Timing

The following considerations must be observed when programming bit timing functions.

- If the programmed PRESDIV value results in a single system clock per one time quantum, then the PSEG2 field in CANCTRL1 register should not be programmed to zero.
- If the programmed PRESDIV value results in a single system clock per one time quantum, then the information processing time (IPT) equals three time quanta, otherwise it equals two time quanta. If PSEG2 equals two, then the FlexCAN transmits one time quantum late relative to the scheduled sync segment.
- If the prescaler and bit timing control fields are programmed to values that result in fewer than ten system clock periods per CAN bit time and the CAN bus loading is 100%, anytime the rising edge of a start-of-frame (SOF) symbol transmitted by another node occurs during the third bit of the intermission between messages, the FlexCAN may not be able to prepare a message buffer for transmission in time to begin its own transmission and arbitrate against the message which transmitted the early SOF.
- The FlexCAN bit time must be programmed to be greater than or equal to nine system clocks, or correct operation is not guaranteed.

25.4.9 FlexCAN Error Counters

There are two error counters in the FlexCAN: transmit error counter (TXECTR), and receive error counter (RXCTR). The rules for increasing and decreasing these counters are described in the CAN protocol, and are fully implemented in the FlexCAN. Each counter comprises the following:

- 8 bit up/down counter
- Increment by 8 (Rx_Err_Counter also by 1)
- Decrement by 1
- Avoid decrement when equal to zero
- Rx_Err_Counter preset to a value $119 \leq x \leq 127$
- Value after reset = zero
- Detect values for Error Passive, Bus Off and Error Active transitions and for alerting the host.

Both counters are read only (except for Test/Freeze/Halt modes).

The 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 TXCTR or RXCTR increases to be greater than or equal to 128, the FCS field in the error status register is updated to reflect it (set Error Passive state).
- If the FlexCAN state is Error Passive, and either TXCTR counter or RXCTR then decrements to a value less than or equal to 127 while the other already satisfies this condition, the ESTAT[FCS] field is updated to reflect it (set Error Active state).
- If the value of the TXCTR increases to be greater than 255, the ESTAT[FCS] field is updated to reflect it (set Bus Off state) and an interrupt may be issued. The value of TXCTR is then reset to zero.
- If the FlexCAN state is Bus_Off, then TXCTR, together with an internal counter are cascaded to count the 128 occurrences of 11 consecutive recessive bits on the bus. Hence, TXCTR is reset to zero, and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the TXCTR. When TXCTR reaches the value of 128, ESTAT[FCS] 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, but does NOT affect the TXCTR value.
- If during system start-up, only one node is operating, then its TXCTR increases with each message it’s trying to transmit as a result of ACK_ERROR. A transition to bus state Error Passive should be executed as described, while this device never enters the Bus_Off state.
- If the RXCTR increases to a value greater than 127, it is no longer incremented, 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, in order to return to Error Active state.

25.4.10 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 following is a generic initialization sequence for the FlexCAN:

1. Initialize all operation modes
   a) Initialize the transmit and receive pin modes in control register 0 (CANCTRL0).
   b) Initialize the bit timing parameters PROPSEG, PSEGS1, PSEG2, and RJW in control registers 1 and 2 (CANCTRL[1:2]).
   c) Select the S-clock rate by programming the PRESDIV register.
   d) Select the internal arbitration mode (LBUF bit in CANCTRL1).
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 mask registers for acceptance mask as needed
4. Initialize FlexCAN interrupt handler
   a) Initialize the interrupt configuration register (ICR*n*) with a specific request level and vector base address.
   b) Set the required mask bits in the IMASK register (for all message buffer interrupts), in CANCTRL0 (for bus off and error interrupts), and in CANMCR for the WAKE interrupt.

5. Negate the HALT bit in the module configuration register
   a) At this point, the FlexCAN will attempt to synchronize with the CAN bus.

NOTE

In both the transmit and receive processes, the first action in preparing a message buffer should be to deactivate the buffer by setting its code field to the proper value. This requirement is mandatory to assure data coherency.

25.4.11 Special Operating Modes

25.4.11.1 Debug Mode

Debug mode is entered by setting the HALT bit in the CANMCR, or by assertion of the $\overline{\text{BKPT}}$ line. In both cases, the FRZ bit in CANMCR must also be set to allow HALT or $\overline{\text{BKPT}}$ to place the FlexCAN in debug mode.

Once entry into debug 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. 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 are set.
- The CPU is allowed to read and write the error counter registers.

After engaging one of the mechanisms to place the FlexCAN in debug 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.

To exit debug mode, the $\overline{\text{BKPT}}$ line must be negated or the HALT bit in CANMCR must be cleared.

Once debug 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.

25.4.11.2 Low-Power Stop Mode for Power Saving

Before entering low-power stop mode, the FlexCAN will wait for the CAN bus to be in an idle state, or for the third bit of intermission to be recessive. The FlexCAN then waits for the completion of all internal activity (except in the CAN bus interface) to be complete. Afterwards, the following events occur:

- The FlexCAN shuts down its clocks, stopping most internal circuits, thus achieving maximum power savings.
- The bus interface unit continues to operate, allowing the CPU to access the module configuration register.
- The FlexCAN ignores its Rx pins and drives its Tx pins as recessive.
- The FlexCAN loses synchronization with the CAN bus, and the STOPACK and NOTRDY bits in the module configuration register are set.

To exit low-power stop mode:

- Reset the FlexCAN either by asserting $\overline{\text{RSTI}}$ or by setting the SOFTRST bit CANMCR.
- Clear the STOP bit in CANMCR.
- The FlexCAN module can optionally exit low-power stop mode via the self-wake mechanism. If the SELFWAKE bit in CANMCR was set at the time the FlexCAN entered stop mode, then upon detection of a recessive to dominant transition on the CAN bus, the FlexCAN clears the STOP bit in CANMCR and its clocks begin running.

When in low-power stop mode, a recessive to dominant transition on the CAN bus causes the WAKEINT bit in the error and status register (ESTAT) to be set. This event can generate an interrupt if the WAKEMSK bit in CANMCR is set.

Consider the following notes regarding low-power stop mode:

- When the self-wake mechanism activates, the FlexCAN tries to receive the frame that woke it up. (It assumes that the dominant bit detected is a start-of-frame bit). It will not arbitrate for the CAN bus at this time.
- The CPU should disable all interrupts in the FlexCAN before entering low-power stop mode. Otherwise it may be interrupted while in STOP mode upon a non wake-up condition; If desired, the WAKEMASK bit should be set to enable the WAKEINT.
- If the STOP bit is set while the FlexCAN is in the bus off state, then the FlexCAN will enter low-power stop mode and stop counting recessive bit times. The count will continue when STOP is cleared.
- To place the FlexCAN in low-power stop mode with the self-wake mechanism engaged, write to CANMCR with both STOP and SELFWAKE set, then wait for the FlexCAN to set the STOPACK bit.
- To take the FlexCAN out of low-power stop mode when the self-wake mechanism is enabled, write to CANMCR with both STOP and SELFWAKE clear, then wait for the FlexCAN to clear the STOPACK bit.
- The SELFWAKE bit should not be set after the FlexCAN has already entered low-power stop mode.
- If both STOP and SELFWAKE are set and a recessive to dominant edge immediately occurs on the CAN bus, the FlexCAN may never set the STOPACK bit, and the STOP bit will be cleared.
- To prevent old frames from being sent when the FlexCAN awakes from low-power stop mode via the self-wake mechanism, disable all transmit sources, including transmit buffers configured for remote request responses, before placing the FlexCAN in low-power stop mode.
- If the FlexCAN is in debug mode when the STOP bit is set, the FlexCAN will assume that debug mode should be exited. As a result, it will try to synchronize with the CAN bus, and only then will it await the conditions required for entry into low-power stop mode.
- Unlike other modules, the FlexCAN does not come out of reset in low-power stop mode. The basic FlexCAN initialization procedure (see Section 25.4.10, “FlexCAN Initialization Sequence”) should be executed before placing the module in low-power stop mode.
- If the FlexCAN is in low-power stop mode with the self-wake mechanism engaged and is operating with a single system clock per time quantum, there can be extreme cases in which FlexCAN wake-up on recessive to dominant edge may not conform to the CAN protocol. FlexCAN synchronization will be shifted one time quantum from the wake-up event. This shift lasts until the next recessive to dominant edge, which resynchronizes the FlexCAN to be in conformance with the CAN protocol. The same holds true when the FlexCAN is in auto-power save mode and awakens on a recessive to dominant edge.

25.4.11.3 Auto-Power Save Mode

Auto-power save mode enables normal operation with optimized power savings. Once the auto-power save (APS) bit in CANMCR is set, the FlexCAN looks for a set of conditions in which there is no need for its clocks to be running. If these conditions are met, the FlexCAN stops its clocks, thus saving power. The following conditions will activate auto-power save mode.

- No Rx/Tx frame in progress.
- No transfer of Rx/Tx frames to and from an SMB, and no Tx frame awaiting transmission in any message buffer.
- No CPU access to the FlexCAN module.
- The FlexCAN is not in debug mode, low-power stop mode, or the bus off state.

While its clocks are stopped, if the FlexCAN senses that any one of the aforementioned conditions is no longer true, it restarts its clocks. The FlexCAN then continues to monitor these conditions and stops/restarts its clocks accordingly.

25.4.12 Interrupts

The module can generate up to 19 interrupt sources (16 interrupts due to message buffers and 3 interrupts due to Bus-off, Error and Wake-up). Each one of the message buffers can be an interrupt source, if its corresponding IMASK bit is set.

There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception, and thus its interrupt routine can be fixed at compilation time. Each of the buffers is assigned a bit in the IFLAG register. The bit is set when the corresponding buffer completes a successful transmission or reception, and cleared when the CPU reads the interrupt flag register (IFLAG) while the associated bit is set, and then writes it back as ‘1’ (and no new event of the same type occurs between the read and the write actions).

The other 3 interrupt sources (Bus-off, Error and Wake-up) act in the same way, and are located in the Error & Status register. The Bus-off and Error interrupt mask bits are located in the CANCTRL0 register, and the Wake-up interrupt mask bit is located in the CANMCR.

25.5 Programmer’s Model

This section describes the registers in the FlexCAN module.

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 HALT bit in the FlexCAN module configuration register (CANMCR). The FlexCAN responds by setting the CANMCR[NOTRDY] bit. Additionally, the control registers can be modified while the MCU is in background debug mode.

25.5.1 CAN Module Configuration Register (CANMCR)

	15	14	13	12	11	10	9	8
Field	STOP	FRZ	—	HALT	NOTRDY	WAKEMSK	SOFTRST	FRZACK
Reset	0101_1001							
R/W	R/W							

	7	6	5	4	3	0
Field	SUPV	SELFWAKE	APS	STOPACK	—	
Reset	1000_0000					
R/W	R/W					
Address	IPSBAR + 0x1C_0000					

Figure 25-6. CAN Module Configuration Register (CANMCR)

Table 25-8 describes the CANMCR fields.

Table 25-8. CANMCR Field Descriptions

Bits	Name	Description
15	STOP	Low-power stop mode enable. The STOP bit may only be set by the CPU. It may be cleared either by the CPU or by the FlexCAN, if the SELFWAKE bit is set. 0 Enable FlexCAN clocks 1 Disable FlexCAN clocks
14	FRZ	FREEZE assertion response. When FRZ = 1, the FlexCAN can enter debug mode when the $\overline{\text{BKPT}}$ line is asserted or the HALT bit is set. Clearing this bit field causes the FlexCAN to exit debug mode. Refer to Section 25.4.11.1, “Debug Mode” for more information. 0 FlexCAN ignores the $\overline{\text{BKPT}}$ signal and the HALT bit in the module configuration register. 1 FlexCAN module enabled to enter debug mode.
13	—	Reserved
12	HALT	Halt FlexCAN S-Clock. Setting the HALT bit has the same effect as assertion of the $\overline{\text{BKPT}}$ signal on the FlexCAN without requiring that $\overline{\text{BKPT}}$ be asserted. This bit is set to one after reset. It 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. When HALT is set, write access to certain registers and bits that are normally read-only is allowed. 0 The FlexCAN operates normally 1 FlexCAN enters debug mode if FRZ = 1
11	NOTRDY	FlexCAN not ready. This bit indicates that the FlexCAN is either in low-power stop mode or debug mode. This bit is read-only and is set only when the FlexCAN enters low-power stop mode or debug mode. It is cleared once the FlexCAN exits either mode, either by synchronization to the CAN bus or by the self-wake mechanism. 0 FlexCAN has exited low-power stop mode or debug mode. 1 FlexCAN is in low-power stop mode or debug mode.
10	WAKEMSK	Wakeup interrupt mask. The WAKEMSK bit enables wake-up interrupt requests. 0 Wake up interrupt is disabled. 1 Wake up interrupt is enabled.

Table 25-8. CANMCR Field Descriptions (continued)

Bits	Name	Description
9	SOFTRST	Soft reset. When this bit is asserted, the FlexCAN resets its internal state machines (sequencer, error counters, error flags, and timer) and the host interface registers (CANMCR, CANICR, CANTCR, IMASK, and IFLAG). The configuration registers that control the interface with the CAN bus are not changed (CANCTRL[0:2] and PRESDIV). Message buffers and receive message masks are also not changed. This allows SOFTRST to be used as a debug feature while the system is running. Setting SOFTRST also clears the STOP bit in CANMCR. After setting SOFTRST, allow one complete bus cycle to elapse for the internal FlexCAN circuitry to completely reset before executing another access to CANMCR. The FlexCAN clears this bit once the internal reset cycle is completed. 0 Soft reset cycle completed 1 Soft reset cycle initiated
8	FRZACK	FlexCAN disable. When the FlexCAN enters debug mode, it sets the FRZACK bit. This bit should be polled to determine if the FlexCAN has entered debug mode. When debug mode is exited, this bit is negated once the FlexCAN prescaler is enabled. This is a read-only bit. 0 The FlexCAN has exited debug mode and the prescaler is enabled. 1 The FlexCAN has entered debug mode, and the prescaler is disabled.
7	SUPV	Supervisor/user data space. The SUPV bit 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.
6	SELF-WAKE	Self wake enable. This bit allows the FlexCAN to wake up when bus activity is detected after the STOP bit is set. If this bit is set when the FlexCAN enters low-power stop mode, the FlexCAN will monitor the bus for a recessive to dominant transition. If a recessive to dominant transition is detected, the FlexCAN immediately clears the STOP bit and restarts its clocks. If a write to CANMCR with SELFWAKE set occurs at the same time a recessive-to-dominant edge appears on the CAN bus, the bit will not be set, and the module clocks will not stop. The user should verify that this bit has been set by reading CANMCR. Refer to Section 25.4.11.2, “Low-Power Stop Mode for Power Saving” for more information on entry into and exit from low-power stop mode. 0 Self wake disabled. 1 Self wake enabled.
5	APS	Auto-power save. The APS bit allows the FlexCAN to automatically shut off its clocks to save power when it has no process to execute, and to automatically restart these clocks when it has a task to execute without any CPU intervention. 0 Auto-power save mode disabled; clocks run normally. 1 Auto-power save mode enabled; clocks stop and restart as needed.
4	STOPACK	Stop acknowledge. When the FlexCAN is placed in low-power stop mode and shuts down its clocks, it sets the STOPACK bit. This bit should be polled to determine if the FlexCAN has entered low-power stop mode. When the FlexCAN exits low-power stop mode, the STOPACK bit is cleared once the FlexCAN’s clocks are running. 0 The FlexCAN is not in low-power stop mode and its clocks are running. 1 The FlexCAN has entered low-power stop mode and its clocks are stopped
3–0	—	Reserved, should be cleared.

25.5.2 FlexCAN Control Register 0 (CANCTRL0)

	7	6	5	4	3	2	1	0
Field	BOFFMSK	ERRMSK	—			RXMODE	TXMODE	
Reset	0000_0000							
R/W	R/W							
Address	IPSBAR + 0x1C_0006							

Figure 25-7. FlexCAN Control Register 0 (CANCTRL0)

Table 25-9 describes the CANCTRL0 fields.

Table 25-9. CANCTRL0 Field Descriptions

Bits	Name	Description
7	BOFFM-SK	Bus off interrupt mask. The BOFF MASK bit provides a mask for the bus off interrupt. 0 Bus off interrupt disabled. 1 Bus off interrupt enabled.
6	ERRMSK	Error interrupt mask. The ERRMSK bit provides a mask for the error interrupt. 0 Error interrupt disabled. 1 Error interrupt enabled.
5–3	—	Reserved
2	RXMODE	Receive pin configuration control. This bit determines the polarity of the CANRX pin. 0 A logical ‘0’ is interpreted as a dominant bit; a logical ‘1’ is interpreted as a recessive bit. 1 A logical ‘1’ is interpreted as a dominant bit; a logical ‘0’ is interpreted as a recessive bit.
1–0	TXMODE	Transmit pin configuration control. This bit field controls the configuration of the CANTX pin. See Table 25-10.

Table 25-10. Transmit Pin Configuration

TXMODE[1:0]	Transmit Pin Configuration
00	Full CMOS[1]; positive polarity (CANTX= 0 is a dominant level)
01	Full CMOS[1]; negative polarity (CANTX = 1 is a dominant level)
1X	Open drain[2]; positive polarity

1 Full CMOS drive indicates that both dominant and recessive levels are driven by the chip.

2 Open drain drive indicates that only a dominant level is driven by the chip. During a recessive level, the CANTX pin is disabled (three stated), and the electrical level is achieved by external pull-up/pull-down devices. The assertion of both Tx mode bits causes the polarity inversion to be cancelled (open drain mode forces the polarity to be positive).

25.5.3 FlexCAN Control Register 1 (CANCTRL1)

	7	6	5	4	3	2	1	0
Field	SAMP	—	TSYNC	LBUF	LOM	PROPSEG		
Reset	0000_0000							
R/W	R/W							
Address	IPSBAR + 0x1C_0007							

Figure 25-8. FlexCAN Control Register 1 (CANCTRL1)

Table 25-11 describes the CANCTRL1 fields.

Table 25-11. CANCTRL1 Field Descriptions

Bits	Name	Description
7	SAMP	Sampling mode. The SAMP bit 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.
6	—	Reserved, should be cleared.
5	TSYNC	Timer synchronize mode. The TSYNC bit 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. The LBUF bit defines the transmit-first scheme. 0 Message buffer with lowest ID is transmitted first. 1 Lowest numbered buffer is transmitted first.
3	LOM	Listen Only Mode. In this mode the FlexCAN is able to receive messages without giving an acknowledgment or being active on the bus. 0 Regular operation (listen only mode off). 1 Enable listen only mode.
2–0	PROPSEG	Propagation segment time. PROPSEG defines the length of the propagation segment in the bit time. The valid programmed values are 0 to 7. The propagation segment time is calculated as follows: Propagation Segment Time = (PROPSEG + 1) Time Quanta where 1 Time Quantum = 1 Serial Clock (S-Clock) Period

25.5.4 Prescaler Divide Register (PRESDIV)

	7						0
Field	PRES_DIV						
Reset	0000_0000						
R/W	R/W						
Address	IPSBAR + 0x1C_0008						

Figure 25-9. Prescaler Divide Register (PRESDIV)

Table 25-12 describes the PRESDIV fields.

Table 25-12. PRESDIV Field Descriptions

Bits	Name	Description
7–0	PRES_DIV	Prescaler divide factor. PRESDIV determines the ratio between the system clock frequency and the serial clock (S-clock). The S-clock is determined by the following calculation: $\text{S-clock} = \frac{f_{sys}}{2(\text{PRESDIV} + 1)}$ The reset value of PRESDIV is 0x00, which forces the S-clock to default to the same frequency as the system clock. The valid programmed values are 0 through 255. See Section 25.4.8, “Bit Timing” for more information.

25.5.5 FlexCAN Control Register 2 (CANCTRL2)

	7	6	5	3	2	0
Field	RJW		PSEG1		PSEG2	
Reset	0000_0000					
R/W	R/W					
Address	IPSBAR + 0x1C_0009					

Figure 25-10. FlexCAN Control Register 2 (CANCTRL2)

Table 25-13 describes the CANCTRL2 fields.

Table 25-13. CANCTRL2 Field Descriptions

Bits	Name	Description
7–6	RJW	Resynchronization jump width. The RJW field defines the maximum number of time quanta a bit time may be changed during resynchronization. The valid programmed values are 0 through 3. The resynchronization jump width is calculated as follows: Resynchronizaton Jump Width = (RJW + 1) Time Quanta

Table 25-13. CANCTRL2 Field Descriptions (continued)

Bits	Name	Description
5–3	PSEG 1	PSEG1[2:0] — Phase buffer segment 1. The PSEG1 field defines the length of phase buffer segment 1 in the bit time. The valid programmed values are 0 through 7. The length of phase buffer segment 1 is calculated as follows: Phase Buffer Segment 1 = (PSEG1 + 1) Time Quanta
2–0	PSEG 2	PSEG2 — Phase Buffer Segment 2. The PSEG2 field defines the length of phase buffer segment 2 in the bit time. The valid programmed values are 0 through 7. The length of phase buffer segment 2 is calculated as follows: Phase Buffer Segment 2 = (PSEG2 + 1) Time Quanta

25.5.6 Free Running Timer (TIMER)

	15	0
Field	TIMER	
Reset	0000_0000_0000_0000	
R/W	R/W	
Address	IPSBAR + 0x1C_000A	

Figure 25-11. Free Running Timer (TIMER)

Table 25-14 describes the TIMER fields.

Table 25-14. TIMER Field Descriptions

Bits	Name	Description
15–0	TIMER	The free running timer counter can be read and written by the CPU. The timer starts from zero after reset, counts linearly to 0xFFFF, and wraps around. The timer is clocked by the FlexCAN bit-clock. During a message, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it increments at the nominal bit rate. The timer value is captured at the beginning of the identifier field of any frame on the CAN bus. The captured value is written into the “time stamp” entry in a message buffer after a successful reception or transmission of a message.

25.5.7 Rx Mask Registers

These registers are used as acceptance masks for received frame IDs. 3 masks are defined: A global mask, used for Rx buffers 0-13, and 2 more separate masks for buffers 14 and 15.

Mask bit = 0: The corresponding incoming ID bit is “don’t care”.

Mask 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 mask registers should NOT be changed while in normal operation, as locked frames which matched a MB through a mask, may be transferred into the MB (upon release) but may no longer match.

Table 25-15. 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 0 1 1 1 1	0		
MB5-Id	0 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 0 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	3[1]
Rx_Msg in	1 1 1 1 1 1 1 1 0 0 1	0		2[2]
Rx_Msg in	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 0	3
Rx_Msg in	0 1 1 1 1 1 1 1 0 0 0	0		4
Rx_Msg in	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	5
Rx_14_Mask	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	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	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	14[7]

1 Match for Extended Format (MB3).
2 Match for Standard Format. (MB2).
3 Un-Match for MB3 because of ID0.
4 Un-Match for MB2 because of ID28.
5 Un-Match for MB3 because of ID28, Match for MB14.
6 Un-Match for MB14 because of ID27.
7 Match for MB14.

25.5.7.1 Receive Mask Registers (RXGMASK, RX14MASK, RX15MASK)

The Rx global mask register (RXGMASK) is composed of 4 bytes. The mask bits are applied to all Rx-Identifiers excluding Rx-buffers 14-15, that have their specific Rx-mask registers (RX14MASK and RX15MASK).

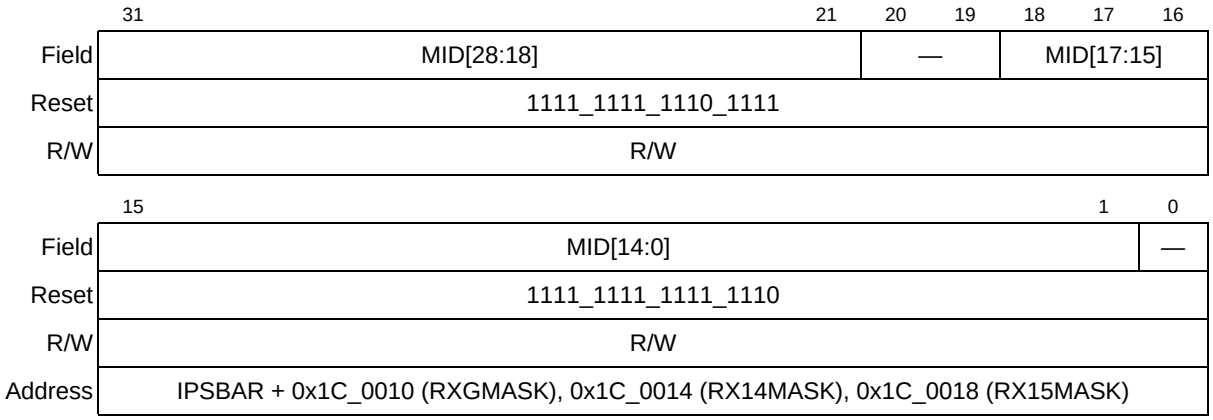


Figure 25-12. Rx Mask Registers (RXGMASK, RX14MASK, and RX15MASK)

Table 25-16. RXGMASK, RX14MASK, and RX15MASK Field Descriptions

Bits	Name	Description
31–21	MID	Mask ID. MID[28:18] are used to mask standard or extended format frames. 0 corresponding incoming ID bit is “don’t care”. 1 corresponding ID bit is checked against the incoming ID bit, to see if a match exists.
20	—	Reserved. The IDE bit of a received frame is always compared. Its location in the mask (bit 19) is always 1, regardless of any CPU write to this bit.
19	—	Reserved. The RTR/SRR bit of a received frame is never compared to the corresponding bit in the MB ID field. Note, however, that remote request frames (RTR = 1) are never received into MBs. RTR mask bits locations in the mask (bits 20 and 0) are always read as ’0’, regardless of any CPU write to these bits.
18–1	MID	Mask ID. MID[17:0] are only used to mask extended format frames. 0 corresponding incoming ID bit is “don’t care”. 1 corresponding ID bit is checked against the incoming ID bit, to see if a match exists.
0	—	Reserved. The RTR/SRR bit of a received frame is never compared to the corresponding bit in the MB ID field. Note, however, that remote request frames (RTR = 1) are never received into MBs. RTR mask bits locations in the mask (bits 20 and 0) are always read as ’0’, regardless of any CPU write to these bits.

25.5.8 FlexCAN Error and Status Register (ESTAT)

ESTAT reflects various error conditions, some general status of the device, and is the source of three interrupts to the host. The reported error conditions (bits 15:10) are those occurred since the last time the host read this register. The read action clears these bits to 0.

All the bits in this register are read only, except for BOFF_INT, WAKE_INT and ERR_INT, which are interrupt sources and can be written by the host to ‘1’. Section 25.4.12, “Interrupts.”

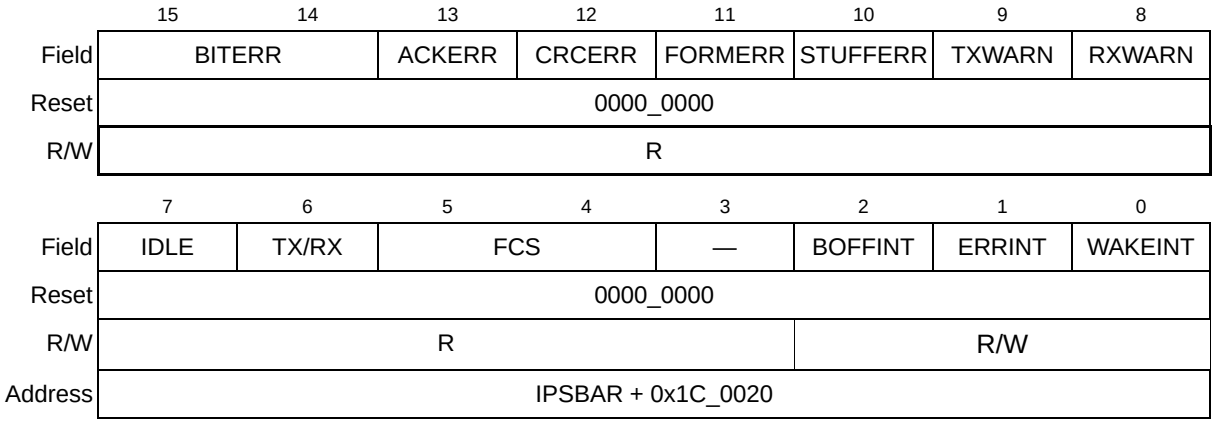


Figure 25-13. FlexCAN Error and Status Register (ESTAT)

Table 25-17 describes the ESTAT fields.

Table 25-17. ESTAT Field Descriptions

Bits	Name	Description
15–14	BITERR	Transmit bit error. The BITERR[1:0] field is used to indicate when a transmit bit error occurs. 00 No transmit bit error 01 At least one bit sent as dominant was received as recessive 10 At least one bit sent as recessive was received as dominant 11 Reserved 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.
13	ACKERR	Acknowledge error. The ACKERR bit 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.
12	CRCERR	Cyclic redundancy check error. The CRCERR bit indicates whether or not the CRC of the last transmitted or received message was valid. 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	FORMERR	Message format error. The FORMERR bit indicates whether or not the message format of the last transmitted or received message was correct. 0 No format error was detected since the last read of this register. 1 A format error was detected since the last read of this register.
10	STUFERR	Bit stuff error. The STUFFERR bit indicates whether or not the bit stuffing that occurred in the last transmitted or received message was correct. 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	TXWARN	Transmit error status flag. The TXWARN status flag reflects the status of the FlexCAN transmit error counter. 0 Transmit error counter $<$ 96. 1 Transmit error counter $\geq$ 96.

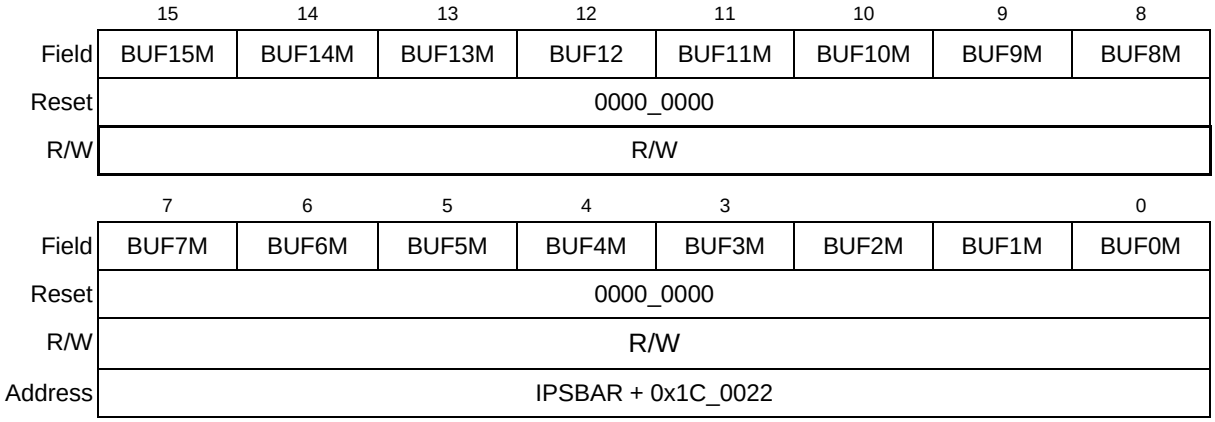
Table 25-17. ESTAT Field Descriptions (continued)

Bits	Name	Description
8	RXWARN	Receiver error status flag. The RXWARN status flag reflects the status of the FlexCAN receive error counter. 0 Receive error counter < 96. 1 Receive error counter ≥ 96.
7	IDLE	Idle status. The IDLE bit indicates when there is activity on the CAN bus. 0 The CAN bus is not idle. 1 The CAN bus is idle.
6	TX/RX	Transmit/receive status. The TX/RX bit indicates when the FlexCAN module is transmitting or receiving a message. TX/RX 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	FCS	Fault confinement state. The FCS[1:0] field describes the state of the FlexCAN. If the SOFTRST bit in CANMCR is asserted while the FlexCAN is in the bus off state, the error and status register is reset, including FCS[1:0]. However, as soon as the FlexCAN exits reset, FCS[1:0] bits will again reflect the bus off state. Refer to Section 25.5.11, “FlexCAN Receive Error Counter (RXECTR)” for more information on entry into and exit from the various fault confinement states. 00 Error active 01 Error passive 1X Reserved
3	—	Reserved, should be cleared.
2	BOFFINT	Bus off interrupt. The BOFFINT bit is used to request an interrupt when the FlexCAN enters the bus off state. To clear this bit, first read it as a one, then write a one. Writing zero has no effect. 0 No bus off interrupt requested. 1 When the FlexCAN state changes to bus off, this bit is set, and if the BOFFMSK bit in CANCTRL0 is set, an interrupt request is generated. This interrupt is not requested after reset.
1	ERRINT	Error interrupt. The ERRINT bit is used to request an interrupt when the FlexCAN detects a transmit or receive error. To clear this bit, first read it as a one, then write a one. Writing zero has no effect. 0 No error interrupt request. 1 If an event which causes one of the error bits in the error and status register to be set occurs, the error interrupt bit is set. If the ERRMSK bit in CANCTRL0 is set, an interrupt request is generated.
0	WAKEINT	Wake interrupt. The WAKEINT bit indicates that bus activity has been detected while the FlexCAN module is in low-power stop mode. To clear this bit, first read it as a one, then write a one. Writing zero has no effect. 0 No wake interrupt requested. 1 When the FlexCAN is in low-power stop mode and a recessive to dominant transition is detected on the CAN bus, this bit is set. If the WAKEMSK bit is set in CANMCR, an interrupt request is generated.

25.5.9 Interrupt Mask Register (IMASK)

IMASK 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 bit is set).

The interrupt mask register contains two 8-bit fields: bits 15-8 (IMASK_H) and bits 7-0 (IMASK_L). The register can be accessed by the master as a 16-bit register, or each byte can be accessed individually using an 8-bit (byte) access cycle.



	15	14	13	12	11	10	9	8
Field	BUF15M	BUF14M	BUF13M	BUF12	BUF11M	BUF10M	BUF9M	BUF8M
Reset	0000_0000							
R/W	R/W							

	7	6	5	4	3			0
Field	BUF7M	BUF6M	BUF5M	BUF4M	BUF3M	BUF2M	BUF1M	BUF0M
Reset	0000_0000							
R/W	R/W							
Address	IPSBAR + 0x1C_0022							

Figure 25-14. Interrupt Mask Register (IMASK)

Table 25-18 describes the IMASK fields.

Table 25-18. IMASK Field Descriptions

Bits	Name	Description
15–0	BUF*n*M	IMASK contains one interrupt mask bit per buffer. It allows 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.

25.5.10 Interrupt Flag Register (IFLAG)

IFLAG contains one interrupt flag bit per buffer. Each successful transmission/reception sets the corresponding IFLAG bit and, if the corresponding IMASK bit is set, will generate an interrupt.

This register contains two 8-bit fields: bits 15-8 (IFLAG_H) and bits 7-0 (IFLAG_L). The register can be accessed by the master as a 16-bit register, or each byte can be accessed individually using an 8-bit (byte) access cycle.

	15	14	13	12	11	10	9	8
Field	BUF15I	BUF14I	BUF13I	BUF1I	BUF11I	BUF10I	BUF9I	BUF8I
Reset	0000_0000							
R/W	R/w							

	7	6	5	4	3			0
Field	BUF7I	BUF6I	BUF5I	BUF4I	BUF3I	BUF2I	BUF1I	BUF0I
Reset	0000_0000							
R/W	R/W							
Address	IPSBAR + 0x1C_0024							

Figure 25-15. Interrupt Flag Register (IFLAG)

Table 25-19 describes the IFLAG fields.

Table 25-19. IFLAG Field Descriptions

Bits	Name	Description
15–0	BUF*n*I	IFLAG contains one interrupt flag bit per buffer. Each successful transmission/reception sets the corresponding IFLAG bit and, if the corresponding IMASK bit is set, an interrupt request will be generated. To clear an interrupt flag, first read the flag as a one, and then write it as a one. Should a new flag setting event occur between the time that the CPU reads the flag as a one and writes the flag as a zero, the flag is not cleared. This register can be written to zeros only. 0 The interrupt for the corresponding buffer is disabled. 1 The interrupt for the corresponding buffer is enabled.

25.5.11 FlexCAN Receive Error Counter (RXECTR)

	7							0
Field	RXECTR							
Reset	0000_0000							
R/W	R							
Address	IPSBAR + 0x1C_0026							

Figure 25-16. FlexCAN Receive Error Counter (RXECTR)

Table 25-20 describes the RXECTR fields.

Table 25-20. RXECTR Field Descriptions

Bits	Name	Description
7–0	RXECTR	Receive error counter. Indicates the current receive error count as defined in the CAN protocol. See Section 25.4.9, “FlexCAN Error Counters” for more details.

25.5.12 FlexCAN Transmit Error Counter (TXECTR)

	7	0
Field	TXECTR	
Reset	0000_0000	
R/W	R	
Address	IPSBAR + 0x1C_0028	

Figure 25-17. FlexCAN Transmit Error Counter (TXECTR)

Table 25-21 describes the TXECTR fields.

Table 25-21. TXECTR Field Descriptions

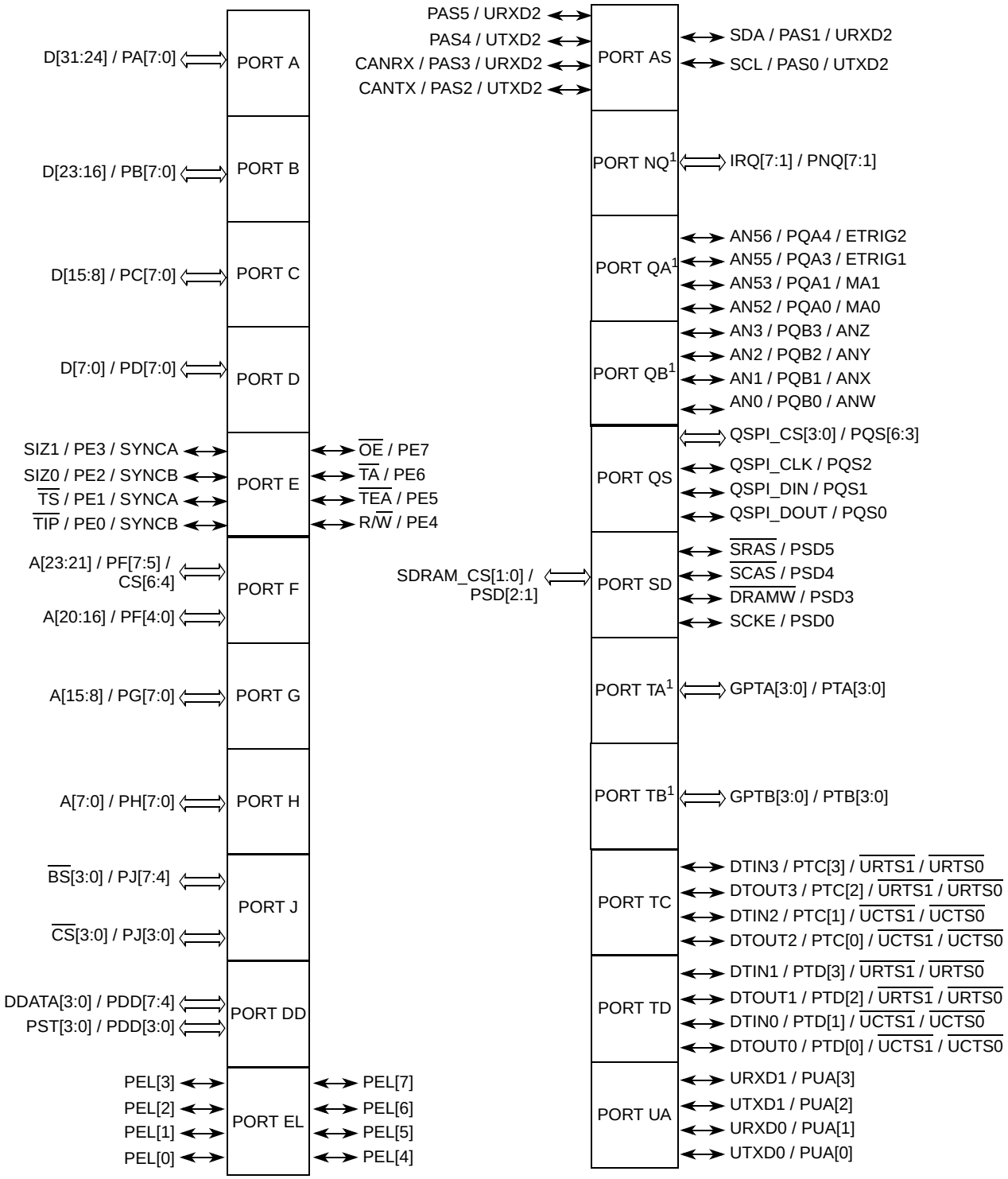
Bits	Name	Description
7–0	TXECTR	Transmit error counter. Indicates the current transmit error count as defined in the CAN protocol. See Section 25.4.9, “FlexCAN Error Counters” for more details.

Chapter 26
General Purpose I/O Module

26.1 Introduction

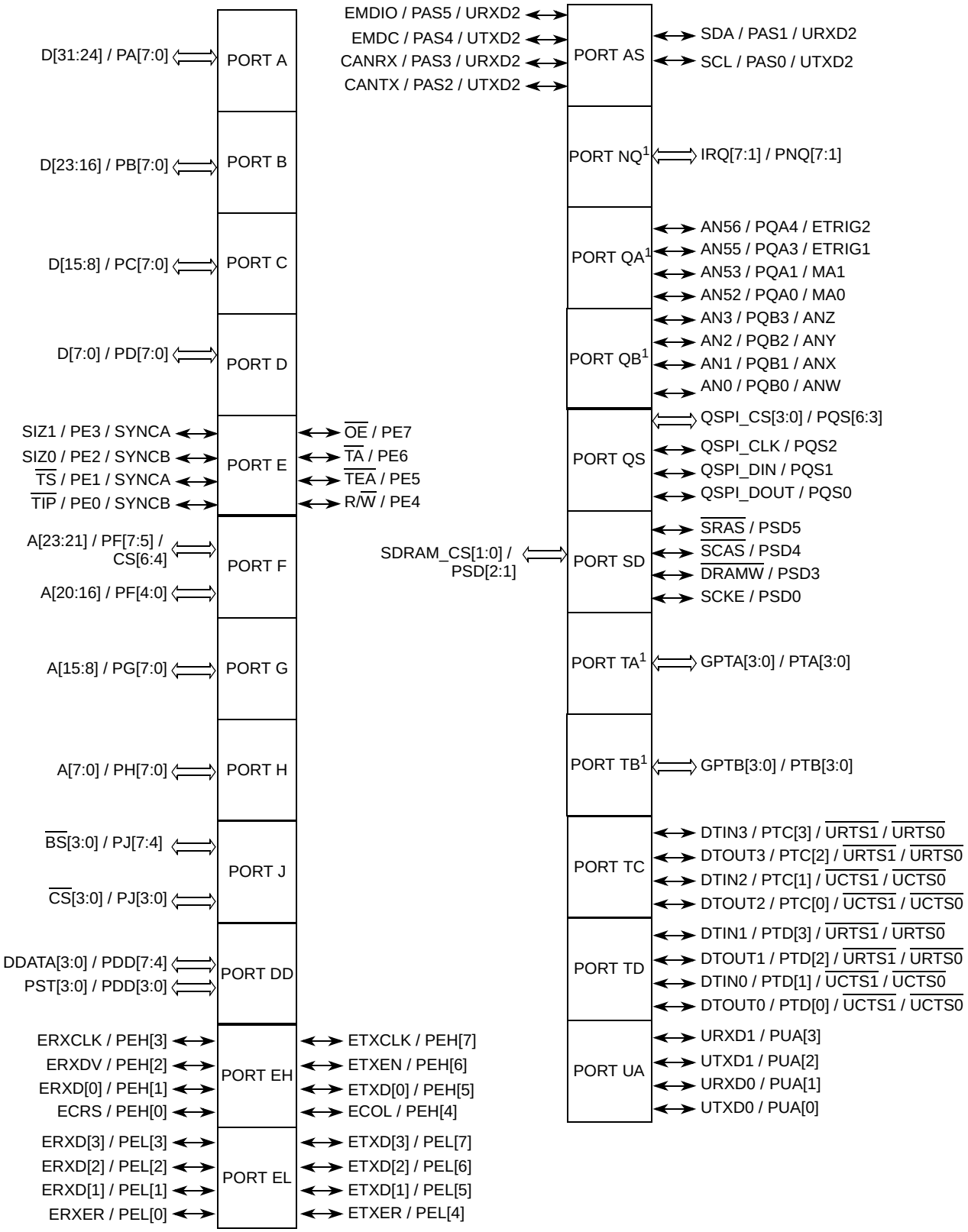
Many of the pins associated with the external interface may be used for several different functions. Their primary function is to provide an external memory interface to access off-chip resources. When not used for their primary function, many of the pins may be used as general-purpose digital I/O pins. 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 26-1 and Figure 26-2 are block diagrams of the ports for the MCF521*x* and MCF528*x*.



1. Although ports NQ, QA, QB, TA, and TB are not part of the ports module, they are included here for comprehensiveness.

Figure 26-1. MCF5214 and MCF5216 Ports Module Block Diagram



1. Although ports NQ, QA, QB, TA, and TB are not part of the ports module, they are included here for comprehensiveness.

Figure 26-2. MCF5280, MCF5281, and MCF5282 Ports Module Block Diagram

26.1.1 Overview

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

- External bus accesses
- Chip selects
- Debug data
- Processor status
- Ethernet data and control (not present on the MCF5214 and MCF5216)
- FlexCAN transmit/receive data
- I^2C serial control
- QSPI
- SDRAM control
- 32-bit DMA timers
- UART transmit/receive

26.1.2 Features

The 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

26.1.3 Modes of Operation

The operational modes for the ports are listed below. For more detailed descriptions of each mode, refer to Section 26.4, “Functional Description.”

- Single-chip mode
  All pins are configured as digital I/O by default, except for debug data pins (DDATA[3:0]) and processor status pins (PST[3:0]).
- Master mode
  Ports A and B function as the upper external data bus. Ports C and D can function as the lower external data bus. Ports E–J are configured to support external memory.

26.2 External Signal Description

The ports control the functionality of several external pins. These pins are listed in Table 26-1.

Table 26-1. Ports External Signals

Primary Function (Pin Name)[1]	GPIO (Default Function)	Alternate Function 1	Alternate Function 2	Description
D[31:0][2]	PA, PB, PC, PD	—	—	External data bus / Ports A, B, C, D
$\overline{OE}$[1]	PE[7]	—	—	Output enable for external reads / Port E[7]
$\overline{TA}$[1]	PE[6]	—	—	Transfer acknowledge for external data transfer / Port E[6]
$\overline{TEA}$[1]	PE[5]	—	—	Transfer error acknowledge for external data transfer / Port E[5]
R/$\overline{W}$[1]	PE[4]	—	—	Read/Write indication for external data transfer / Port E[4]
SIZ1[1]	PE[3]	SYNCA	—	Size of the external data transfer / Port E[3] / Timer A sync
SIZ0[1]	PE[2]	SYNCB	—	Size of the external data transfer / Port E[3:2] / Timer B sync
$\overline{TS}$[1]	PE[1]	SYNCA	—	Transfer start indication for external data transfer / Port E[1] / Timer A sync
$\overline{TIP}$[1]	PE[0]	SYNCB	—	Transfer in progress indication for external data transfer / Port E[0] / Timer B sync
A[23:21][1]	PF[7:5]	$\overline{CS}$[6:4]	—	External address bus [23:21] / Port F[7:5] / Chip selects 6–4
A[20:0][1]	PF[4:0], PG, PH	—	—	External address bus [20:0] / Ports F[4:0], G, H
$\overline{BS}$[3:0][1]	PJ[7:4]	—	—	Byte strobes for external data transfer / Port J[7:4] / SDRAM column address strobes
$\overline{CS}$[3:0][1]	PJ[3:0]	—	—	Chip selects 3 - 0 / Port J[3:0]
DDATA[3:0]	PDD[7:4]	—	—	Debug data / Port DD[7:4]
PST[3:0]	PDD[7:4]	—	—	Processor status / Port DD[3:0]
CANRX	PAS[3]	URXD2	—	FlexCAN receive data / Port AS[3] / URXD2
CANTX	PAS[2]	UTXD2	—	FlexCAN transmit data / Port AS[2] / UTXD2
SDA	PAS[1]	URXD2	—	I^2C serial data / Port AS[1] / URXD2
SCL	PAS[0]	UTXD2	—	I^2C serial clock / Port AS[0] / UTXD2
IRQ[7:1][3]	PNQ[7:1]	—	—	Edge Port external interrupt pins / Port NQ[7:1]
AN[56:55:53:52][2]	PQA [4:3:1:0]	ETRIG[2:1], MA[1:0]	—	QADC analog inputs / Port QA[4:3:1:0] / external triggers / external multiplex control
AN[3:0] [2]	PQB[3:0]	ANZ, ANY, ANX, ANW	—	QADC analog inputs / Port QB[3:0] / multiplexed analog inputs
QSPI_CS [3:0]	PQS[6:3]	—	—	QSPI synchronous peripheral chip selects / Port QS[3:6]
QSPI_CLK	PQS[2]	—	—	QSPI serial clock / Port QS[2]
QSPI_DIN	PQS[1]	—	—	QSPI serial data input / Port QS[1]
QSPI_DOUT	PQS[0]	—	—	QSPI serial data output / Port QS[0]
$\overline{SRAS}$	PSD[5]	—	—	SDRAM synchronous row address strobe / Port SD[5]

Table 26-1. Ports External Signals (continued)

Primary Function (Pin Name)[1]	GPIO (Default Function)	Alternate Function 1	Alternate Function 2	Description
$\overline{\text{SCAS}}$	PSD[4]		—	SDRAM synchronous column address strobe / Port SD[4]
$\overline{\text{DRAMW}}$	PSD[3]	—	—	SDRAM write enable / Port SD[3]
$\overline{\text{SDRAM\_CS}}$[1:0]	PSD[2:1]	—	—	SDRAM row address strobes 1, 0 / Port SD[2:1]
SCKE	PSD[0]	—	—	SDRAM clock enable / Port SD[0]
GPTA[3:0] [2]	PTA[3:0]	—	—	General purpose timer A input/output / Port TA[3:0]
GPTB[3:0] [2]	PTB[3:0]	—	—	General purpose timer B input/output / Port TB[3:0]
DTIN3	PTC[3]	$\overline{\text{URTS1}}$	$\overline{\text{URTS0}}$	DMA timer 3 input / Port TC[3] / UART1 request to send / UART0 request to send
DTOUT3	PTC[2]	$\overline{\text{URTS1}}$	$\overline{\text{URTS0}}$	DMA timer 3 output / Port TC[2] / UART1 request to send / UART0 request to send
DTIN2	PTC[1]	$\overline{\text{UCTS1}}$	$\overline{\text{UCTS0}}$	DMA timer 2 input / Port TC[1] / UART1 clear to send / UART0 clear to send
DTOUT2	PTC[0]	$\overline{\text{UCTS1}}$	$\overline{\text{UCTS0}}$	DMA timer 2 output / Port TC[0] / UART1 clear to send / UART0 clear to send
DTIN1	PTD[3]	$\overline{\text{URTS1}}$	$\overline{\text{URTS0}}$	DMA timer 1 input / Port TD[3] / UART1 request to send / UART0 request to send
DTOUT1	PTD[2]	$\overline{\text{URTS1}}$	$\overline{\text{URTS0}}$	DMA timer 1 output / Port TD[2] / UART1 request to send / UART0 request to send
DTIN0	PTD[1]	$\overline{\text{UCTS1}}$	$\overline{\text{UCTS0}}$	DMA timer 0 input / Port TD[1] / UART1 clear to send / UART0 clear to send
DTOUT0	PTD[0]	$\overline{\text{UCTS1}}$	$\overline{\text{UCTS0}}$	DMA timer 0 output / Port TD[0] / UART1 clear to send / UART0 clear to send
URXD1	PUA[3]	—	—	UART1 receive serial data / Port UA[3]
UTXD1	PUA[2]	—	—	UART1 transmit serial data / Port UA[2]
URXD0	PUA[1]	—	—	UART0 receive serial data / Port UA[1]
UTXD0	PUA[0]	—	—	UART0 transmit serial data / Port UA[0]
The following signals apply to the MCF5280, MCF5281, and MCF5282				
ETXCLK	PEH[7]	—	—	Ethernet transmit clock / PEH[7]
ETXEN	PEH[6]	—	—	Ethernet transmit enable / PEH[6]
ETXD[0]	PEH[5]	—	—	Ethernet transmit data [0] / PEH[5]
ECOL	PEH[4]	—	—	Ethernet collision / PEH[4]
ERXCLK	PEH[3]	—	—	Ethernet receive clock / PEH[3]
ERXDV	PEH[2]	—	—	Ethernet receive data valid / PEH[2]
ERXD[0]	PEH[1]	—	—	Ethernet receive data [0] / PEH[1]

Table 26-1. Ports External Signals (continued)

Primary Function (Pin Name)[1]	GPIO (Default Function)	Alternate Function 1	Alternate Function 2	Description
ECRS	PEH[0]	—	—	Ethernet carrier receive sense / PEH[0]
ETXD[3:1]	PEL[7:5]	—	—	Ethernet transmit data / Port EL[7:5]
ETXER	PEL[4]	—	—	Ethernet transmit error / Port EL[4]
ERXD[3:1]	PEL[3:1]	—	—	Ethernet receive data [3:1] / Port EL[3:1]
ERXER	PEL[0]	—	—	Ethernet receive error / Port EL[0]
EMDIO	PAS[5]	URXD2	—	Ethernet management data control / Port AS[5] / URXD2
EMDC	PAS[4]	UTXD2	—	Ethernet management data clock / Port AS[4] / UTXD2
The following signals apply to MCF5214 and MCF5216 only				
PEL[0]	—	—	—	Port EL[0]
PAS[4]	UTXD[2]	—	—	Port AS[4] / UTXD2
PEL[7:5]	—	—	—	Port EL[7:5]
PEL[4]	—	—	—	Port EL[4]
PEL[3:1]	—	—	—	Port EL[3:1]
PAS[5]	URXD2	—	—	Port AS[5] / URXD2

1 The primary functionality of a pin is not necessarily the default function of the pin after reset. Pins that have muxed GPIO functionality will default to GPIO inputs.

2 Function available in master mode only

3 Pins not actually part of Port Module, but included here for complete listing of available I/O ports. See separate section for description of this port.

Refer to Chapter 14, “Signal Descriptions” for more detailed descriptions of these pins. The function of the pins (primary function, GPIO, etc.) is determined by the various ports registers and the mode of operation. Refer to Table 14-1 for detailed descriptions of pin functions.

26.3 Memory Map/Register Definition

26.3.1 Register Overview

Table 26-2 summarizes all the registers in the ports address space.

Table 26-2. Ports Module Memory Map

IPSBAR + Offset	31–24	23–16	15–8	7–0	Access[1]
Port Output Data Registers					
0x10_0000	PORTA	PORTB	PORTC	PORTD	S/U
0x10_0004	PORTE	PORTF	PORTG	PORTH	S/U

Table 26-2. Ports Module Memory Map (continued)

IPSBAR + Offset	31–24	23–16	15–8	7–0	Access[1]
0x10_0008	PORTJ	PORTDD	PORTEH (Reserved on MCF521x)	PORTEL	S/U
0x10_000C	PORTAS	PORTQS	PORTSD	PORTTC	S/U
0x10_0010	PORTTD	PORTUA	Reserved[2]		S/U

Table 26-2. Ports Module Memory Map (continued)

IPSBAR + Offset	31–24	23–16	15–8	7–0	Access[1]
Port Data Direction Registers					
0x10_0014	DDRA	DDRB	DDRC	DDRD	S/U
0x10_0018	DDRE	DDRF	DDRG	DDRH	S/U
0x10_001C	DDRJ	DDRDD	DDREH (Reserved on MCF521x)	DDREL	S/U
0x10_0020	DDRAS	DDRQS	DDRSD	DDRTC	S/U
0x10_0024	DDRTD	DDRUA	Reserved[2]		S/U
Port Pin Data/Set Data Registers					
0x10_0028	PORTAP/SETA	PORTBP/SETB	PORTCP/SETC	PORTDP/SETD	S/U
0x10_002C	PORTEP/SETE	PORTFP/SETF	PORTGP/SETG	PORTHP/SETH	S/U
0x10_0030	PORTJP/SETJ	PORTDDP/SETDD	PORTEHP/SETEH (Reserved on MCF521x)	PORTELP/SETEL	S/U
0x10_0034	PORTASP/SETAS	PORTQSP/SETQS	PORTSDP/SETSD	PORTTCP/SETTC	S/U
0x10_0038	PORTTDP/SETTD	PORTUAP/SETUA	Reserved[2]		S/U
Port Clear Output Data Registers					
0x10_003C	CLRA	CLRB	CLRC	CLRD	S/U
0x10_0040	CLRE	CLRF	CLRG	CLRH	S/U
0x10_0044	CLRJ	CLRDD	CLREH (Reserved on MCF521x)	CLREL	S/U
0x10_0048	CLRAS	CLRQS	CLRSD	CLRTC	S/U
0x10_004C	CLRTD	CLRUA	Reserved[2]		S/U
Port Pin Assignment Registers					
0x10_0050	PBCDPAR	PFPAR	PEPAR		S/U
0x10_0054	PJPAR	PSDPAR	PASPAR		S/U
0x10_0058	PEHLPAR	PQSPAR	PTCPAR	PTDPAR	S/U
0x10_005C	PUAPAR	Reserved[2]			S/U
0x10_0060 – 0x10_007C	Reserved[2]				

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

2 Writing to reserved address locations has no effect and reading returns 0s.

26.3.2 Register Descriptions

26.3.2.1 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.

Most PORT*n* registers have a full 8-bit implementation, as shown in Figure 26-3. The remaining PORT*n* registers use fewer than eight bits. Their bit definitions are shown in Figure 26-4, Figure 26-5, and Figure 26-6.

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.

	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							
Address	IPSBAR + 0x10_0000 (PORTA), 0x10_0001 (PORTB), 0x10_0002 (PORTC), 0x10_0003 (PORTD), 0x10_0004 (PORTE), 0x10_0005 (PORTF), 0x10_0006 (PORTG), 0x10_0007 (PORTH), 0x10_0008 (PORTJ), 0x10_0009 (PORTDD), 0x10_000A (PORTEH), 0x10_000B (PORTEL)							

Figure 26-3. Port Output Data Registers (8-bit)

	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	0011_1111							
R/W:	R	R/W						
Address	IPSBAR + 0x10_000D (PORTQS)							

Figure 26-4. Port Output Data Register (7-bit)

	7	6	5	4	3	2	1	0
Field	—		PORT*n*5	PORT*n*4	PORT*n*3	PORT*n*2	PORT*n*1	PORT*n*0
Reset	0011_1111							
R/W:	R		R/W					
Address	IPSBAR + 0x10_000C (PORTAS), 0x10_000E (PORTSD)							

Figure 26-5. Port Output Data Registers (6-bit)

	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			
Address	IPSBAR + 0x10_000F (PORTTC), 0x10_0010 (PORTTD), 0x10_0011 (PORTUA)							

Figure 26-6. Port Output Data Registers (4-bit)

PORT*n* bits are described in Table 26-3.

Table 26-3. PORT*n* (8-bit, 7-bit, 6-bit, and 4-bit) Field Descriptions

Register	Bits	Name	Description
8-bit	7–0	PORT*nx*	Port output data bits. 1 Drives 1 when the port *n* pin is a digital output 0 Drives 0 when the port *n* pin is a digital output
7-bit	6–0		
6-bit	5–0		
4-bit	3–0		
7-bit	7	—	Reserved, should be cleared.
6-bit	7–6		
4-bit	7–4		

26.3.2.2 Port Data Direction Registers (DDR*n*)

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

Most DDRs have a full 8-bit implementation, as shown in Figure 26-7. The remaining DDRs use fewer than eight bits. Their bit definitions are shown in Figure 26-8, Figure 26-9, and Figure 26-10.

The DDRs are read/write. At reset, all bits in the DDRs are cleared.

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.

	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							
Address	IPSBAR + 0x10_0014 (DDRA), 0x10_0015 (DDRB), 0x10_0016 (DDRC), 0x10_0017 (DDRD), 0x10_0018 (DDRE), 0x10_0019 (DDRF), 0x10_001A (DDRG), 0x10_001B (DDRH), 0x10_001C (DDRJ), 0x10_001D (DDRDD), 0x10_001E (DDREH), 0x10_001F (DDREL)							

Figure 26-7. Port Data Direction Registers (8-bit)

	7	6						0
Field	—	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	R/W						
Address	IPSBAR + 0x10_0021(DDRQS)							

Figure 26-8. Port Data Direction Register (7-bit)

	7	6	5	4	3	2	1	0
Field	—		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					
Address	IPSBAR + 0x10_0020 (DDRAS), 0x10_0022 (DDRSD)							

Figure 26-9. Port Data Direction Registers (6-bit)

	7				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			
Address	IPSBAR + 0x10_0023 (DDRTC), 0x10_0024 (DDRTD), 0x10_0025 (DDRUA)								

Figure 26-10. Port Data Direction Registers (4-bit)

Table 26-4. DDR*n* (8-bit, 6-bit, and 4-bit) Field Descriptions

Register	Bits	Name	Description
8-bit	7–0	DDR*nx*	Port *n* data direction bits. 1 Port *n* pin configured as an output 0 Port *n* pin configured as an input
7-bit	6–0		
6-bit	5–0		
4-bit	3–0		
7-bit	7	—	Reserved, should be cleared.
6-bit	7–6		
4-bit	7–4		

26.3.2.3 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.

Most PORT*n* registers have a full 8-bit implementation, as shown in Figure 26-11. The remaining PORT*n* registers use fewer than eight bits. Their bit definitions are shown in Figure 26-12, Figure 26-13, and Figure 26-14.

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.

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

	7	6	5	4	3	2	1	0
Field	PORT*n*P7/ SET*n*7	PORT*n*P6/ SET*n*6	PORT*n*P5/ SET*n*5	PORT*n*P4/ SET*n*4	PORT*n*P3/ SET*n*3	PORT*n*P2/ SET*n*2	PORT*n*P1/ SET*n*1	PORT*n*P0/ SET*n*0
Reset	Current Pin State							
R/W:	R/W							
Address	IPSBAR + 0x10_0028 (PORTAP/SETA), 0x10_0029 (PORTBP/SETB), 0x10_002A (PORTCP/SETC), 0x10_002B (PORTDP/SETD), 0x10_002C (PORTEP/SETE), 0x10_002D (PORTFP/SETF), 0x10_002E (PORTGP/SETG), 0x10_002F (PORTHP/SETH), 0x10_0030 (PORTJP/SETJ), 0x10_0031 (PORTDDP/SETDD), 0x10_0032 (PORTEHP/SETEH), 0x10_0033 (PORTELP/SETEL)							

Figure 26-11. Port Pin Data/Set Data Registers (8-bit)

	7	6	5	4	3	2	1	0
Field	—	PORT*n*P6/S ET*n*6	PORT*n*P5/S ET*n*5	PORT*n*P4/S ET*n*4	PORT*n*P3/S ET*n*3	PORT*n*P2/S ET*n*2	PORT*n*P1/S ET*n*1	PORT*n*P0/S ET*n*0
Reset	0	Current Pin State						
R/W:	—	R/W						
Address	IPSBAR + 0x10_0035 (PORTQSP/SETQS)							

Figure 26-12. Port Pin Data/Set Data Register (7-bit)

	7	6	5	4	3	2	1	0
Field	—		PORT*n*P5/ SET*n*5	PORT*n*P4/ SET*n*4	PORT*n*P3/ SET*n*3	PORT*n*P2/ SET*n*2	PORT*n*P1/ SET*n*1	PORT*n*P0/ SET*n*0
Reset	00		Current Pin State					
R/W:	—		R/W					
Address	IPSBAR + 0x10_0034 (PORTASP/SETAS), 0x10_0036 (PORTSDP/SETSD)							

Figure 26-13. Port Pin Data/Set Data Registers (6-bit)

	7	6	5	4	3	2	1	0
Field	—				PORT*n*P3/ SET*n*3	PORT*n*P2/ SET*n*2	PORT*n*P1/ SET*n*1	PORT*n*P0/ SET*n*0
Reset	0000				Current Pin State			
R/W:	—				R/W			
Address	IPSBAR + 0x10_0037 (PORTTCP/SETTC), 0x10_0038 (PORTTDP/SETTD), 0x10_0039 (PORTUAP/SETUA)							

Figure 26-14. Port Pin Data/Set Data Registers (4-bit)

PORT*n*P/SET*n* bits are described in Table 26-5.

Table 26-5. PORT*n*P/SET*n* (8-bit, 6-bit, and 4-bit) Field Descriptions

Register	Bits	Name	Description
8-bit	7–0	PORTx*n*P/SETx*n*	Port x Pin Data/Set Data Bits 1 Port x pin state is 1 (read); set corresponding PORTx bit (write) 0 Port x pin state is 0 (read)
7-bit	6–0		
6-bit	5–0		
4-bit	3–0		
7-bit	7	—	Reserved, should be cleared.
6-bit	7–6		
4-bit	7–4		

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

Clearing a CLR*n* register clears the corresponding bits in the PORT*n* register. Setting it has no effect. Reading the CLR*n* register returns 0s.

Most PORT*n* registers have a full 8-bit implementation, as shown in Figure 26-15. The remaining PORT*n* registers use fewer than eight bits. Their bit definitions are shown in Figure 26-16, Figure 26-17, and Figure 26-18.

The CLR*n* registers are read/write accessible.

	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							
Address	IPSBAR + 0x10_003C (CLRA), 0x10_003D (CLRB), 003E (CLRC), 0x10_003F (CLRD), 0x10_0040 (CLRE), 0x10_0041 (CLRF), 0x10_0042 (CLRG), 0x10_0043 (CLRH), 0x10_0044 (CLRJ), 0x10_0045 (CLRDD), 0x10_0046 (CLREH), 0x10_0047 (CLREL)							

Figure 26-15. Port Clear Output Data Registers (8-bit)

	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						
Address	IPSBAR + 0x10_0049 (CLRQS)							

Figure 26-16. Port Clear Output Data Register (7-bit)

	7	6	5	4	3	2	1	0
Field	—		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					
Address	IPSBAR + 0x10_0048 (CLRAS), 0x10_004A (CLRSD)							

Figure 26-17. Port Clear Output Data Registers (6-bit)

	7	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			
Address	IPSBAR + 0x10_004B (CLRTC), 0x10_004C (CLRTD), 0x10_004D (CLRUA)					

Figure 26-18. Port Clear Output Data Registers (4-bit)

CLR*n* register bits are described in Table 26-6.

Table 26-6. CLR*n* (8-bit,7-bit, 6-bit, and 4-bit) Field Descriptions

Register	Bits	Name	Description
8-bit	7–0	CLR*nx*	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
7-bit	6–0		
6-bit	5–0		
4-bit	3–0		
7-bit	7	—	Reserved, should be cleared.
6-bit	7–6		
4-bit	7–4		

26.3.2.5 Port B/C/D Pin Assignment Register (PBCDPAR)

The PBCDPAR controls the pin function of ports B, C, and D.

The PBCDPAR register is read/write.

	7	6	5 … 0
Field	PBPA	PCDPA	—
Reset	See Note 1	See Note 1	00_0000
R/W:	R/W	R/W	R
Address	IPSBAR + 0x10_0050		

Figure 26-19. Port B/C/D Pin Assignment Register (PBCDPAR)

1 Reset state determined during reset configuration as shown in Table 26-8.

Table 26-7. PBCDPAR Field Descriptions

Bits	Name	Description
7	PBPA	Port B pin assignment. Configures the port B pins for their primary function (D[23:16]) or digital I/O. 1 Port B pins configured for primary function (D[23:16]) 0 Port B pins configured for digital I/O
6	PCDPA	Ports C,D pin assignment. Configures the port C and D pins for their primary functions (D[15:8], D[7:0]) or digital I/O. 1 Port C,D pins configured for primary function (D[15:8], D[7:0]) 0 Port C,D pins configured for digital I/O
5–0	—	Reserved, should be cleared.

Table 26-8. Reset Values for PBCDPAR Bits

Mode of Operation	Port Size of External Boot Device[1]	PBPA Reset Value	PCDPA Reset Value
Master mode	8-bit	0	0
	16-bit	1	0
	32-bit	1	1
Single chip mode	N/A	0	0

1 Note if the port size of the external boot device is less than the port size of the external SDRAM, the PBCDPAR register must be written after reset to enable the primary function(s) on ports B,C, and D, before any SDRAM accesses are attempted.

26.3.2.6 Port E Pin Assignment Register (PEPAR)

The PEPAR controls the pin function of port E.

The PEPAR register is read/write.

	15	14	13	12	11	10	9	8
Field	—	PEPA7	—	PEPA6	—	PEPA5	—	PEPA4
Reset	0	See Note 1	0	See Note 1	0	See Note 1	0	See Note 1
R/W:	R	R/W	R	R/W	R	R/W	R	R/W

	7	6	5	4	3	2	1	0
Field	—	PEPA3	—	PEPA2	PEPA1		PEPA0	
Reset	0	See Note 1	0	See Note 1	See Note 1		See Note 1	
R/W:	R	R/W	R	R/W	R/W		R/W	
Address	IPSBAR + 0x10_0052							

Figure 26-20. Port E Pin Assignment Register (PEPAR)

[1] Reset state determined during reset configuration as shown in Table 26-10.

Table 26-9. PEPAR Field Descriptions

Bits	Name	Description
14	PEPA7	Port E pin assignment 7. This bit configures the port E7 pin for its primary function ($\overline{OE}$) or digital I/O. 1 Port E7 pin configured for primary function ($\overline{OE}$) 0 Port E7 pin configured for digital I/O
12	PEPA6	Port E pin assignment 6. This bit configures the port E6 pin for its primary function ($\overline{TA}$) or digital I/O. 1 Port E6 pin configured for primary function ($\overline{TA}$) 0 Port E6 pin configured for digital I/O
10	PEPA5	Port E pin assignment 5. This bit configures the port E5 pin for its primary function ($\overline{TEA}$) or digital I/O. 1 Port E5 pin configured for primary function ($\overline{TEA}$) 0 Port E5 pin configured for digital I/O
8	PEPA4	Port E pin assignment 4. This bit configures the port E4 pin for its primary function (R/$\overline{W}$) or digital I/O. 1 Port E4 pin configured for primary function (R/$\overline{W}$) 0 Port E4 pin configured for digital I/O

Table 26-9. PEPAR Field Descriptions (continued)

Bits	Name	Description
6	PEPA3	Port E pin assignment 3 This bit configures the port E3 pin for its alternate function (SYNCA) or digital I/O. 1 Port E3 pin configured for alternate function (SYNCA) 0 Port E3 pin configured for digital I/O NOTE: The SIZ1 primary function on the port E3 pin is enabled by the SZEN bit in the CCR register. Please refer to the Chapter 27, "Chip Configuration Module (CCM) for more information on chip configuration and the SZEN bit.
4	PEPA2	Port E pin assignment 2 This bit configures the port E2 pin for its alternate function (SYNCB) or digital I/O. 1 Port E2 pin configured for alternate function (SYNCB) 0 Port E2 pin configured for digital I/O NOTE: The SIZ0 primary function on the port E2 pin is enabled by the SZEN bit in the CCR register. Please refer to the Chapter 27, "Chip Configuration Module (CCM)," for more information on chip configuration and the SZEN bit.
3–2	PEPA1	Port E pin assignment 1. This two-bit field in PEPAR[3:2] configures the port E1 pin for its primary function ($\overline{TS}$), alternate function (SYNCA), or digital I/O. 0x Port E1 pin configured for digital I/O 10 Port E1 pin configured for alternate function (SYNCA) 11 Port E1 pin configured for primary function ($\overline{TS}$)
1-0	PEPA0	Port E pin assignment 0. This two-bit field in PEPAR[1:0] configures the port E0 pin for its primary function ($\overline{TIP}$), alternate function (SYNCB), or digital I/O. 0x Port E0 pin configured for digital I/O 10 Port E0 pin configured for alternate function (SYNCB) 11 Port E0 pin configured for primary function ($\overline{TIP}$)

The reset values for all bits and fields in PEPAR are shown in Table 26-10.

Table 26-10. Reset Values for PEPAR Bits and Fields

Mode of Operation	Reset Values for PEPA*n* Bits (*n* = 2,3,4,5,6,7)	Reset Values for PEPA*n* Fields (*n* = 0,1)
Master mode	1	11
Single chip mode	0	00

26.3.2.7 Port F Pin Assignment Register (PFPAR)

The PFPAR controls the pin function of port F[7:5].

	7	6	5	4 … 0
Field	PFPA7	PFPA6	PFPA5	—
Reset	See Note 1			0_0000
R/W:	R/W			R
Address	IPSBAR + 0x10_0051			

Figure 26-21. Port F Pin Assignment Register (PFPAR)

[1] Reset state determined during reset configuration. PFPA*n* = 1 in master mode and 0 in all other modes.

Table 26-11. PFPAR Field Descriptions

Bits	Name	Description
7	PFPA7	Port F pin assignment 1. The PFPA7 bit configures the port F7 pin for its primary function (A23), alternate function ($\overline{CS6}$), or digital I/O. 1 Port F7 pin configured for primary function (A23) or alternate function ($\overline{CS6}$), depending on the chip configuration. 0 Port F7 pin configured for digital I/O Refer to Chapter 27, “Chip Configuration Module (CCM)” for more information on reset configuration.
6	PFPA6	Port F pin assignment 6. The PFPA6 bit configures the port F6 pin for its primary function (A22), alternate function ($\overline{CS5}$), or digital I/O. 1 Port F6 pin configured for primary function (A22) or alternate function ($\overline{CS5}$), depending on the chip configuration. 0 Port F6 pin configured for digital I/O Refer to Chapter 27, “Chip Configuration Module (CCM)” for more information on reset configuration.
5	PFPA5	Port F pin assignment 5. The PFPAR bit configures the port F5 pin for its primary function (A21), alternate function ($\overline{CS4}$), or digital I/O. 1 Port F5 pin configured for primary function (A21) or alternate function ($\overline{CS4}$), depending on the chip configuration. 0 Port F5 pin configured for digital I/O Refer to Chapter 27, “Chip Configuration Module (CCM)” for more information on reset configuration.
4–0	—	Reserved, should be cleared.

26.3.2.8 Port J Pin Assignment Register (PJPAR)

The PJPAR controls the pin function of port J.

	7	6	5	4	3	2	1	0
Field	PJPA7	PJPA6	PJPA5	PJPA4	PJPA3	PJPA2	PJPA1	PJPA0
Reset	See Note 1							
R/W:	R/W							
Address	IPSBAR + 0x10_0054							

Figure 26-22. Port J Pin Assignment Register (PJPAR)

[1] Reset state determined during reset configuration. PJPA*n* = 1 in master mode and 0 in all other modes.

Table 26-12. PJPAR Field Descriptions

Bits	Name	Description
7	PJPA7	Port J pin assignment 7. This bit configures the port J7 pin for its primary function ($\overline{BS3}$) or digital I/O. 1 Port J7 pin configured for its primary function ($\overline{BS3}$) 0 Port J7 pin configured for digital I/O
6	PJPA6	Port J pin assignment 6. This bit configures the port J6 pin for its primary function ($\overline{BS2}$) or digital I/O. 1 Port J6 pin configured for its primary function ($\overline{BS2}$) 0 Port J6 pin configured for digital I/O
5	PJPA5	Port J pin assignment 5. This bit configures the port J5 pin for its primary function ($\overline{BS1}$) or digital I/O. 1 Port J5 pin configured for its primary function ($\overline{BS1}$) 0 Port J5 pin configured for digital I/O
4	PJPA4	Port J pin assignment 4. This bit configures the port J4 pin for its primary function ($\overline{BS0}$) or digital I/O. 1 Port J4 pin configured for its primary function ($\overline{BS0}$) 0 Port J4 pin configured for digital I/O
3	PJPA3	Port J pin assignment 3. This bit configures the port J3 pin for its primary function ($\overline{CS3}$) or digital I/O. 1 Port J3 pin configured for its primary function ($\overline{CS3}$) 0 Port J3 pin configured for digital I/O
2	PJPA2	Port J pin assignment 2. This bit configures the port J2 pin for its primary function ($\overline{CS2}$) or digital I/O. 1 Port J2 pin configured for its primary function ($\overline{CS2}$) 0 Port J2 pin configured for digital I/O
1	PJPA1	Port J pin assignment 1. This bit configures the port J1 pin for its primary function ($\overline{CS1}$) or digital I/O. 1 Port J1 pin configured for its primary function ($\overline{CS1}$) 0 Port J1 pin configured for digital I/O
0	PJPA0	Port J pin assignment 0. This bit configures the port J0 pin for its primary function ($\overline{CS0}$) or digital I/O. 1 Port J0 pin configured for its primary function ($\overline{CS0}$) 0 Port J0 pin configured for digital I/O

26.3.2.9 Port SD Pin Assignment Register (PSDPAR)

The PSDPAR controls the pin function of port SD.

	7	6–0
Field	PSDPA	—
Reset	See Note 1	000_0000
R/W:	R/W	R
Address	IPSBAR + 0x10_0055	

Figure 26-23. Port SD Pin Assignment Register (PSDPAR)

1 Reset state determined during reset configuration. PJPA*n* = 1 in master mode and 0 in all other modes.

Table 26-13. PSDPAR Field Descriptions

Bits	Name	Description
7	PSDPA	Port SD pin assignment. This bit configures the port SD[5:0] pins for their primary functions ($\overline{SRAS}$, $\overline{SCAS}$, $\overline{DRAMW}$, $\overline{SDRAM\_CS}$[1:0], SCKE) or digital I/O. 1 Port SD[5:0] pins configured for primary functions ($\overline{SRAS}$, $\overline{SCAS}$, $\overline{DRAMW}$, $\overline{SDRAM\_CS}$[1:0], SCKE) 0 Port SD[5:0] pin configured for digital I/O
6–0	—	Reserved, should be cleared.

26.3.2.10 Port AS Pin Assignment Register (PASPAR)

The PASPAR controls the pin function of port AS.

	15–12	11	10	9	8
Field	—	PASPA5		PASPA4	
Reset	0000_0000				
R/W:	R	R/W			

	7	6	5	4	3	2	1	0
Field	PASPA3		PASPA2		PASPA1		PASPA0	
Reset	0000_0000							
R/W:	R/W							
Address	IPSBAR + 0x10_0056							

Figure 26-24. Port AS Pin Assignment Register (PASPAR)

Table 26-14. PASPAR Field Descriptions

Bits	Name	Description
15–12	—	Reserved, should be cleared.
11–10	PASPA5	Port AS pin assignment 5. These bits configure the AS5 pin for its primary function (EMDIO), alternate function (URXD2), or digital I/O. 0x Port AS5 pin configured for digital I/0 10 Port AS5 pin configured for alternate function (URXD2) 11 Port AS5 pin configured for primary function (EMDIO) **Note:** Setting 11 is reserved for the MCF5214 and MCF5216.
9-8	PASPA4	Port AS pin assignment 4. These bits configure the AS4 pin for its primary function (EMDC), alternate function (UTXD2), or digital I/O. 0x Port AS4 pin configured for digital I/0 10 Port AS4 pin configured for alternate function (UTXD2) 11 Port AS4 pin configured for primary function (EMDC) **Note:** Setting 11 is reserved for the MCF5214 and MCF5216.
7-6	PASPA3	Port AS pin assignment 3. These bits configure the AS3 pin for its primary function (CANRX), alternate function (URXD2), or digital I/O. 0x Port AS3 pin configured for digital I/0 10 Port AS3 pin configured for alternate function (URXD2) 11 Port AS3 pin configured for primary function (CANRX)
5-4	PASPA2	Port AS pin assignment 2. These bits configure the AS2 pin for its primary function (CANTX), alternate function (UTXD2), or digital I/O. 0x Port AS2 pin configured for digital I/0 10 Port AS2 pin configured for alternate function (UTXD2) 11 Port AS2 pin configured for primary function (CANTX)
3-2	PASPA1	Port AS pin assignment 1. These bits configure the AS1 pin for its primary function (SDA), alternate function (URXD2), or digital I/O. 0x Port AS1 pin configured for digital I/0 10 Port AS1 pin configured for alternate function (URXD2) 11 Port AS1 pin configured for primary function (SDA)
1-0	PASPA0	Port AS pin assignment 0. These bits configure the AS0 pin for its primary function (SCL), alternate function (UTXD2), or digital I/O. 0x Port AS0 pin configured for digital I/0 10 Port AS0 pin configured for alternate function (UTXD2) 11 Port AS0 pin configured for primary function (SCL)

26.3.2.11 Port EH/EL Pin Assignment Register (PEHLPAR)

The PEHLPAR controls the pin function of ports EH and EL.

NOTE

Port EH is not available on the MCF5214 and MCF5216.

The PEHLPAR is read/write accessible.

	7	6	5	0
Field	PEHPA	PELPA	—	
Reset	0000_0000			
R/W:	R/W		R	
Address	IPSBAR + 0x10_0058			

Figure 26-25. Port EH/EL Pin Assignment Register (PEHLPAR)

Table 26-15. PEHLPAR Field Descriptions

Bits	Name	Description
7	PEHPA	Port EH pin assignment. This bit configures the port EH pins for its primary functions (ETXCLK, ETXEN, ETXD[0], ECOL, ERXCLK, ERXDV, ERXD[0], ECRS) or digital I/O. 0 Port EH pins configured for digital I/O 1 Port EH pins configured for primary functions (ETXCLK, ETXEN, ETXD[0], ECOL, ERXCLK, ERXDV, ERXD[0], ECRS) **Note:** This bit is reserved for the MCF5214 and MCF5216.
6	PELPA	Port EL pin assignment. This bit configures the port EL pins for their primary functions (ETXD[3], ETXD[2], ETXD[1], ETXER, ERXD[3], ERXD[2], ERXD[1], ERXER) or digital I/O. 0 Port EL pins configured for digital I/O 1 Port EL pins configured for primary functions (ETXD[3], ETXD[2], ETXD[1], ETXER, ERXD[3], ERXD[2], ERXD[1], ERXER) **Note:** Setting 1 is reserved for the MCF5214 and MCF5216.
5–0	—	Reserved, should be cleared.

26.3.2.12 Port QS Pin Assignment Register (PQSPAR)

The PQSPAR controls the pin function of port QS.

	7	6	5	4	3	2	1	0
Field	—	PQSPA6	PQSPA5	PQSPA4	PQSPA3	PQSPA2	PQSPA1	PQSPA0
Reset	0000_0000							
R/W:	R	R/W						
Address	IPSBAR + 0x10_0059							

Figure 26-26. Port QS Pin Assignment Register (PQSPAR)

Table 26-16. PQSPAR Field Description

Bits	Name	Description
7	—	Reserved, should be cleared.
6	PQSPA6	Port QS pin assignment 6. This bit configures the port QS6 pin for its primary function (QSPI_CS3) or digital I/O. 1 Port QS6 pin configured for primary function (QSPI_CS3) 0 Port QS6 pin configured for digital I/O

Table 26-16. PQSPAR Field Description (continued)

Bits	Name	Description
5	PQSPA5	Port QS pin assignment 5. This bit configures the port QS5 pin for its primary function (QSPI_CS2) or digital I/O. 1 Port QS5 pin configured for primary function (QSPI_CS2) 0 Port QS5 pin configured for digital I/O
4	PQSPA4	Port QS pin assignment 4. This bit configures the port QS4 pin for its primary function (QSPI_CS1) or digital I/O. 1 Port QS4 pin configured for primary function (QSPI_CS1) 0 Port QS4 pin configured for digital I/O
3	PQSPA3	Port QS pin assignment 3. This bit configures the port QS3 pin for its primary function (QSPI_CS0) or digital I/O. 1 Port QS3 pin configured for primary function (QSPI_CS0) 0 Port QS3 pin configured for digital I/O
2	PQSPA2	Port QS pin assignment 2. This bit configures the port QS2 pin for its primary function (QSPI_CLK) or digital I/O. 1 Port QS2 pin configured for primary function (QSPI_CLK) 0 Port QS2 pin configured for digital I/O
1	PQSPA1	Port QS pin assignment 1. This bit configures the port QS1 pin for its primary function (QSPI_DIN) or digital I/O. 1 Port QS1 pin configured for primary function (QSPI_DIN) 0 Port QS1 pin configured for digital I/O
0	PQSPA0	Port QS pin assignment 0. This bit configures the port QS0 pin for its primary function (QSPI_DOUT) or digital I/O. 1 Port QS0 pin configured for primary function (QSPI_DOUT) 0 Port QS0 pin configured for digital I/O

26.3.2.13 Port TC Pin Assignment Register (PTCPAR)

The PTCPAR controls the pin function of port TC.

	7	6	5	4	3	2	1	0
Field	PTCPA3		PTCPA2		PTCPA1		PTCPA0	
Reset	0000_0000							
R/W:	R/W							
Address	IPSBAR + 0x10_005A							

Figure 26-27. Port TC Pin Assignment Register (PTCPAR)

Table 26-17. PTCPAR Field Descriptions

Bits	Name	Description
7–6	PTCPA3	Port TC pin assignment 3. This field configures the port TC3 pin for its primary function (DTIN3), alternate 1 function ($\overline{\text{URTS1}}$), alternate 2 function ($\overline{\text{URTS0}}$) or digital I/O. 00 Port TC3 pin configured for digital I/O 01 Port TC3 pin configured for alternate 2 function ($\overline{\text{URTS0}}$) 10 Port TC3 pin configured for alternate 1 function ($\overline{\text{URTS1}}$) 11 Port TC3 pin configured for primary function (DTIN3)
5-4	PTCPA2	Port TC pin assignment 2. This field configures the port TC2 pin for its primary function (DTOUT3), alternate 1 function ($\overline{\text{URTS1}}$), alternate 2 function ($\overline{\text{URTS0}}$) or digital I/O. 00 Port TC2 pin configured for digital I/O 01 Port TC2 pin configured for alternate 2 function ($\overline{\text{URTS0}}$) 10 Port TC2 pin configured for alternate 1 function ($\overline{\text{URTS1}}$) 11 Port TC2 pin configured for primary function (DTOUT3)
3-2	PTCPA1	Port TC pin assignment 1. This field configures the port TC1 pin for its primary function (DTIN2), alternate 1 function ($\overline{\text{UCTS1}}$), alternate 2 function ($\overline{\text{UCTS0}}$) or digital I/O. 00 Port TC1 pin configured for digital I/O 01 Port TC1 pin configured for alternate 2 function ($\overline{\text{UCTS0}}$) 10 Port TC1 pin configured for alternate 1 function ($\overline{\text{UCTS1}}$) 11 Port TC1 pin configured for primary function (DTIN2)
1-0	PTCPA0	Port TC pin assignment 0. This field configures the port TC0 pin for its primary function (DTOUT2), alternate 1 function ($\overline{\text{UCTS1}}$), alternate 2 function ($\overline{\text{UCTS0}}$) or digital I/O. 00 Port TC0 pin configured for digital I/O 01 Port TC0 pin configured for alternate 2 function ($\overline{\text{UCTS0}}$) 10 Port TC0 pin configured for alternate 1 function ($\overline{\text{UCTS1}}$) 11 Port TC0 pin configured for primary function (DTOUT2)

26.3.2.14 Port TD Pin Assignment Register (PTDPAR)

The PTDPAR controls the pin function of port TD.

	7	6	5	4	3	2	1	0
Field	PTDPA3		PTDPA2		PTDPA1		PTDPA0	
Reset	0000_0000							
R/W:	R/W							
Address	IPSBAR + 0x10_005B							

Figure 26-28. Port TD Pin Assignment Register (PTDPAR)

Table 26-18. PTDPAR Field Descriptions

Bits	Name	Description
7–6	PTDPA3	Port TD pin assignment 3. This field configures the port TD3 pin for its primary function (DTIN1), alternate 1 function ($\overline{\text{URTS1}}$), alternate 2 function ($\overline{\text{URTS0}}$) or digital I/O. 00 Port TD3 pin configured for digital I/O 01 Port TD3 pin configured for alternate 2 function ($\overline{\text{URTS0}}$) 10 Port TD3 pin configured for alternate 1 function ($\overline{\text{URTS1}}$) 11 Port TD3 pin configured for primary function (DTIN1)
5-4	PTDPA2	Port TD pin assignment 2. This field configures the port TD2 pin for its primary function (DTOUT1), alternate 1 function ($\overline{\text{URTS1}}$), alternate 2 function ($\overline{\text{URTS0}}$) or digital I/O. 00 Port TD2 pin configured for digital I/O 01 Port TD2 pin configured for alternate 2 function ($\overline{\text{URTS0}}$) 10 Port TD2 pin configured for alternate 1 function ($\overline{\text{URTS1}}$) 11 Port TD2 pin configured for primary function (DTOUT1)
3-2	PTDPA1	Port TD pin assignment 1. This field configures the port TD1 pin for its primary function (DTIN0), alternate 1 function ($\overline{\text{UCTS1}}$), alternate 2 function ($\overline{\text{UCTS0}}$) or digital I/O. 00 Port TD1 pin configured for digital I/O 01 Port TD1 pin configured for alternate 2 function ($\overline{\text{UCTS0}}$) 10 Port TD1 pin configured for alternate 1 function ($\overline{\text{UCTS1}}$) 11 Port TD1 pin configured for primary function (DTIN0)
1-0	PTDPA0	Port TD pin assignment 0. This field configures the port TD0 pin for its primary function (DTOUT0), alternate 1 function ($\overline{\text{UCTS1}}$), alternate 2 function ($\overline{\text{UCTS0}}$) or digital I/O. 00 Port TD0 pin configured for digital I/O 01 Port TD0 pin configured for alternate 2 function ($\overline{\text{UCTS0}}$) 10 Port TD0 pin configured for alternate 1 function ($\overline{\text{UCTS1}}$) 11 Port TD0 pin configured for primary function (DTOUT0)

26.3.2.15 Port UA Pin Assignment Register (PUAPAR)

The PUAPAR controls the pin function of port UA.

	7–4	3	2	1	0
Field	—	PUAPA3	PUAPA2	PUAPA1	PUAPA0
Reset	0000_0000				
R/W:	R	R/W			
Address	IPSBAR + 0x10_005C				

Figure 26-29. Port UA Pin Assignment Register (PUAPAR)

Table 26-19. PUAPAR Field Descriptions

Bits	Name	Description
7–4	—	Reserved, should be cleared.
3	PUAPA3	Port UA pin assignment 3. This bit configures the port UA3 pin for its primary function (URXD1) or digital I/O. 1 Port UA3 pin configured for primary function (URXD1) 0 Port UA3 pin configured for digital I/O

Table 26-19. PUAPAR Field Descriptions (continued)

Bits	Name	Description
2	PUAPA2	Port UA pin assignment 2. This bit configures the port UA2 pin for its primary function (UTXD1) or digital I/O. 1 Port UA2 pin configured for primary function (UTXD1) 0 Port UA2 pin configured for digital I/O
1	PUAPA1	Port UA pin assignment 1. This bit configures the port UA1 pin for its primary function (URXD0) or digital I/O. 1 Port UA1 pin configured for primary function (URXD0) 0 Port UA1 pin configured for digital I/O
0	PUAPA0	Port UA pin assignment 0. This bit configures the port UA0 pin for its primary function (UTXD0) or digital I/O. 1 Port UA0 pin configured for primary function (UTXD0) 0 Port UA0 pin configured for digital I/O

26.4 Functional Description

26.4.1 Overview

The initial pin function is determined during reset configuration. The pin assignment registers allow the user to select between digital I/O or another pin function after reset.

In single-chip mode, all pins are configured as digital I/O by default, except for debug data pins (DDATA[3:0]) and processor status pins (PST[3:0]). These pins are configured for their primary functions by default in all modes.

Every digital I/O pin is individually configurable as an input or an output via a data direction register (DDR*n*).

Every port has an output data register (PORT*n*) and a pin data register (PORT*n*P/SET*n*) to monitor and control the state of its pins. Data written to a PORT*n* register is stored and then driven to the corresponding port n pins configured as outputs.

Reading a PORT*n* register returns the current state of the register regardless of the state of the corresponding pins.

Reading a PORT*n*P/PSET*n* register returns the current state of the corresponding pins when configured as digital I/O, regardless of whether the pins are inputs or outputs.

Every port has a PORT*n*P/SET*n* register and a clear register (CLR*n*) for setting or clearing individual bits in the PORT*n* register.

In master mode, port A and B function as the upper external data bus, D[31:16]. When the PCDPA bit is set, ports C and D function as the lower external data bus, D[15:0]. Ports E–J are configured to support external memory functions.

The ports module does not generate interrupt requests.

26.4.2 Port Digital I/O Timing

Input data on all pins configured as digital I/O is synchronized to the rising edge of CLKOUT, as shown in Figure 26-30.

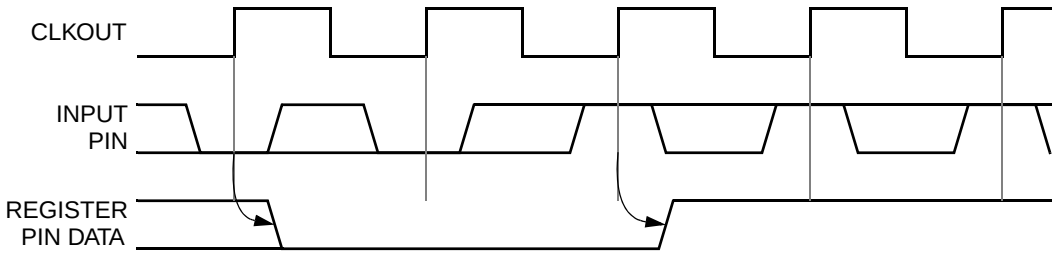


Figure 26-30. Digital Input Timing

Data written to the PORT*n* register of any pin configured as a digital output is immediately driven to its respective pin, as shown in Figure 26-31.

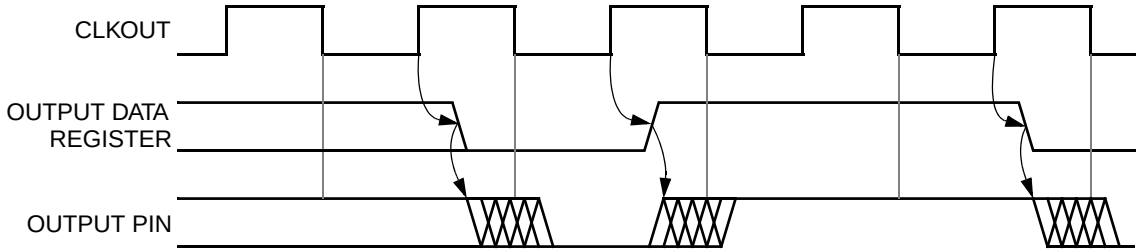


Figure 26-31. Digital Output Timing

26.5 Initialization/Application Information

The initialization for the ports module is done during reset configuration. Some of the registers are reset to a predetermined state, and others are reset according to which mode of operation is chosen by reset configuration. Refer to Section 26.3, “Memory Map/Register Definition,” for more details on reset and initialization.

Chapter 27
Chip Configuration Module (CCM)

The chip configuration module (CCM) controls the chip configuration and operating mode.

27.1 Features

The CCM performs these operations.

- Selects the chip operating mode:
  - Master mode
  - Single-chip mode
- Selects external clock or phase-lock loop (PLL) mode with internal or external reference
- Selects output pad drive strength
- Selects boot device and data port size
- Selects bus monitor configuration
- Selects low-power configuration
- Selects transfer size function of the external bus
- Selects processor status (PSTAT) and processor debug data (DDATA) functions
- Selects BDM or JTAG mode

27.2 Modes of Operation

The CCM configures the chip for two modes of operation:

- Master mode
- Single-chip mode

The operating mode is determined at reset and cannot be changed thereafter.

27.2.1 Master Mode

In master mode, the central processor unit (CPU) can access external memories and peripherals. The external bus consists of a 32-bit data bus and 24 address lines. The available bus control signals include R/$\overline{W}$, $\overline{TS}$, $\overline{TIP}$, TSIZ[1:0], $\overline{TA}$, $\overline{TEA}$, $\overline{OE}$, and $\overline{BS}$[3:0]. Up to seven chip selects can be programmed to select and control external devices and to provide bus cycle termination. When interfacing to 16-bit ports, the port C and D pins and PJ[5:4] ($\overline{BS}$[1:0]) can be configured as general-purpose input/output (I/O), and when interfacing to 8-bit ports, the ports B, C and D pins and PJ[7:5] ($\overline{BS}$[3:1]) can be configured as general purpose input/output (I/O).

27.2.2 Single-Chip Mode

In single-chip mode, all memory is internal to the chip. All external bus pins are configured as general purpose I/O.

27.3 Block Diagram

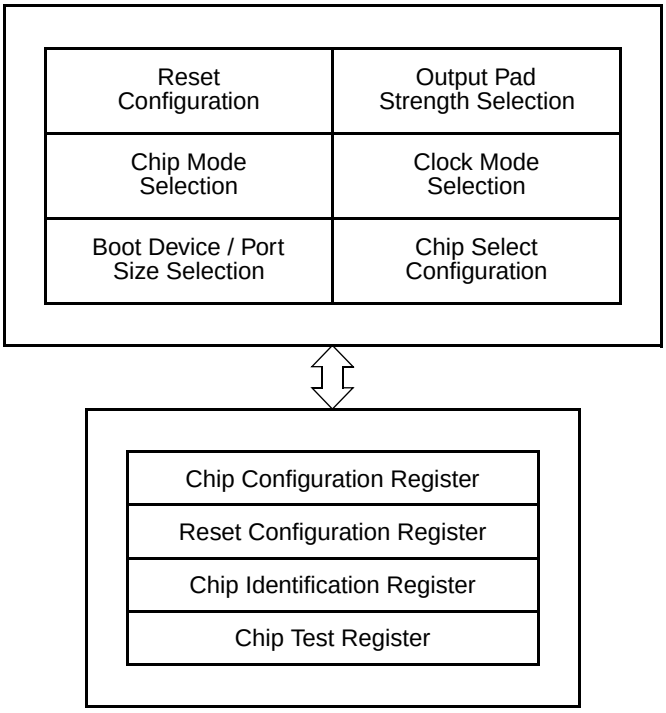


Figure 27-1. Chip Configuration Module Block Diagram

27.4 Signal Descriptions

Table 27-1 provides an overview of the CCM signals.

Table 27-1. Signal Properties

Name	Function	Reset State
$\overline{\text{RCON}}$	Reset configuration select	Internal weak pull-up device
CLKMOD[1:0]	Clock mode select	—
D[26:24, 21, 19:16]	Reset configuration override pins	—

27.4.1 $\overline{\text{RCON}}$

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 27.6, “Functional Description”). The internal configuration signals are driven to reflect the levels on the external configuration pins to allow for module configuration.

27.4.2 CLKMOD[1:0]

The state of the CLKMOD[1:0] pins during reset determines the clock mode.

27.4.3 D[26:24, 21, 19:16] (Reset Configuration Override)

If the external $\overline{\text{RCON}}$ pin is asserted during reset, then the states of these data pins during reset determine the chip mode of operation, boot device, clock mode, and certain module configurations after reset.

27.5 Memory Map and Registers

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

27.5.1 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 27-2 shows the accessibility of write-once bits.

Table 27-2. Write-Once Bits Read/Write Accessibility

Configuration	Read/Write Access
All configurations	Read-always
Debug operation (all modes)	Write-always
Master mode	Write-once
Single-chip mode	Write-once

27.5.2 Memory Map

Table 27-3. Chip Configuration Module Memory Map

IPSBAR Offset	Bits 31–16	Bits 15–0	Access[1]
0x0011_0004	Chip Configuration Register (CCR)	Low-Power Control Register (LPCR)[2]	S
0x0011_0008	Reset Configuration Register (RCON)	Chip Identification Register (CIR)	S
0x0011_000c	Reserved[3]		S
0x0011_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 7, “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.

27.5.3 Register Descriptions

The following subsection describes the CCM registers.

27.5.3.1 Chip Configuration Register (CCR)

	15	14–11	10–8	7	6	5	4	3	2–0
Field	LOAD	—	MODE	—	SZEN	PSTEN	—	BME	BMT
Reset	See Note								
R/W	R/W		Read Only	R/W					
Address	IPSBAR + 0x11_0004								

Note: The reset value of the LOAD and MODE fields is determined during reset configuration. The SZEN is set for master mode and cleared for all other configurations. The BME bit is set to enable the bus monitor and all other bits in the register are cleared at reset.

Figure 27-2. Chip Configuration Register (CCR)

Table 27-4. CCR Field Descriptions

Bits	Name	Description
15	LOAD	Pad driver load. The LOAD bit selects full or partial drive strength for selected pad output drivers. For maximum capacitive load, set the LOAD bit to select full drive strength. For reduced power consumption and reduced electromagnetic interference (EMI), clear the LOAD bit to select partial drive strength. 0 Default drive strength. 1 Full drive strength. Table 27-2 shows the read/write accessibility of this write-once bit.
14–11	—	Reserved, should be cleared.
10–8	MODE	Chip configuration mode. This read-only field reflects the chip configuration mode. 000-101 Reserved. 110 Single-chip mode. 111 Master mode.
7	—	Reserved, should be cleared.
6	SZEN	SIZ[1:0] enable. This read/write bit enables the SIZ[1:0] function of the external pins. 0 SIZ[1:0] function disabled. 1 SIZ[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.

Table 27-4. CCR Field Descriptions (continued)

Bits	Name	Description
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 27-2 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 27-2 shows the read/write accessibility of this write-once bit.

27.5.3.2 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 27.6.1, “Reset Configuration”) if the external RCON pin is asserted. RCON is a read-only register.

	15–10	9–8	7–6	5	4–3	2	1	0
Field	—	RCSC	—	RLOAD	BOOTPS	BOOTSEL	—	MODE
Reset	0000_0000_1110_0000							
R/W	R							
Address	IPSBAR + 0x11_0008							

Figure 27-3. Reset Configuration Register (RCON)

Table 27-5. RCON Field Descriptions

Bits	Name	Description
15–10	—	Reserved, should be cleared.
9–8	RCSC	Chip select configuration. Reflects the default chip select configuration. The default function of the chip select configuration can be overridden during reset configuration. 00 PF[7:5] = A[23:21] (This is the value used for this device.) 01 PF[7] = $\overline{CS6}$ / PF[6:5] = A[22:21] 10 PF[7:6] = $\overline{CS6}$, $\overline{CS5}$ / PF[5] = A[21] 11 PF[7:5] = $\overline{CS6}$, $\overline{CS5}$, $\overline{CS4}$
7–6	—	Reserved, should be cleared.
5	RLOAD	Pad driver load. Reflects the default pad driver strength configuration. 0 Partial drive strength 1 Full drive strength (This is the value used for this device.)

Table 27-5. RCON Field Descriptions (continued)

Bits	Name	Description
4–3	BOOTPS	Boot port size. Reflects the default selection for the boot port size if the boot device is configured to be external. 00 Internal (32 bits) (This is the value used for this device.) 01 16 bits 10 8 bits 11 32 bits The default function of the boot port size can be overridden during reset configuration.
2	BOOTSEL	Boot select. Reflects the default selection for the boot device. 0 Boot from internal boot device (This is the value used for this device.) 1 Boot from external boot device
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 this device.) 1 Master mode The default mode can be overridden during reset configuration. If the default is overridden, the CCR[MODE0] reflects the mode configuration.

27.5.3.3 Chip Identification Register (CIR)

	15 … 8	7 … 0
Field	PIN	PRN
Reset	0010_0000_0000_0000	
R/W	R	
Address	IPSBAR + 0x11_000a	

Figure 27-4. Chip Identification Register (CIR)

Table 27-6. CIR Field Description

Bits	Name	Description
15–8	PIN	Part identification number. Contains a unique identification number for the device.
7–0	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.

27.6 Functional Description

Six functions are defined within the chip configuration module:

1. Reset configuration
2. Chip mode selection
3. Boot device selection
4. Output pad strength configuration

5. Clock mode selections
6. Chip select configuration

These functions are described here.

27.6.1 Reset Configuration

During reset, the pins for the reset override functions are immediately configured to known states. Table 27-7 shows the states of the external pins while in reset.

Table 27-7. Reset Configuration Pin States During Reset

Pin	Pin Function[1]	I/O	Output State	Input State
D[26:24, 21, 19:16], PA[2:0], PB[5, 3:0]	Digital I/O or primary function	Input	—	Must be driven by external logic
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 27-8.

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 the RCON register 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 27-8) The internal configuration signals are driven to reflect the levels on the external configuration pins to allow for module configuration.

NOTE

During reset, the CLKMOD pins always determine the clock mode, regardless of the $\overline{\text{RCON}}$ pin state.

Table 27-8. Configuration During Reset[1]

Pin(s) Affected	Default Configuration	Override Pins in Reset[2,3,4]	Function
D[31:0], R/$\overline{\text{W}}$, $\overline{\text{TA}}$, $\overline{\text{TEA}}$, TSIZ[1:0], $\overline{\text{TS}}$, $\overline{\text{TIP}}$, $\overline{\text{OE}}$, A[23:0], BS[3:0], CS[3:0]	RCON0 = 0	**D[26,17:16]**	**Chip Mode Selected**
		111	Master mode
		110	Single-chip mode[5]
		100 or 0xx	Reserved

Table 27-8. Configuration During Reset[1] (continued)

Pin(s) Affected	Default Configuration	Override Pins in Reset[2,3,4]	Function
CS0	RCON[4:3] = 00 RCON2 = 0	**D[19:18]**	**Boot Device**
		00	Internal with 32-bit port[5]
		10	External with 8-bit port
		01	External with 16-bit port
		11	External with 32-bit port
All output pins	RCON5 = 1	**D21**	**Output Pad Drive Strength**
		0	Partial strength
		1	Full strength[5]
Clock mode	N/A	**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
A[23:21]/$\overline{CS}$[6:4]	RCON[9:8] = 00	**D[25:24]**	**Chip Select Configuration**
		00	PF[7:5] = A[23:21][5]
		10	PF[7] = $\overline{CS6}$ / PF[6:5] = A[22:21]
		01	PF[7:6] = $\overline{CS6}$, $\overline{CS5}$ / PF[5] = A[21]
		11	PF[7:6] = $\overline{CS6}$, $\overline{CS5}$, $\overline{CS4}$

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

2 The D[31:27, 23:22, 20, 15:0] pins do not affect reset configuration.

3 The external reset override circuitry drives the data bus pins with the override values while $\overline{RSTO}$ is asserted. It must stop driving the data bus pins within one CLKOUT cycle after $\overline{RSTO}$ 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 $\overline{RSTO}$ is negated.

4 RCON[0] has higher priority than RCON[3:2]. When RCON[0] is configured to boot the chip in single chip mode, the part will boot internally with a 32-bit port overriding any configuration set by RCON[3:2].

5 Default configuration

27.6.2 Chip Mode Selection

The chip mode is selected during reset and reflected in the MODE field of the chip configuration register (CCR). See Section 27.5.3.1, “Chip Configuration Register (CCR).” Once reset is exited, the operating mode cannot be changed. Table 27-9 shows the mode selection during reset configuration.

NOTE

When Flash security is enabled, the chip will boot in single chip mode regardless of the external reset configuration.

Table 27-9. Chip Configuration Mode Selection[1]

Chip Configuration Mode	CCR Register MODE Field		
	MODE2	MODE1	MODE0
Master mode	D26 driven high	D17 driven high	D16 driven high
Single-chip mode	D26 driven high	D17 driven high	D16 driven low
Reserved	D26 driven high	D17 driven low	D16 driven high
Reserved	D26 driven low	D17 don't care	D16 don't care

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

During reset, certain module configurations depend on whether emulation mode is active as determined by the state of the internal emulation signal.

27.6.3 Boot Device Selection

During reset configuration, the $\overline{\text{CS0}}$ chip select pin is optionally configured to select an external boot device. In this case, the V (valid) bit in the CSMR0 register is ignored, and $\overline{\text{CS0}}$ is enabled after reset. $\overline{\text{CS0}}$ is asserted for the initial boot fetch accessed from address 0x0000_0000 for the Stack Pointer and address 0x0000_0004 for the program counter (PC). It is assumed that the reset vector loaded from address 0x0000_0004 causes the CPU to start executing from external memory space decoded by $\overline{\text{CS0}}$.

27.6.4 Output Pad Strength Configuration

Output pad strength is determined during reset configuration as shown in Table 27-10. Once reset is exited, the output pad strength configuration can be changed by programming the LOAD bit of the chip configuration register.

Table 27-10. Output Pad Driver Strength Selection[1]

Optional Pin Function Selection	CCR Register LOAD Bit
Output pads configured for partial strength	D21 driven low
Output pads configured for full strength	D21 driven high

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

27.6.5 Clock Mode Selection

The clock mode is selected during reset by the CLKMOD pins and reflected in the PLLMODE, PLLSEL, and PLLREF bits of SYNSR. After reset is exited, the clock mode cannot be changed.

Table 27-11 summarizes clock mode selection during reset configuration.

Table 27-11. Clock Mode Selection

Clock Mode	CLKMOD[1:0]	Synthesizer Status Register (SYNSR)		
		PLLSEL	PLLREF	PLLMOD
External clock mode (PLL disabled)	00	0	0	0
1:1 PLL mode	01	0	0	1
Normal PLL mode, external clock reference	10	1	0	1
Normal PLL mode, crystal oscillator reference	11	1	1	1

27.6.6 Chip Select Configuration

The chip select configuration ($\overline{CS}$[6:4]) is selected during reset and reflected in the RCSC field of the CCR. Once reset is exited, the chip select configuration cannot be changed. Table 27-8 shows the different chip select configurations that can be implemented during reset configuration.

27.7 Reset

Reset initializes CCM registers to a known startup state as described in Section 27.5, “Memory Map and Registers.” The CCM controls chip configuration at reset as described in Section 27.6, “Functional Description.”

27.8 Interrupts

The CCM does not generate interrupt requests.

Chapter 28
Queued Analog-to-Digital Converter (QADC)

The queued analog-to-digital converter (QADC) is a 10-bit, unipolar, successive approximation converter. Up to eight analog input channels can be supported using internal multiplexing. A maximum of 18 input channels can be supported in the expanded, externally multiplexed mode.

The QADC consists of an analog front-end and a digital control subsystem. The analog section includes input pins, an analog multiplexer, and sample and hold analog circuits.

The digital control section contains queue control logic to sequence the conversion process and interrupt generation logic. Also included are the periodic/interval timer, control and status registers, the conversion command word (CCW) table, random-access memory (RAM), and the result table RAM.

The bus interface unit (BIU) provides access to registers that configure the QADC, control the analog-to-digital converter and queue mechanism, and present formatted conversion results.

28.1 Features

Features of the QADC module include:

- Internal sample and hold
- Up to eight analog input channels using internal multiplexing
- Up to four external analog multiplexers directly supported
- Up to 18 total input channels with internal and external multiplexing
- Programmable input sample time for various source impedances
- Two conversion command word (CCW) queues with a total of 64 entries for setting conversion parameters of each A/D conversion
- Subqueues possible using pause mechanism
- Queue complete and pause interrupts available on both queues
- Queue pointers indicating current location for each queue
- Automated queue modes initiated by:
  - External edge trigger and gated trigger
  - Periodic/interval timer, within QADC module (queues 1 and 2)
  - Software command
- Single scan or continuous scan of queues
- 64 result registers
- Output data readable in three formats:
  - Right-justified unsigned
  - Left-justified signed
  - Left-justified unsigned
- Unused analog channels can be used as discrete input/output pins.

28.2 Block Diagram

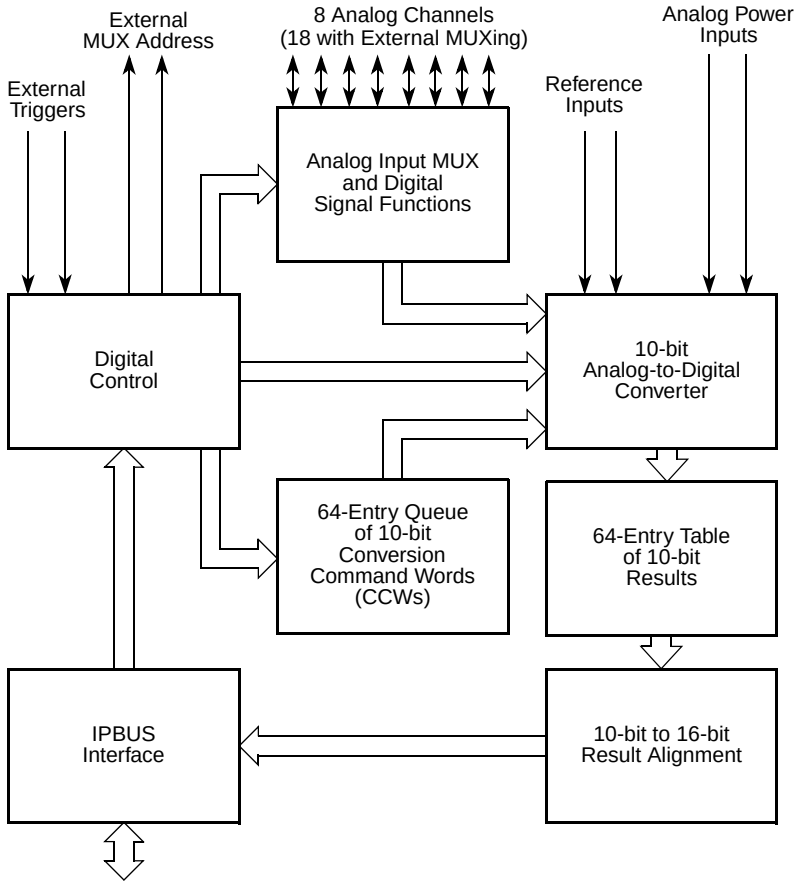


Figure 28-1. QADC Block Diagram

28.3 Modes of Operation

This subsection describes the two modes of operation in which the QADC does not perform conversions in a regular fashion:

- Debug mode
- Stop mode

28.3.1 Debug Mode

The QDBG bit in the module configuration register (QADCMCR) governs behavior of the QADC when the CPU enters background debug mode. When QDBG is clear, the QADC operates normally and is unaffected by CPU background debug mode. See Section 28.6.1, “QADC Module Configuration Register (QADCMCR).

When QDBG is set and the CPU enters background debug mode, the QADC finishes any conversion in progress and then freezes. This is QADC debug mode. Depending on when debug mode is entered, the three possible queue freeze scenarios are:

- When a queue is not executing, the QADC freezes immediately.
- When a queue is executing, the QADC completes the current conversion and then freezes.

- If during the execution of the current conversion, the queue operating mode for the active queue is changed, or a queue 2 abort occurs, the QADC freezes immediately.

When the QADC enters debug mode while a queue is active, the current CCW location of the queue pointer is saved.

Debug mode:

- Stops the analog clock
- Holds the periodic/interval timer in reset
- Prevents external trigger events from being captured
- Keeps all QADC registers and RAM accessible

Although the QADC saves a pointer to the next CCW in the current queue, software can force the QADC to execute a different CCW by reconfiguring the QADC. When the QADC exits debug mode, it looks at the queue operating modes, the current queue pointer, and any pending trigger events to decide which CCW to execute.

28.3.2 Stop Mode

The QADC enters a low-power idle state whenever the QSTOP bit is set or the CPU enters low-power stop mode.

QADC stop:

- Disables the analog-to-digital converter, effectively turning off the analog circuit
- Aborts the conversion sequence in progress
- Makes the data direction register (DDRQA), port data registers (PORTQA and PORTQB), control registers (QACR2, QACR1, and QACR0) and the status registers (QASR1 and QASR0) read-only. Only the module configuration register (QADCMCR) remains writable.
- Makes the RAM inaccessible, so that valid data cannot be read from RAM (result word table and CCW) or written to RAM (result word table and CCW)
- Resets QACR1, QACR2, QASR0, and QASR1
- Holds the QADC periodic/interval timer in reset

Because the bias currents to the analog circuit are turned off in stop mode, the QADC requires some recovery time (t_{SR}) to stabilize the analog circuits.

28.4 Signals

The QADC uses the external signals shown in Figure 28-2. There are eight channel/port signals that can support up to 18 channels when external multiplexing is used (including internal channels). All of the channel signals also have some general-purpose input or input/output (GPIO) functionality. In addition, there are also two analog reference signals and two analog submodule power signals.

The QADC has external trigger inputs and multiplexer outputs that are shared with some of the analog input signals.

28.4.1 Port QA Signal Functions

The four port QA signals can be used as analog inputs or as a bidirectional 4-bit digital input/output port.

28.4.1.1 Port QA Analog Input Signals

When used as analog inputs, the four port QA signals are referred to as AN[56:55, 53:52].

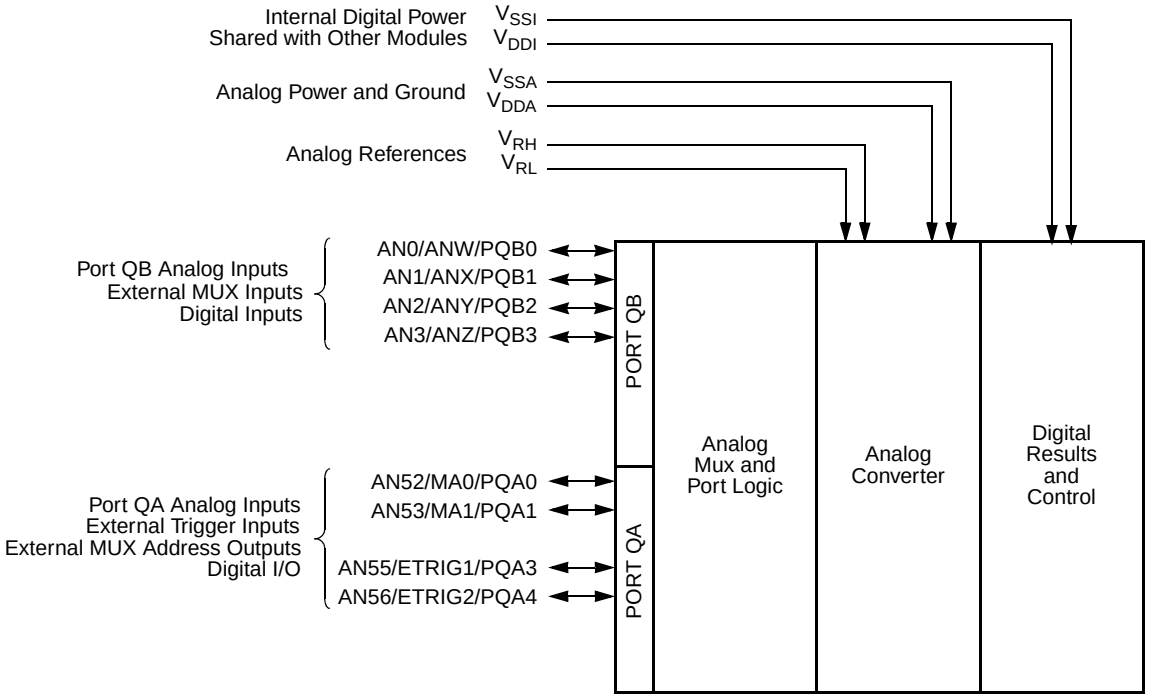


Figure 28-2. QADC Input and Output Signals

28.4.1.2 Port QA Digital Input/Output Signals

Port QA signals are referred to as PQA[4:3, 1:0] when used as a bidirectional 4-bit digital input/output port. These four signals may be used for general-purpose digital input or digital output.

Port QA signals are connected to a digital input synchronizer during reads and may be used as general-purpose digital inputs when the applied voltages meet high-voltage input (V_{IH}) and low-voltage input (V_{IL}) requirements.

Each port QA signal is configured as an input or output by programming the port data direction register (DDRQA). The digital input signal states are read from the port QA data register (PORTQA) when DDRQA specifies that the signals are inputs. The digital data in PORTQA is driven onto the port QA signals when the corresponding bits in DDRQA specify output. See Section 28.6.4, “Port QA and QB Data Direction Register (DDRQA & DDRQB).

28.4.2 Port QB Signal Functions

The four port QB signals can be used as analog inputs or as a 4-bit digital I/O port.

28.4.2.1 Port QB Analog Input Signals

When used as analog inputs, the four port QB signals are referred to as AN[3:0].

28.4.2.2 Port QB Digital I/O Signals

Port QB signals are referred to as PQB[3:0] when used as a 4-bit digital input/output port. In addition to functioning as analog input signals, the port QB signals are also connected to the input of a synchronizer during reads and may be used as general-purpose digital inputs when the applied voltages meet V_{IH} and V_{IL} requirements.

Each port QB signal is configured as an input or output by programming the port data direction register (DDRQB). The digital input signal states are read from the port QB data register (PORTQB) when DDRQB specifies that the signals are inputs. The digital data in PORTQB is driven onto the port QB signals when the corresponding bits in DDRQB specify output. See Section 28.6.4, “Port QA and QB Data Direction Register (DDRQA & DDRQB).

28.4.3 External Trigger Input Signals

The QADC has two external trigger signals, ETRIG2 and ETRIG1. Each external trigger input is associated with one of the scan queues, queue 1 or queue 2. The assignment of ETRIG[2:1] to a queue is made by the TRG bit in QADC control register 0 (QACR0). When TRG = 0, ETRIG1 triggers queue 1 and ETRIG2 triggers queue 2. When TRG = 1, ETRIG1 triggers queue 2 and ETRIG2 triggers queue 1. See Section 28.6.5, “Control Registers “Control Registers.”

28.4.4 Multiplexed Address Output Signals

In non-multiplexed mode, the QADC analog input signals are connected to an internal multiplexer which routes the analog signals into the internal A/D converter.

In externally multiplexed mode, the QADC allows automatic channel selection through up to four external 4-to-1 multiplexer chips. The QADC provides a 2-bit multiplexed address output to the external multiplexer chips to allow selection of one of four inputs. The multiplexed address output signals, MA1 and MA0, can be used as multiplexed address output bits or as general-purpose I/O when external multiplexed mode is not being used.

MA[1:0] are used as the address inputs for up to four 4-channel multiplexer chips. Because the MA[1:0] signals are digital outputs in multiplexed mode, the state of their corresponding data direction bits in DDRQA is ignored.

28.4.5 Multiplexed Analog Input Signals

In external multiplexed mode, four of the port QB signals are redefined so that each represent four analog input channels. See Table 28-1.

Table 28-1. Multiplexed Analog Input Channels

Multiplexed Analog Input	Channels
ANW	Even numbered channels from 0 to 6
ANX	Odd numbered channels from 1 to 7
ANY	Even numbered channels from 16 to 22
ANZ	Odd numbered channels from 17 to 23

28.4.6 Voltage Reference Signals

V_{RH} and V_{RL} are the dedicated input signals for the high and low reference voltages. Separating the reference inputs from the power supply signals allows for additional external filtering, which increases reference voltage precision and stability, and subsequently contributes to a higher degree of conversion accuracy.

NOTE

V_{RH} and V_{RL} must be set to V_{DDA} and V_{SSA} potential, respectively. For more information, refer to Section 28.9, "Signal Connection Considerations".

28.4.7 Dedicated Analog Supply Signals

The V_{DDA} and V_{SSA} signals supply power to the analog subsystems of the QADC module. Dedicated power is required to isolate the sensitive analog circuitry from the normal levels of noise present on the digital power supply.

28.4.8 Dedicated Digital I/O Port Supply Signal

V_{DDH} provides 5-V power to the digital I/O functions of QADC port QA and port QB. This allows those signals to tolerate 5 volts when configured as inputs and drive 5 volts when configured as outputs.

28.5 Memory Map

The QADC occupies 1 Kbyte, or 512 half-word (16-bit) entries, of address space. Ten half-word registers are control, port, and status registers, 64 half-word entries are the CCW table, and 64 half-word entries are the result table which occupies 192 half-word address locations because the result data is readable in three data alignment formats. Table 28-2 is the QADC memory map.

Table 28-2. QADC Memory Map

IPSBAR + Offset	MSB	LSB	Access[1]
0x19_0000	QADC Module Configuration Register (QADCMCR)		S
0x19_0002	QADC Test Register (QADCTEST)[2]		S
0x19_0004	Reserved[3]		—
0x19_0006	Port QA Data Register (PORTQA)	Port QB Data Register (PORTQB)	S/U
0x19_0008	Port QA Data Direction Register (DDRQA)	Port QB Data Direction Register (DDRQB)	S/U
0x19_000a	QADC Control Register 0 (QACR0)		S/U
0x19_000c	QADC Control Register 1 (QACR1)		S/U
0x19_000e	QADC Control Register 2 (QACR2)		S/U
0x19_0010	QADC Status Register 0 (QASR0)		S/U
0x19_0012	QADC Status Register 1 (QASR1)		S/U

Table 28-2. QADC Memory Map (continued)

IPSBAR + Offset	MSB	LSB	Access[1]
0x19_0014–0x19_01fe	Reserved[3]		—
0x19_0200–0x19_027e	Conversion Command Word Table (CCW)		S/U
0x19_0280–0x19_02fe	Right Justified, Unsigned Result Register (RJURR)		S/U
0x19_0300–0x19_037e	Left Justified, Signed Result Register (LJSRR)		S/U
0x19_0380–0x19_03fe	Left Justified, Unsigned Result Register (LJURR)		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 Access results in the module generating an access termination transfer error if not in test mode.

3 Read/writes have no effect and the access terminates with a transfer error exception.

28.6 Register Descriptions

This subsection describes the QADC registers.

28.6.1 QADC Module Configuration Register (QADCMCR)

The QADCMCR contains bits that control QADC debug and stop modes and determine the privilege level required to access most registers.

	15	14	13	8
Field	QSTOP	QDBG	—	
Reset	0000_0000			
R/W:	R/W		R	

	7	6	0
Field	SUPV	—	
Reset	1000_0000		
R/W:	R/W	R	
Address	IPSBAR + 0x19_0000, 0x19_0001		

Figure 28-3. QADC Module Configuration Register (QADCMCR)

Table 28-3. QADCMCR Field Descriptions

Bit(s)	Name	Description
15	QSTOP	Stop enable. 1 Force QADC to idle state. 0 QADC operates normally.
14	QDBG	Debug enable. 1 Finish any conversion in progress, then freeze in debug mode 0 QADC operates normally.
13–8	—	Reserved, should be cleared.
7	SUPV	Supervisor/unrestricted data space. 1 All QADC registers are accessible in supervisor mode only; user mode accesses have no effect and result in a cycle termination error. 0 Only QADCMCR and QADCTEST require supervisor mode access; access to all other QADC registers is unrestricted
6–0	—	Reserved, should be cleared.

28.6.2 QADC Test Register (QADCTEST)

The QADCTEST is a reserved register. Attempts to access this register outside of factory test mode will result in access privilege violation.

28.6.3 Port Data Registers (PORTQA & PORTQB)

QADC ports QA and QB are accessed through the 8-bit PORTQA and PORTQB.

Port QA signals are referred to as PQA[4:3, 1:0] when used as a bidirectional, 4-bit, input/output port. Port QA can also be used for analog inputs (AN[56:55, 53:52]), external trigger inputs (ETRIG[2:1]), and external multiplexer address outputs (MA[1:0]).

Port QB signals are referred to as PQB[3:0] when used as a 4-bit, digital input-only port. Port QB can also be used for non-multiplexed (AN[3:0]) and multiplexed (ANZ, ANY, ANX, ANW) analog inputs.

PORTQA and PORTQB are not initialized by reset.

	7	6	5	4	3	2	1	0
Field	—			PQA4 (AN56) (ETRIG2)	PQA3 (AN55) (ETRIG1)	—	PQA1 (AN53) (MA1)	PQA0 (AN52) (MA0)
Reset	000			See Note		0	See Note	
R/W:	R			R/W		R	R/W	
Address	IPSBAR + 0x19_0006							

Figure 28-4. QADC Port QA Data Register (PORTQA)

	7	6	5	4	3	2	1	0
Field	—				PQB3 (AN3) (ANZ)	PQB2 (AN2) (ANY)	PQA1 (AN1) (ANX)	PQA0 (AN0) (ANW)
Reset	0000				See Note			
R/W:	R				R/W			
Address	IPSBAR + 0x19_0007							

Figure 28-5. QADC Port QB Data Register (PORTQB)

Note: The reset value for these fields is the current signal state if DDR is an input; otherwise, they are undefined.

28.6.4 Port QA and QB Data Direction Register (DDRQA & DDRQB)

DDRQA and DDRQB are associated with port QA and QB digital I/O signals. Setting a bit in these registers configures the corresponding signal as an output. Clearing a bit in these registers configures the corresponding signal as an input. During QADC initialization, port QA and QB signals that will be used as direct or multiplexed analog inputs must have their corresponding data direction register bits cleared. When a port QA or QB signal that is programmed as an output is selected for analog conversion, the voltage sampled is that of the output digital driver as influenced by the load.

When the MUX (externally multiplexed) bit is set in QACR0, the data direction register settings are ignored for the bits corresponding to PQA[1:0], and the two multiplexed address (MA[1:0]) output signals. The MA[1:0] signals are forced to be digital outputs, regardless of their data direction setting, and the multiplexed address outputs are driven. The data returned during a port data register read is the value of the MA[1:0] signals, regardless of their data direction setting.

Similarly, when the external trigger signals are assigned to port signals and external trigger queue operating mode is selected, the data direction setting for the corresponding signals, PQA3 and/or PQA4, is ignored. The port signals are forced to be digital inputs for ETRIG1 and/or ETRIG2. The data returned during a port data register read is the value of the ETRIG[2:1] signals, regardless of their data direction setting.

NOTE

Use caution when mixing digital and analog inputs. They should be isolated as much as possible. Rise and fall times should be as large as possible to minimize ac coupling effects.

	7	6	5	4	3	2	1	0
Field	—			DDQA4	DDQA3	—	DDQA1	DDQA0
Reset	0000_0000							
R/W:	R			R/W		R	R/W	
Address	IPSBAR + 0x19_0008							

Figure 28-6. QADC Port QA Data Direction Register (DDRQA)

	7	6	5	4	3	2	1	0
Field	—				DDQB3	DDQB2	DDQB1	DDQB0
Reset	0000_0000							
R/W	R							
Address	IPSBAR + 0x19_0009							

Figure 28-7. Port QB Data Direction Register (DDRQB)

28.6.5 Control Registers

This subsection describes the QADC control registers.

28.6.5.1 QADC Control Register 0 (QACR0)

QACR0 establishes the QADC sampling clock (QCLK) with prescaler parameter fields and defines whether external multiplexing is enabled. Typically, these bits are written once when the QADC is initialized and not changed thereafter. The bits in this register are read anytime, write anytime (except during stop mode).

	15	14	13	12	11	8
Field	MUX	—		TRG	—	
Reset	0000_0000					
R/W:	R/W	R		R/W	R	

	7	6	5	4	3	2	1	0
Field	—	QPR6	QPR5	QPR4	QPR3	QPR2	QPR1	QPR0
Reset	0001_0011							
R/W:	R	R/W						
Address	IPSBAR + 0x19_000a, 0x19_000b							

Figure 28-8. QADC Control Register 0 (QACR0)

Table 28-4. QACR0 Field Descriptions

Bit(s)	Name	Description
15	MUX	Externally multiplexed mode. Configures the QADC for operation in externally multiplexed mode, which affects the interpretation of the channel numbers and forces the MA[1:0] signals to be outputs. 1 Externally multiplexed, up to 18 possible channels 0 Internally multiplexed, up to 8 possible channels
14–13	—	Reserved, should be cleared.
12	TRG	Trigger assignment. Determines the queue assignment of the ETRIG[2:1] signals. 1 ETRIG1 triggers queue 2; ETRIG2 triggers queue 1. 0 ETRIG1 triggers queue 1; ETRIG2 triggers queue 2.

Table 28-4. QACR0 Field Descriptions (continued)

Bit(s)	Name	Description
11–7	—	Reserved, should be cleared.
6–0	QPR	Prescaler clock divider. Selects the system clock divisor to generate the QADC clock as Table 28-5 shows. The resulting QADC clock rate can be given as: $f_{QCLK} = \frac{f_{SYS}}{2(QPR[6:0] + 1)}$ where: $1 \le QPR[6:0] \le 127$. If QPR[6:0] = 0, then the QPR register field value is read as a 1 and the prescaler divisor is 2. The prescaler should be selected so that the QADC clock rate is within the required f_{QCLK} range. See Chapter 33, “Electrical Characteristics”.

Table 28-5. Prescaler f_{SYS} Divide-by Values

QPR[6:0]	f_{SYS} Divisor	QPR[6:0]	f_{SYS} Divisor	QPR[6:0]	f_{SYS} Divisor	QPR[6:0]	f_{SYS} Divisor
0000000	4	0100000	66	1000000	130	1100000	194
0000001	4	0100001	68	1000001	132	1100001	196
0000010	6	0100010	70	1000010	134	1100010	198
0000011	8	0100011	72	1000011	136	1100011	200
0000100	10	0100100	74	1000100	138	1100100	202
0000101	12	0100101	76	1000101	140	1100101	204
0000110	14	0100110	78	1000110	142	1100110	206
0000111	16	0100111	80	1000111	144	1100111	208
0001000	18	0101000	82	1001000	146	1101000	210
0001001	20	0101001	84	1001001	148	1101001	212
0001010	22	0101010	86	1001010	150	1101010	214
0001011	24	0101011	88	1001011	152	1101011	216
0001100	26	0101100	90	1001100	154	1101100	218
0001101	28	0101101	92	1001101	156	1101101	220
0001110	30	0101110	94	1001110	158	1101110	222
0001111	32	0101111	96	1001111	160	1101111	224
0010000	34	0110000	98	1010000	162	1110000	226
0010001	36	0110001	100	1010001	164	1110001	228
0010010	38	0110010	102	1010010	166	1110010	230
0010011	40	0110011	104	1010011	168	1110011	232
0010100	42	0110100	106	1010100	170	1110100	234

Table 28-5. Prescaler f_{SYS} Divide-by Values (continued)

QPR[6:0]	f_{SYS} Divisor	QPR[6:0]	f_{SYS} Divisor	QPR[6:0]	f_{SYS} Divisor	QPR[6:0]	f_{SYS} Divisor
0010101	44	0110101	108	1010101	172	1110101	236
0010110	46	0110110	110	1010110	174	1110110	238
0010111	48	0110111	112	1010111	176	1110111	240
0011000	50	0111000	114	1011000	178	1111000	242
0011001	52	0111001	116	1011001	180	1111001	244
0011010	54	0111010	118	1011010	182	1111010	246
0011011	56	0111011	120	1011011	184	1111011	248
0011100	58	0111100	122	1011100	186	1111100	250
0011101	60	0111101	124	1011101	188	1111101	252
0011110	62	0111110	126	1011110	190	1111110	254
0011111	64	0111111	128	1011111	192	1111111	256

28.6.5.2 QADC Control Register 1 (QACR1)

QACR1 is the mode control register for queue 1. This register governs queue operating mode and the use of completion and/or pause interrupts. Typically, these bits are written once when the QADC is initialized and are not changed thereafter.

Stop mode resets this register.

	15	14	13	12	11	10	9	8
Field	CIE1	PIE1	SSE1	MQ112	MQ111	MQ110	MQ19	MQ18
Reset	0000_0000							
R/W:	R/W							

	7	0
Field	—	
Reset	0000_0000	
R/W:	R	
Address	IPSBAR + 0x19_000c, 0x19_000d	

Figure 28-9. QADC Control Register 1 (QACR1)

Table 28-6. QACR1 Field Descriptions

Bit(s)	Name	Description
15	CIE1	Queue 1 completion interrupt enable. Enables an interrupt request upon completion of queue 1. The interrupt request is initiated when the conversion is complete for the last CCW in queue 1. 1 Enable queue 1 completion interrupt. 0 Disable queue 1 completion interrupt.
14	PIE1	Queue 1 pause interrupt enable. Enables an interrupt request when queue 1 enters the pause state. The interrupt request is initiated when conversion is complete for a CCW that has the pause bit set. 1 Enable the queue 1 pause interrupt. 0 Disable the queue 1 pause interrupt.
13	SSE1	Queue 1 single-scan enable. Enables a single-scan of queue 1 after a trigger event occurs. SSE1 may be set during the same write cycle that sets the MQ1 bits for one of the single-scan queue operating modes. The single-scan enable bit can be written to 1 or 0, but is always read as a 0, unless the QADC is in test mode. The QADC clears SSE1 when the single-scan is complete. 1 Allow a trigger event to start queue 1 in a single-scan mode. 0 Trigger events are ignored for queue 1 single-scan modes.
12–8	MQ1*n*	Selects the operating mode for queue 1. Table 28-7 shows the bits in the MQ1 field which enable different queue 1 operating modes.
7–0	—	Reserved, should be cleared.

Table 28-7. Queue 1 Operating Modes

MQ1[12:8]	Operating Mode
00000	Disabled mode, conversions do not occur
00001	Software-triggered single-scan mode (started with SSE1)
00010	External-trigger rising-edge single-scan mode
00011	External-trigger falling-edge single-scan mode
00100	Interval timer single-scan mode: time = QCLK period × 2^7
00101	Interval timer single-scan mode: time = QCLK period × 2^8
00110	Interval timer single-scan mode: time = QCLK period × 2^9
00111	Interval timer single-scan mode: time = QCLK period × 2^{10}
01000	Interval timer single-scan mode: time = QCLK period × 2^{11}
01001	Interval timer single-scan mode: time = QCLK period × 2^{12}
01010	Interval timer single-scan mode: time = QCLK period × 2^{13}
01011	Interval timer single-scan mode: time = QCLK period × 2^{14}
01100	Interval timer single-scan mode: time = QCLK period × 2^{15}
01101	Interval timer single-scan mode: time = QCLK period × 2^{16}
01110	Interval timer single-scan mode: time = QCLK period × 2^{17}
01111	Externally gated single-scan mode (started with SSE1)

Table 28-7. Queue 1 Operating Modes (continued)

MQ1[12:8]	Operating Mode
10000	Reserved mode
10001	Software-triggered continuous-scan mode
10010	External-trigger rising-edge continuous-scan mode
10011	External-trigger falling-edge continuous-scan mode
10100	Periodic timer continuous-scan mode: time = QCLK period × 2^7
10101	Periodic timer continuous-scan mode: time = QCLK period × 2^8
10110	Periodic timer continuous-scan mode: time = QCLK period × 2^9
10111	Periodic timer continuous-scan mode: time = QCLK period × 2^{10}
11000	Periodic timer continuous-scan mode: time = QCLK period × 2^{11}
11001	Periodic timer continuous-scan mode: time = QCLK period × 2^{12}
11010	Periodic timer continuous-scan mode: time = QCLK period × 2^{13}
11011	Periodic timer continuous-scan mode: time = QCLK period × 2^{14}
11100	Periodic timer continuous-scan mode: time = QCLK period × 2^{15}
11101	Periodic timer continuous-scan mode: time = QCLK period × 2^{16}
11110	Periodic timer continuous-scan mode: time = QCLK period × 2^{17}
11111	Externally gated continuous-scan mode

28.6.5.3 QADC Control Register 2 (QACR2)

QACR2 is the mode control register for queue 2. This register governs queue operating mode and the use of completion and/or pause interrupts. Typically, these bits are written once when the QADC is initialized and not changed thereafter.

QACR2 also includes a resume feature that selects the resumption point for queue 2 after its operation is suspended by a queue 1 trigger event. The primary reason for selecting re-execution of the entire queue or subqueue is to guarantee that all samples are taken consecutively in one scan (coherency).

When subqueues are not used, queue 2 execution restarts after suspension with the first CCW in queue 2. When a pause has previously occurred in queue 2 execution, queue execution restarts after suspension with the first CCW in the current subqueue.

A subqueue is considered to be a stand-alone sequence of conversions. Once a pause flag has been set to report subqueue completion, that subqueue is not repeated until all CCWs in queue 2 are executed.

For example, the RESUME bit can be used when the frequency of queue 1 trigger events prohibit queue 2 completion. If the rate of queue 1 execution is too high, it is best for queue 2 execution to continue with the CCW that was being converted when queue 2 was suspended. This allows queue 2 to eventually complete execution.

The beginning of queue 2 is defined by programming the BQ2 field in QACR2. BQ2 is usually set before or at the same time as the queue operating mode for queue 2 is selected. If BQ2[6:0] ≥ 64, queue 2 has no entries, the entire CCW table is dedicated to queue 1, and CCW63 is the end-of-queue 1. If BQ2[6:0] is 0, the entire CCW table is dedicated to queue 2. A special case occurs when an operating mode is selected

for queue 1 and a trigger event occurs for queue 1 with BQ2 set to 0. Queue 1 execution starts momentarily, but is terminated after CCW0 is read. No conversions occur.

The BQ2[6:0] pointer may be changed dynamically to alternate between queue 2 scan sequences. A change in BQ2 after queue 2 has begun or when queue 2 has a trigger pending does not affect queue 2 until it is started again. For example, two scan sequences could be defined as follows: The first sequence starts at CCW10, with a pause after CCW11 and an end of queue (EOQ) programmed in CCW15; the second sequence starts at CCW16, with a pause after CCW17 and an EOQ programmed in CCW39.

With BQ2[6:0] set to CCW10 and the continuous-scan mode selected, queue execution begins. When the pause is encountered in CCW11, an interrupt service routine can retarget BQ2[6:0] to CCW16. When the end-of-queue is recognized in CCW15, an internal retrigger event is generated and execution restarts at CCW16. When the pause software interrupt occurs again, BQ2 can be changed back to CCW10. After the end-of-queue is recognized in CCW39, an internal retrigger event is created and execution now restarts at CCW10.

If BQ2[6:0] is changed while queue 1 is active, the effect of BQ2[6:0] as an end-of-queue indication for queue 1 is immediate. However, beware of the risk of losing the end-of-queue 1 when changing BQ2[6:0]. Using EOQ (channel 63) to end queue 1 is recommended.

NOTE

If BQ2[6:0] was assigned to the CCW that queue 1 is currently working on, then that conversion is completed before the change to BQ2[6:0] takes effect.

Each time a CCW is read for queue 1, the CCW location is compared with the current value of the BQ2[6:0] pointer to detect a possible end-of-queue condition. For example, if BQ2[6:0] is changed to CCW3 while queue 1 is converting CCW2, queue 1 is terminated after the conversion is completed. However, if BQ2[6:0] is changed to CCW1 while queue 1 is converting CCW2, the QADC would not recognize a BQ2[6:0] end-of-queue condition until queue 1 execution reached CCW1 again, presumably on the next pass through the queue.

Stop mode resets this register (0x007f)

	15	14	13	12	11	10	9	8
Field	CIE2	PIE2	SSE2	MQ212	MQ211	MQ210	MQ29	MQ28
Reset	0000_0000							
R/W:	R/W							

	7	6	5	4	3	2	1	0
Field	RESUME	BQ26	BQ25	BQ24	BQ23	BQ22	BQ21	BQ20
Reset	0111_1111							
R/W:	R/W							
Address	IPSBAR + 0x19_000e, 0x19_000f							

Figure 28-10. QADC Control Register 2 (QACR2)

Table 28-8. QACR2 Field Descriptions

Bit(s)	Name	Description
15	CIE2	Queue 2 completion software interrupt enable. Enables an interrupt request upon completion of queue 2. The interrupt request is initiated when the conversion is complete for the last CCW in queue 2. 1 Enable queue 2 completion interrupt. 0 Disable queue 2 completion interrupt.
14	PIE2	Queue 2 pause interrupt enable. Enables an interrupt request when queue 2 enters the pause state. The interrupt request is initiated when conversion is complete for a CCW that has the pause bit set. 1 Enable the queue 2 pause interrupt. 0 Disable the queue 2 pause interrupt.
13	SSE2	Queue 2 single-scan enable. Enables a single-scan of queue 2 after a trigger event occurs. SSE2 may be set during the same write cycle that sets the MQ2 bits for one of the single-scan queue operating modes. The single-scan enable bit can be written to 1 or 0, but is always read as a 0, unless the QADC is in test mode. The QADC clears SSE2 when the single-scan is complete. 1 Allow a trigger event to start queue 2 in a single-scan mode. 0 Trigger events are ignored for queue 2 single-scan modes.
12–8	MQ2	Selects the operating mode for queue 2. Table 28-9 shows the bits in the MQ1 field which enable different queue 2 operating modes.
7	RESUME	Selects the resumption point for queue 2 after its operation is suspended due to a queue 1 trigger event. If RESUME is changed during the execution of queue 2, the change is not recognized until an end-of-queue condition is reached or the operating mode of queue 2 is changed. 1 After suspension, begin execution with the aborted CCW in queue 2. 0 After suspension, begin execution with the first CCW of queue 2 or the current subqueue of queue 2.
6–0	BQ2	Beginning of queue 2. Denotes the CCW location where queue 2 begins. This allows the length of queue 1 and queue 2 to vary. The BQ2 field also serves as an end-of-queue condition for queue 1.

Table 28-9. Queue 2 Operating Modes

MQ2[12:8]	Operating Modes
00000	Disabled mode, conversions do not occur
00001	Software triggered single-scan mode (started with SSE2)
00010	Externally triggered rising edge single-scan mode
00011	Externally triggered falling edge single-scan mode
00100	Interval timer single-scan mode: time = QCLK period x 2^7
00101	Interval timer single-scan mode: time = QCLK period x 2^8
00110	Interval timer single-scan mode: time = QCLK period x 2^9
00111	Interval timer single-scan mode: time = QCLK period x 2^{10}
01000	Interval timer single-scan mode: time = QCLK period x 2^{11}
01001	Interval timer single-scan mode: time = QCLK period x 2^{12}

Table 28-9. Queue 2 Operating Modes (continued)

MQ2[12:8]	Operating Modes
01010	Interval timer single-scan mode: time = QCLK period x 2^{13}
01011	Interval timer single-scan mode: time = QCLK period x 2^{14}
01100	Interval timer single-scan mode: time = QCLK period x 2^{15}
01101	Interval timer single-scan mode: time = QCLK period x 2^{16}
01110	Interval timer single-scan mode: time = QCLK period x 2^{17}
01111	Reserved mode
10000	Reserved mode
10001	Software triggered continuous-scan mode
10010	Externally triggered rising edge continuous-scan mode
10011	Externally triggered falling edge continuous-scan mode
10100	Periodic timer continuous-scan mode: time = QCLK period x 2^{7}
10101	Periodic timer continuous-scan mode: time = QCLK period x 2^{8}
10110	Periodic timer continuous-scan mode: time = QCLK period x 2^{9}
10111	Periodic timer continuous-scan mode: time = QCLK period x 2^{10}
11000	Periodic timer continuous-scan mode: time = QCLK period x 2^{11}
11001	Periodic timer continuous-scan mode: time = QCLK period x 2^{12}
11010	Periodic timer continuous-scan mode: time = QCLK period x 2^{13}
11011	Periodic timer continuous-scan mode: time = QCLK period x 2^{14}
11100	Periodic timer continuous-scan mode: time = QCLK period x 2^{15}
11101	Periodic timer continuous-scan mode: time = QCLK period x 2^{16}
11110	Periodic timer continuous-scan mode: time = QCLK period x 2^{17}
11111	Reserved mode

28.6.6 Status Registers

This subsection describes the QADC status registers.

28.6.6.1 QADC Status Register 0 (QASR0)

QASR0 contains information about the state of each queue and the current A/D conversion. The bits in this register are read anytime. For flag bits (CF1, PF1, CF2, PF2, TOR1, TOR2), writing a 1 has no effect; writing a 0 clears the bit. For QS[9:6] and CWP, writes have no effect. Stop mode resets this register.

The end of a queue is identified in the following cases:

- When execution is complete on the CCW in the location prior to the one pointed to by BQ2
- When the current CCW contains the end-of-queue code (channel 63) instead of a valid channel number

- When the currently completed CCW is in the last location of the CCW RAM.

Once PF*n* is set, the queue enters the paused state and waits for a trigger event to allow queue execution to continue. However, a special case occurs when the CCW with the pause bit set is the last CCW in a queue; queue execution is complete. The queue status becomes idle, not paused, and both the pause and completion flags are set.

Another special case occurs when the queue is operating in software-initiated single-scan or continuous-scan mode and a CCW pause bit is set. The QADC will set PF*n* and will also automatically generate a retrigger event that restarts execution after two QCLK cycles. Pause mode is never entered.

In externally gated single-scan and continuous-scan mode, the behavior of PF*n* has been redefined. When the gate closes before the end of the queue is reached, PF*n* is set to indicate that an incomplete scan has occurred. In single-scan mode, a resultant interrupt can be used to determine if the queue should be enabled again. In either externally gated mode, setting PF*n* indicates that the results for the queue have not been collected during one scan (coherently).

NOTE:

If a set CCW pause bit is encountered in either externally gated mode, the pause flag will not set, and execution continues without pausing. This has allowed for the modified behavior of PF1 in the externally gated modes.

PF*n* is maintained by the QADC regardless of whether the corresponding interrupt is enabled. PF*n* may be polled to determine if the QADC has reached a pause in scanning a queue.

A trigger event generated by a transition on the external trigger signal or by the periodic/interval timer may be captured as a trigger overrun. TOR*n* cannot be set when the software-initiated single-scan mode or the software-initiated continuous-scan mode is selected.

TOR*n* is set when a trigger event is received while a queue is executing and before the scan has completed or paused. TOR*n* has no effect on queue execution.

After a trigger event has occurred for queue 1, and before the scan has completed or paused, additional queue 1 trigger events are not retained. Such trigger events are considered unexpected, and the QADC sets the TOR*n* error status bit. An unexpected trigger event may denote a system overrun situation.

In externally gated continuous-scan mode, the behavior of TOR*n* has been redefined. In the case that the queue reaches an end-of-queue condition for the second time during an open gate, TOR*n* is set. This is considered an overrun condition. In this case, CF1 has been set for the first end-of-queue condition and TOR*n* sets for the second end-of-queue condition. For TOR1 to set, CF2 must not be cleared before the second end-of-queue.

The QS field indicates the status of queue 1 and queue 2. Following are the five queue status conditions:

- Idle
- Active
- Paused
- Suspended
- Trigger pending

The idle state occurs when a queue is disabled, when a queue is in a reserved mode, or when a queue is in a valid queue operating mode awaiting a trigger event to initiate queue execution. One or both queues may be in the idle state. When a queue is idle, CCWs are not being executed for that queue, the queue is not in the pause state, and no trigger is pending.

A queue is in the active state when a valid queue operating mode is selected, when the selected trigger event has occurred, or when the QADC is performing a conversion specified by a CCW from that queue. Only one queue can be active at a time.

One or both queues can be in the paused state. A queue is paused when the previous CCW executed from that queue had the pause bit set. The QADC does not execute any CCWs from the paused queue until a trigger event occurs. Consequently, the QADC can service queue 2 while queue 1 is paused.

Only queue 2 can be in the suspended state. When a trigger event occurs on queue 1 while queue 2 is executing, the current queue 2 conversion is aborted and the queue 2 status is reported as suspended. Queue 2 transitions back to the active state when queue 1 becomes idle or paused.

A trigger pending state is required because both queues cannot be active at the same time. The status of queue 2 is changed to trigger pending when a trigger event occurs for queue 2 while queue 1 is active. In the opposite case, when a trigger event occurs for queue 1 while queue 2 is active, queue 2 is aborted and the status is reported as queue 1 active, queue 2 suspended. So due to the priority scheme, only queue 2 can be in the trigger pending state.

Two transition cases cause the queue 2 status to be trigger pending before queue 2 is shown to be in the active state. When queue 1 is active and there is a trigger pending on queue 2, after queue 1 completes or pauses, queue 2 continues to be in the trigger pending state for a few clock cycles. The fleeting status conditions are:

- Queue 1 idle with queue 2 trigger pending
- Queue 1 paused with queue 2 trigger pending

Figure 28-12 displays the status conditions of the QS field as the QADC goes through the transition from queue 1 active to queue 2 active.

When a queue enters the paused state, CWP points to the CCW with the pause bit set. While in pause, the CWP value is maintained until a trigger event occurs on either queue. Usually, the CWP is updated a few clock cycles before the queue status field shows that the queue has become active. For example, a read of CWP may point to a CCW in queue 2, while the queue status field shows queue 1 paused and queue 2 trigger pending.

When the QADC finishes a queue scan, the CWP points to the CCW where the end-of-queue condition was detected. Therefore, when the end-of-queue condition is a CCW with the EOQ code (channel 63), the CWP points to the CCW containing the EOQ.

When the last CCW in a queue is the last CCW table location (CCW63), and it does not contain the EOQ code, the end-of-queue is detected when the following CCW is read, so the CWP points to word CCW0.

Finally, when queue 1 operation is terminated after a CCW is read that is pointed to by BQ2, the CWP points to the same CCW as BQ2.

	15	14	13	12	11	10	9	8
Field	CF1	PF1	CF2	PF2	TOR1	TOR2	QS9	QS8
Reset	0000_0000							
R/W:	R/W						R	

	7	6	5	4	3	2	1	0
Field	QS7	QS6	CWP5	CWP4	CWP3	CWP2	CWP1	CWP0
Reset	0000_0000							
R/W:	R							
Address	IPSBAR + 0x19_0010, 0x19_0011							

Figure 28-11. QADC Status Register 0 (QASR0)

Table 28-10. QASR0 Field Descriptions

Bit(s)	Name	Description
15, 13	CF*n*	Queue completion flag. Indicates that a queue scan has been completed. CF[1:2] is set by the QADC when the input channel sample requested by the last CCW in the queue is converted, and the result is stored in the result table. When CF*n* is set and queue completion interrupts are enabled (QACR*n*[CIE*n*] = 1), the QADC requests an interrupt. The interrupt request is cleared when a 0 is written to the CF1 bit after it has been read as a 1. Once set, CF1 can be cleared only by a reset or by writing a 0 to it. CF[1:2] is updated by the QADC regardless of whether the corresponding interrupt is enabled. This allows polled recognition of the queue scan completion.
14, 12	PF*n*	Queue pause flag. Indicates that a queue scan has reached a pause. PF[1:2] is set by the QADC when the current queue 1 CCW has the pause bit set, the selected input channel has been converted, and the result has been stored in the result table. When PF*n* is set and interrupts are enabled (QACR*n*[PIE*n*] = 1), the QADC requests an interrupt. The interrupt request is cleared when a 0 is written to PF*n*, after it has been read as a 1. Once set, PF*n* can be cleared only by reset or by writing a 0 to it. PF1: 1 Queue 1 has reached a pause or gate closed before end-of-queue in gated mode. 0 Queue 1 has not reached a pause or gate has not closed before end-of-queue in gated mode. PF2: 1 Queue 2 has reached a pause. 0 Queue 2 has not reached a pause. See Table 28-11 for a summary of CCW pause bit response in all scan modes.
11–10	TOR*n*	Queue trigger overrun flag. Indicates that an unexpected trigger event has occurred for queue 1. TOR[1:2] can be set only while the queue is in the active state. Once set, TOR[1:2] is cleared only by a reset or by writing a 0 to it. 1 At least one unexpected queue 1 trigger event has occurred or queue 1 reaches an end-of-queue condition for the second time in externally gated continuous scan. 0 No unexpected queue 1 trigger events have occurred.

Table 28-10. QASR0 Field Descriptions (continued)

Bit(s)	Name	Description
9–6	QS	Queue status. Indicates the current condition of queue 1 and queue 2. The two most significant bits are associated primarily with queue 1, and the remaining two bits are associated with queue 2. Because the priority scheme between the two queues causes the status to be interlinked, the status bits must be considered as one 4-bit field. Table 28-12 shows the bits in the QS field and how they denote the status of queue 1 and queue 2. The queue status field is affected by QADC stop mode. Because all of the analog logic and control registers are reset, the queue status field is reset to queue 1 idle, queue 2 idle. During debug mode, the queue status field is not modified. The queue status field retains the status it held prior to freezing. As a result, the queue status can show queue 1 active, queue 2 idle, even though neither queue is being executed during freeze.
5–0	CWP	Command word pointer. Denotes which CCW is executing at present or was last completed. CWP is a read-only field with a valid range of 0 to 63; write operations have no effect. During stop mode, CWP is reset to 0 because the control registers and the analog logic are reset. When debug mode is entered, CWP is not changed; it points to the last executed CCW.

Table 28-11. CCW Pause Bit Response

Scan Mode	Queue Operation	PF Asserts?
Externally triggered single-scan	Pauses	Yes
Externally triggered continuous-scan	Pauses	Yes
Interval timer trigger single-scan	Pauses	Yes
Interval timer continuous-scan	Pauses	Yes
Software-initiated single-scan	Continues	Yes
Software-initiated continuous-scan	Continues	Yes
Externally gated single-scan	Continues	No
Externally gated continuous-scan	Continues	No

Table 28-12. Queue Status

QS[9:6]	Queue 1/Queue 2 States
0000	Queue 1 idle, queue 2 idle
0001	Queue 1 idle, queue 2 paused
0010	Queue 1 idle, queue 2 active
0011	Queue 1 idle, queue 2 trigger pending
0100	Queue 1 paused, queue 2 idle
0101	Queue 1 paused, queue 2 paused
0110	Queue 1 paused, queue 2 active

Table 28-12. Queue Status (continued)

QS[9:6]	Queue 1/Queue 2 States
0111	Queue 1 paused, queue 2 trigger pending
1000	Queue 1 active, queue 2 idle
1001	Queue 1 active, queue 2 paused
1010	Queue 1 active, queue 2 suspended
1011	Queue 1 active, queue 2 trigger pending
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

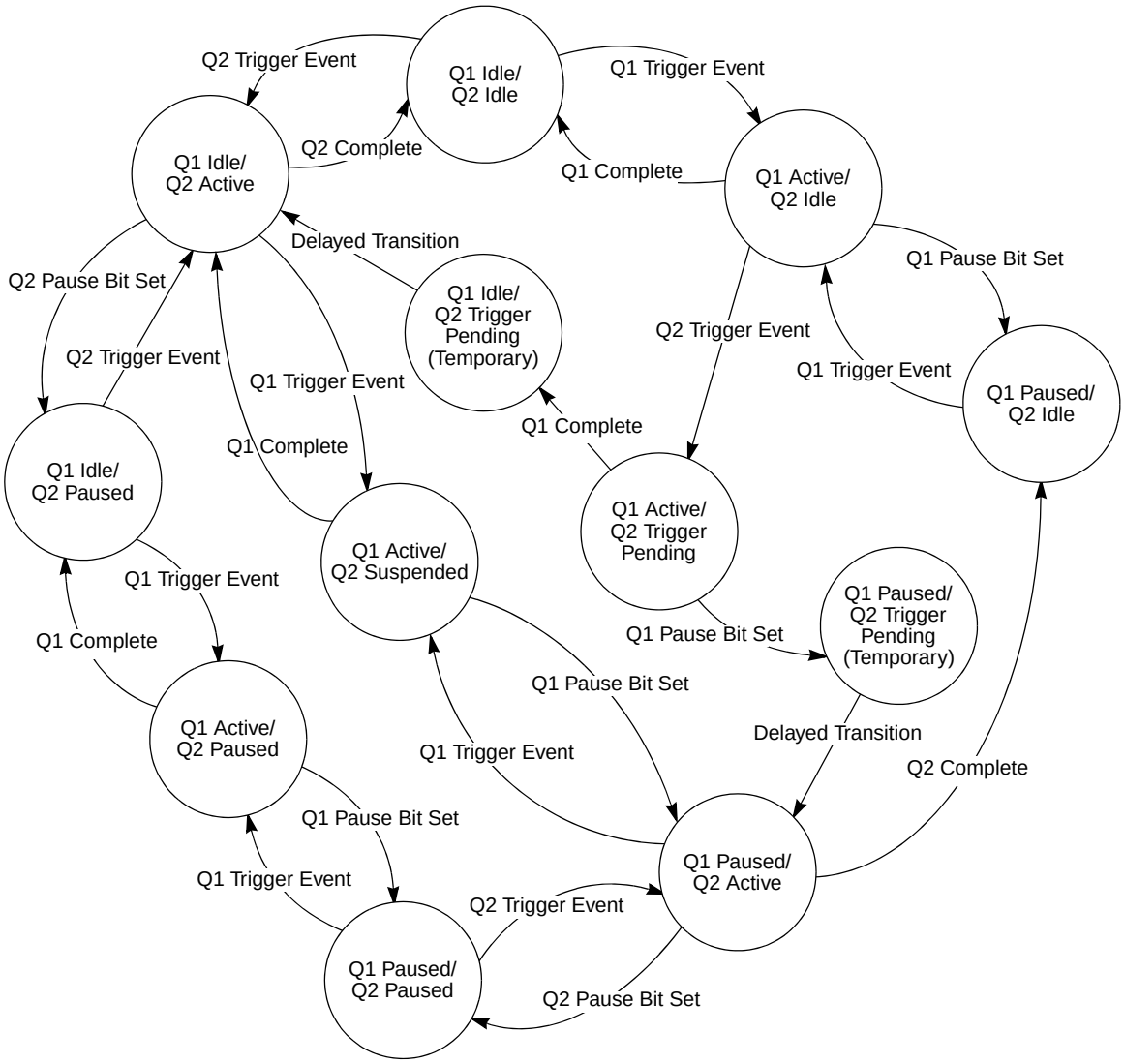


Figure 28-12. Queue Status Transition

28.6.6.2 QADC Status Register 1 (QASR1)

Stop mode resets this register .

	15	14	13	12	11	10	9	8
Field	—		CWPQ15	CWPQ14	CWPQ13	CWPQ12	CWPQ11	CWPQ10
Reset	0011_1111							
R/W:	R							

	7	6	5	4	3	2	1	0
Field	—		CWPQ25	CWPQ24	CWPQ23	CWPQ22	CWPQ21	CWPQ20
Reset	0011_1111							
R/W:	R							
Address	IPSBAR + 0x19_0012, 0x19_0013							

Figure 28-13. QADC Status Register 1 (QASR1)

Table 28-13. QASR1 Field Descriptions

Bit(s)	Name	Description
15–14	—	Reserved, should be cleared.
13–8	CWPQ1	Queue 1 command word pointer. Points to the last queue 1 CCW executed. This is a read-only field with a valid range of 0 to 63; writes have no effect. CWPQ1 always points to the last executed CCW in queue 1, regardless of which queue is active. In contrast to CWP, CPWQ1 is updated when a conversion result is written. When the QADC finishes a conversion in queue 1, both the result register is written and CWPQ1 is updated. When queue 1 operation is terminated after a CCW is read that is pointed to by BQ2, CWP points to BQ2 while CWPQ1 points to the last queue 1 CCW. During stop mode, CWPQ1 is reset to 63, because the control registers and the analog logic are reset. When debug mode is entered, CWPQ1 is not changed; it points to the last executed CCW in queue 1.
7–6	—	Reserved, should be cleared.
5–0	CWPQ	Queue 2 command word pointer. Points to the last queue 2 CCW executed. This is a read-only field with a valid range of 0 to 63; writes have no effect. CWPQ2 always points to the last executed CCW in queue 2, regardless which queue is active. In contrast to CWP, CPWQ2 is updated when a conversion result is written. When the QADC finishes a conversion in queue 2, both the result register is written and CWPQ2 is updated. During stop mode, CWPQ2 is reset to 63 because the control registers and the analog logic are reset. When debug mode is entered, CWPQ2 is not changed; it points to the last executed CCW in queue 2.

28.6.7 Conversion Command Word Table (CCW)

The CCW table is 64 half-word (128 byte) long RAM with 10 bits of each entry implemented. The CCW table is written by the user and is not modified by the QADC. Each CCW requests the conversion of one analog channel to a digital result. The CCW specifies the analog channel number, the input sample time, and whether the queue is to pause after the current CCW. The bits in this register are read anytime (except during stop mode), write anytime (except during stop mode).

	15	10	9	8
Field	—		P	BYP
Reset	0000_00		Unaffected	
R/W:	R		R/W	

	7	6	5	4	3	2	1	0
Field	IST1	IST0	CHAN5	CHAN4	CHAN3	CHAN2	CHAN1	CHAN0
Reset	Undefined							
R/W:	R							
Address	IPSBAR + 0x19_0200, 0x19_027e							

Figure 28-14. Conversion Command Word Table (CCW)

Table 28-14. CCW Field Descriptions

Bit(s)	Name	Description
15–10	—	Reserved, should be cleared.
9	P	Pause. Allows subqueues to be created within queue 1 and queue 2. The QADC performs the conversion specified by the CCW with the pause bit set and then the queue enters the pause state. Another trigger event causes execution to continue from the pause to the next CCW. 1 Enter pause state after execution of current CCW. 0 Do not enter pause state after execution of current CCW. NOTE: The P bit does not cause the queue to pause in software-initiated modes or externally gated modes.
8	BYP	Sample amplifier bypass. Enables the amplifier bypass mode for a conversion and subsequently changes the timing. The initial sample time is eliminated, reducing the potential conversion time by two QCLKs. However, due to internal RC effects, a minimum final sample time of four QCLKs must be allowed. When using this mode, the external circuit should be of low source impedance. Loading effects of the external circuitry need to be considered because the benefits of the sample amplifier are not present. 1 Amplifier bypass mode enabled 0 Amplifier bypass mode disabled NOTE: BYP is maintained for software compatibility but has no functional benefit on this version of the QADC.
7–6	IST	Input sample time. Specifies the length of the sample window. The input sample time can be varied, under software control, to accommodate various input channel source impedances. Longer sample times permit more accurate A/D conversions of signals with higher source impedances. Table 28-15 shows the four selectable input sample times. The programmable sample time can also be used to adjust queue execution time or sampling rate by increasing the time interval between conversions.
5–0	CHAN	Selects the input channel number. The CCW channel field is programmed with the channel number corresponding to the analog input signal to be sampled and converted. The analog input signal channel number assignments and the signal definitions vary depending on whether the QADC multiplexed or non-multiplexed mode is used by the application. As far as queue scanning operations are concerned, there is no distinction between an internally or externally multiplexed analog input. Table 28-16 shows the channel number assignments for non-multiplexed mode. Table 28-17 shows the channel number assignments for multiplexed mode. Programming the channel field to channel 63 denotes the end of the queue. Channels 60 to 62 are special internal channels. When one of the special channels is selected, the sampling amplifier is not used. The value of V_{RL}, V_{RH}, or $(V_{RH}–V_{RL})/2$ is converted directly. Programming any input sample time other than two has no benefit for the special internal channels except to lengthen the overall conversion time.

Table 28-15. Input Sample Times

IST[1:0]	Input Sample Times
00	Input sample time = QCLK period × 2
01	Input sample time = QCLK period × 4
10	Input sample time = QCLK period × 8
11	Input sample time = QCLK period × 16

Table 28-16. Non-Multiplexed Channel Assignments and Signal Designations

Non-Multiplexed Input Signals				Channel Number[1] in CCW CHAN Field	
Port Signal Name	Analog Signal Name	Other Functions	Signal Type	Binary	Decimal
PQB0 PQB1 PQB2 PQB3	AN0 AN1 AN2 AN3	— — — —	Input Input Input Input	000000 000001 000010 000011	0 1 2 3
PQA0 PQA1	AN52 AN53	— —	Input/Output Input/Output	110100 110101	52 53
PQA3 PQA4	AN55 AN56	ETRIG1 ETRIG2	Input/Output Input/Output	110111 111000	55 56
V_{RL} V_{RH} —	Low reference High reference —	— — $(V_{RH}–V_{RL})/2$	Input Input —	111100 111101 111110	60 61 62
—	—	End-of-Queue Code	—	111111	63

1 All channels not listed are reserved or unimplemented and return undefined results.

Table 28-17. Multiplexed Channel Assignments and Signal Designations

Multiplexed Input Signals				Channel Number[1] in CCW CHAN Field	
Port Signal Name	Analog Signal Name	Other Functions	Signal Type	Binary	Decimal
PQB0 PQB1 PQB2 PQB3	ANW ANX ANY ANZ	— — — —	Input Input Input Input	000XX0 000XX1 010XX0 010XX1	0, 2, 4, 6 1, 3, 5, 7 16, 18, 20, 22 17, 19, 21, 23
PQA0 PQA1	— —	MA0 MA1	Output Output	—	52 53
PQA3 PQA4	AN55 AN56	ETRIG1 ETRIG2	Input/Output Input/Output	110111 111000	55 56
V_{RL} V_{RH} —	Low Reference High Reference —	— — $(V_{RH}–V_{RL})/2$	Input Input —	111100 111101 111110	60 61 62
—	—	End-of-Queue Code	—	111111	63

[1] All channels not listed are reserved or unimplemented and return undefined results.

28.6.8 Result Registers

The result word table is a 64 half-word (128 byte) long by 10-bit wide RAM. An entry is written by the QADC after completing an analog conversion specified by the corresponding CCW table entry.

28.6.8.1 Right-Justified Unsigned Result Register (RJURR)

	15 … 10	9 … 8
Field	—	RESULT
Reset	0000_00	Undefined
R/W:	R	R/W

	7 … 0
Field	RESULT
Reset	Undefined
R/W:	R/W
Address	IPSBAR + 0x19_0280, 0x19_02fe

Figure 28-15. Right-Justified Unsigned Result Register (RJURR)

Table 28-18. RJURR Field Descriptions

Bit(s)	Name	Description
15–10	—	Reserved, should be cleared.
9–0	RESULT	The conversion result is unsigned, right-justified data.

28.6.8.2 Left-Justified Signed Result Register (LJSRR)

	15	14 … 8
Field	S	RESULT
Reset	Undefined	
R/W:	R/W	

	7 … 0
Field	RESULT
Reset	Undefined
R/W:	R/W
Address	IPSBAR + 0x19_0300, 0x19_037e

Figure 28-16. Left-Justified Signed Result Register (LJSRR)

Table 28-19. LJSRR Field Descriptions

Bit(s)	Name	Description
15	S	The left justified, signed format corresponds to a half-scale, offset binary, two's complement data format. Conversion values corresponding to 1/2 full scale, 0x0200, or higher are interpreted as positive values and have a sign bit of 0. An unsigned, right justified conversion of 0x0200 would be represented as 0x0000 in this signed register, where the sign = 0 and the result = 0. For an unsigned, right justified conversion of 0x3FF (full range or V_{RH}), the signed equivalent in this register would be 0x7FC0, sign = 0 and result = 0x1FF. For an unsigned, right justified conversion of 0x0000 (V_{RL}), the signed equivalent in this register would be 0x8000, sign = 1 and result = 0x000, a two's complement value representing –512.
14–6	RESULT	The conversion result is signed, left-justified data.
5–0	—	Reserved, should be cleared.

28.6.8.3 Left-Justified Unsigned Result Register (LJURR)

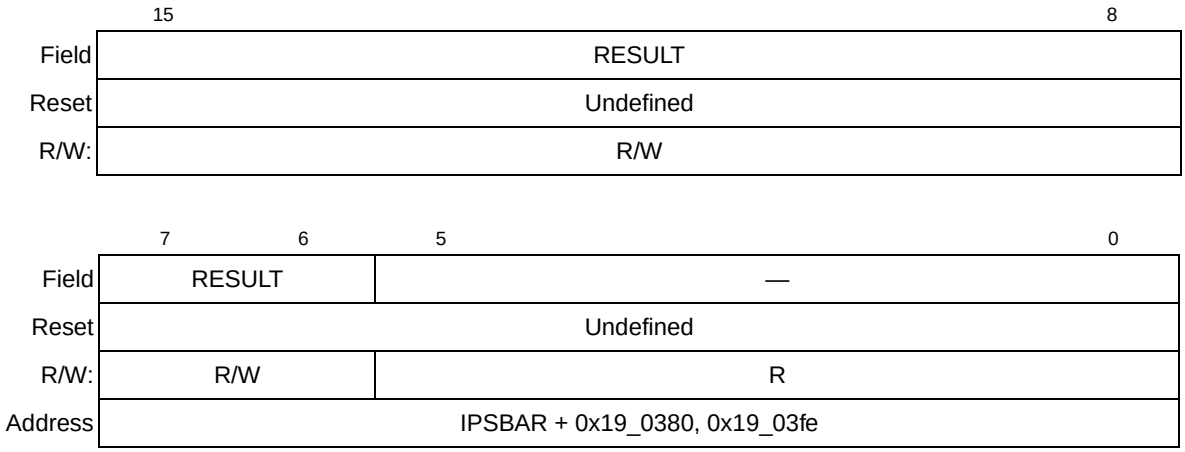


Figure 28-17. Left-Justified Unsigned Result Register (LJURR)

Table 28-20. LJURR Field Descriptions

Bit(s)	Name	Description
15–6	RESULT	The conversion result is unsigned, left-justified data.
5–0	—	Reserved, should be cleared.

28.7 Functional Description

This subsection provides a functional description of the QADC.

28.7.1 Result Coherency

The QADC supports byte and half-word reads and writes across a 16-bit data bus interface. All conversion results are stored in half-word registers, and the QADC does not allow more than one result register to be read at a time. For this reason, the QADC does not guarantee read coherency.

Specifically, this means that while the QADC is operating, the data in the result registers can change from one read to the next. Simply initiating a read of one result register will not prevent another from being updated with a new conversion result.

Thus, to read any given number of result registers coherently, the queue or queues capable of modifying these registers must be inactive. This can be guaranteed by system operating conditions (such as, known completion of a software-initiated queue single-scan or no possibility of an externally triggered/gated queue scan) or by simply disabling the queues (writing MQ1 and/or MQ2 to 0).

28.7.2 External Multiplexing

External multiplexer chips concentrate a number of analog signals onto a few QADC inputs. This is useful for applications that need to convert more analog signals than the QADC converter can normally support. External multiplexing also puts the multiplexed chip closer to the signal source. This minimizes the number of analog signals that need to be shielded due to the proximity of noisy high speed digital signals at the microcontroller chip.

For example, four 4-input multiplexer chips can be put at the connector where the analog signals first arrive on the printed circuit board. As a result, only four analog signals need to be shielded from noise as they approach the microcontroller chip, rather than having to protect 16 analog signals. However, external multiplexer chips may introduce additional noise and errors if not properly utilized. Therefore, it is necessary to maintain low “on” resistance (the impedance of an analog switch when active within a multiplexed chip) and insert a low pass filter (R/C) on the input side of the multiplexed chip.

28.7.2.1 External Multiplexing Operation

The QADC can use from one to four external multiplexer chips to expand the number of analog signals that may be converted. Up to 16 analog channels can be converted through external multiplexer selection. The externally multiplexed channels are automatically selected from the channel field of the CCW, the same as internally multiplexed channels. The QADC is configured for the externally multiplexed mode by setting the MUX bit in control register 0 (QACR0).

Figure 28-18 shows the maximum configuration of four external multiplexer chips connected to the QADC. The external multiplexer chips select one of four analog inputs and connect it to one analog output, which becomes an input to the QADC. The QADC provides two multiplexed address signals, MA[1:0], to select one of four inputs. These inputs are connected to all four external multiplexer chips. The analog output of the four multiplexer chips are each connected to separate QADC inputs (ANW, ANX, ANY, and ANZ) as shown in Figure 28-18

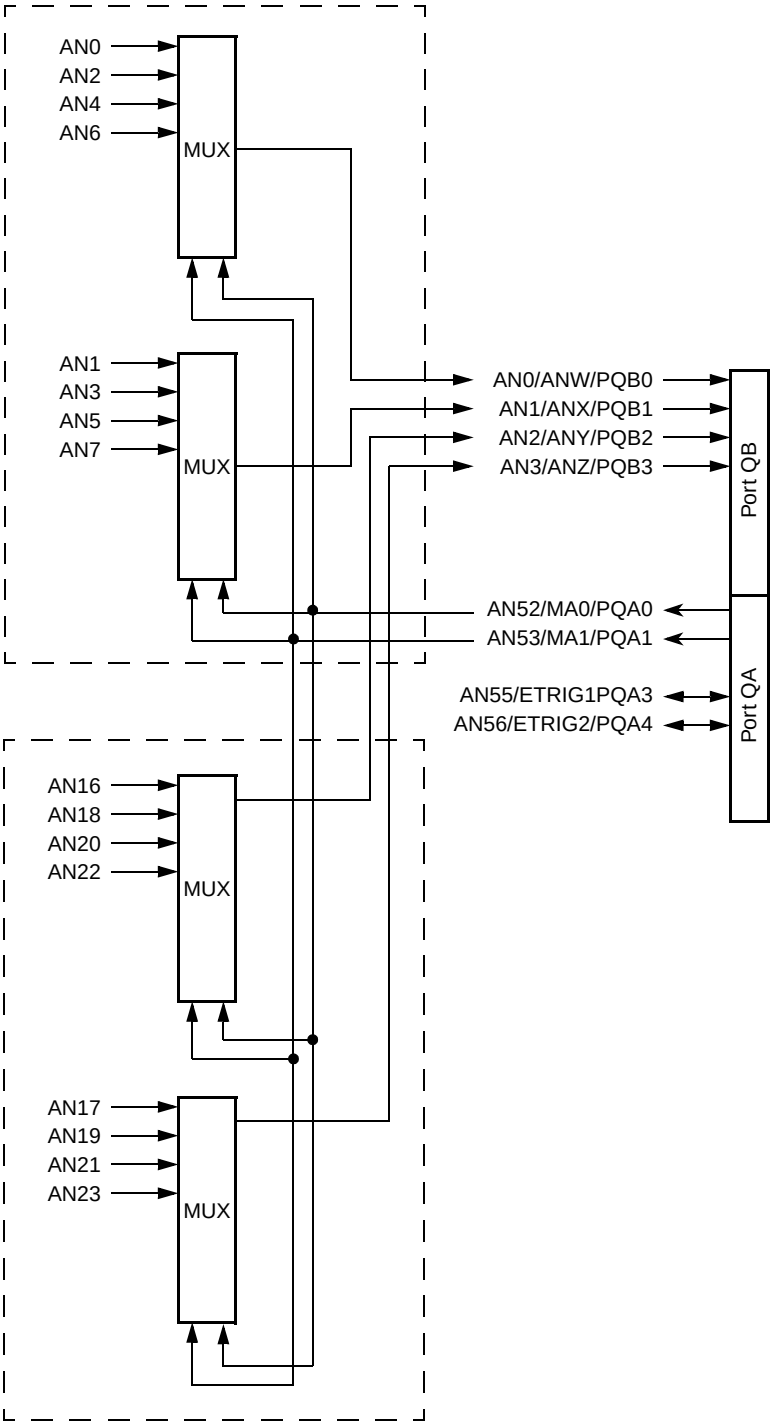


Figure 28-18. External Multiplexing Configuration

When externally multiplexed mode is selected, the QADC automatically drives the MA output signals from the channel number in each CCW. The QADC also converts the proper input channel (ANW, ANX, ANY, and ANZ) by interpreting the CCW channel number. As a result, up to 16 externally multiplexed channels appear to the conversion queues as directly connected signals. User software simply puts the channel number of externally multiplexed channels into CCWs.

Figure 28-18 shows that the two MA signals may also be analog input signals. When external multiplexing is selected, none of the MA signals can be used for analog or digital inputs. They become multiplexed address outputs and are unaffected by DDRQA[1:0].

28.7.2.2 Module Version Options

The number of available analog channels varies, depending on whether external multiplexing is used. A maximum of eight analog channels are supported by the internal multiplexing circuitry of the converter. Table 28-21 shows the total number of analog input channels supported with 0 to 4 external multiplexer chips.

Table 28-21. Analog Input Channels

Number of Analog Input Channels Available Directly Connected + External Multiplexed = Total Channels[1, 2]				
No External Mux	**One External Mux**	**Two External Muxes**	**Three External Muxes**	**Four External Muxes**
8	5 + 4 = 9	4 + 8 = 12	3 + 12 = 15	2 + 16 = 18

1 The external trigger inputs are not shared with two analog input signals.

2 When external multiplexing is used, two input channels are configured as multiplexed address outputs, and for each external multiplexer chip, one input channel is a multiplexed analog input.

28.7.3 Analog Subsystem

This section describes the QADC analog subsystem, which includes the front-end analog multiplexer and analog-to-digital converter.

28.7.3.1 Analog-to-Digital Converter Operation

The analog subsystem consists of the path from the input signals to the A/D converter block. Signals from the queue control logic are fed to the multiplexer and state machine. The end-of-conversion (EOC) signal and the successive approximation register (SAR) reflect the result of the conversion. Figure 28-19 shows a block diagram of the QADC analog subsystem.

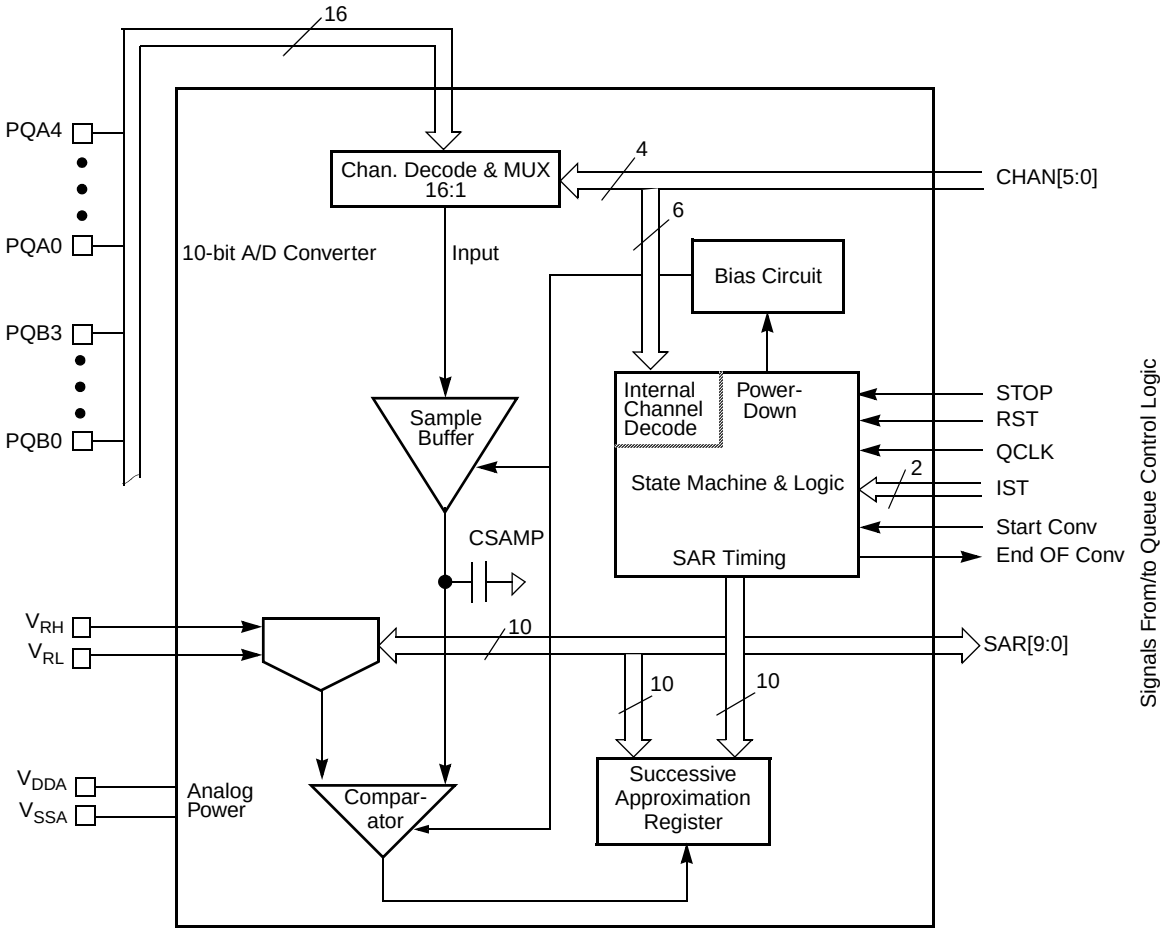


Figure 28-19. QADC Analog Subsystem Block Diagram

28.7.3.2 Conversion Cycle Times

Total conversion time is made up of initial sample time, final sample time, and resolution time. Initial sample time refers to the time during which the selected input channel is coupled through the sample buffer amplifier to the sample capacitor. The sample buffer is used to quickly reproduce its input signal on the sample capacitor and minimize charge sharing errors. During the final sampling period the amplifier is bypassed, and the multiplexer input charges the sample capacitor array directly for improved accuracy. During the resolution period, the voltage in the sample capacitor is converted to a digital value and stored in the SAR as shown in Figure 28-20.

Initial sample time is fixed at two QCLK cycles. Final sample time can be 2, 4, 8, or 16 QCLK cycles, depending on the value of the IST field in the CCW. Resolution time is 10 QCLK cycles.

A conversion requires a minimum of 14 QCLK cycles (7 µs with a 2.0-MHz QCLK). If the maximum final sample time period of 16 QCLKs is selected, the total conversion time is 28 QCLKs or 14 µs (with a 2.0-MHz QCLK).

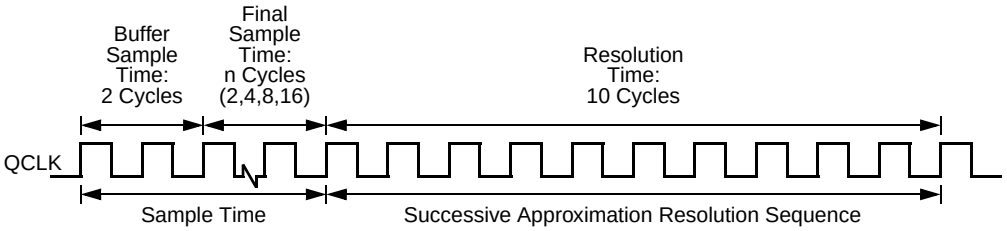


Figure 28-20. Conversion Timing

If the amplifier bypass mode is enabled for a conversion by setting the amplifier bypass (BYP) field in the CCW, the timing changes to that shown in Figure 28-21. See Section 28.6.7, "Conversion Command Word Table (CCW) for more information on the BYP field. The initial sample time is eliminated, reducing the potential conversion time by two QCLKs. When using the bypass mode, the external circuit should be of low source impedance (typically less than 10 kΩ). Also, the loading effects on the external circuitry of the QADC need to be considered, because the benefits of the sample amplifier are not present.

NOTE

Because of internal RC time constants, use of a two QCLK sample time in bypass mode will cause serious errors when operating the QADC at high frequencies.

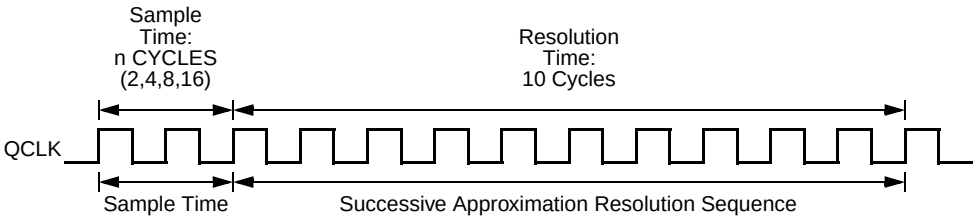


Figure 28-21. Bypass Mode Conversion Timing

28.7.3.3 Channel Decode and Multiplexer

The internal multiplexer selects one of the eight analog input signals for conversion. The selected input is connected to the sample buffer amplifier or to the sample capacitor. The multiplexer also includes positive and negative stress protection circuitry, which prevents deselected channels from affecting the selected channel when current is injected into the deselected channels.

28.7.3.4 Sample Buffer

The sample buffer is used to raise the effective input impedance of the A/D converter, so that external factors (higher bandwidth or higher impedance) are less critical to accuracy. The input voltage is buffered onto the sample capacitor to reduce crosstalk between channels.

28.7.3.5 Comparator

The comparator output feeds into the SAR, which accumulates the A/D conversion result sequentially, beginning with the MSB.

28.7.3.6 Bias

The bias circuit is controlled by the STOP signal to power-up and power-down all the analog circuits.

28.7.3.7 Successive Approximation Register (SAR)

The input of the SAR is connected to the comparator output. The SAR sequentially receives the conversion value one bit at a time, starting with the MSB. After accumulating the 10 bits of the conversion result, the SAR data is transferred to the appropriate result location, where it may be read by user software.

28.7.3.8 State Machine

The state machine generates all timing to perform an A/D conversion. An internal start-conversion signal indicates to the A/D converter that the desired channel has been sent to the MUX. CCW[IST[1:0]] denotes the desired sample time. CCW[BYP] determines whether to bypass the sample amplifier. Once the end of conversion has been reached a signal is sent to the queue control logic indicating that a result is available for storage in the result RAM.

28.8 Digital Control Subsystem

The digital control subsystem includes the control logic to sequence the conversion activity, the system clock and periodic/interval timer, control and status registers, the conversion command word table RAM, and the result word table RAM.

The central element for control of QADC conversions is the 64-entry conversion command word (CCW) table. Each CCW specifies the conversion of one input channel. Depending on the application, one or two queues can be established in the CCW table. A queue is a scan sequence of one or more input channels. By using a pause mechanism, subqueues can be created in the two queues. Each queue can be operated using one of several different scan modes. The scan modes for queue 1 and queue 2 are programmed in control registers QACR1 and QACR2. Once a queue has been started by a trigger event (any of the ways to cause the QADC to begin executing the CCWs in a queue or subqueue), the QADC performs a sequence of conversions and places the results in the result word table.

28.8.1 Queue Priority Timing Examples

This subsection describes the QADC priority scheme when trigger events on two queues overlap or conflict.

28.8.1.1 Queue Priority

Queue 1 has priority over queue 2 execution. These cases show the conditions under which queue 1 asserts its priority:

- When a queue is not active, a trigger event for queue 1 or queue 2 causes the corresponding queue execution to begin.
- When queue 1 is active and a trigger event occurs for queue 2, queue 2 cannot begin execution until queue 1 reaches completion or the paused state. The status register records the trigger event by reporting the queue 2 status as trigger pending. Additional trigger events for queue 2, which occur before execution can begin, are flagged as trigger overruns.
- When queue 2 is active and a trigger event occurs for queue 1, the current queue 2 conversion is aborted. The status register reports the queue 2 status as suspended. Any trigger events occurring for queue 2 while it is suspended are flagged as trigger overruns. Once queue 1 reaches the

completion or the paused state, queue 2 begins executing again. The programming of the RESUME bit in QACR2 determines which CCW is executed in queue 2.

- When simultaneous trigger events occur for queue 1 and queue 2, queue 1 begins execution and the queue 2 status is changed to trigger pending.
- When subqueues are paused

The pause feature can be used to divide queue 1 and/or queue 2 into multiple subqueues. A subqueue is defined by setting the pause bit in the last CCW of the subqueue.

Figure 28-22 shows the CCW format and an example of using pause to create subqueues. Queue 1 is shown with four CCWs in each subqueue and queue 2 has two CCWs in each subqueue.

The operating mode selected for queue 1 determines what type of trigger event causes the execution of each of the subqueues within queue 1. Similarly, the operating mode for queue 2 determines the type of trigger event required to execute each of the subqueues within queue 2.

For example, when the external trigger rising edge continuous-scan mode is selected for queue 1, and there are six subqueues within queue 1, a separate rising edge is required on the external trigger signal after every pause to begin the execution of each subqueue (refer to Figure 28-22).

The choice of single-scan or continuous-scan applies to the full queue, and is not applied to each subqueue. Once a subqueue is initiated, each CCW is executed sequentially until the last CCW in the subqueue is executed and the pause state is entered. Execution can only continue with the next CCW, which is the beginning of the next subqueue. A subqueue cannot be executed a second time before the overall queue execution has been completed.

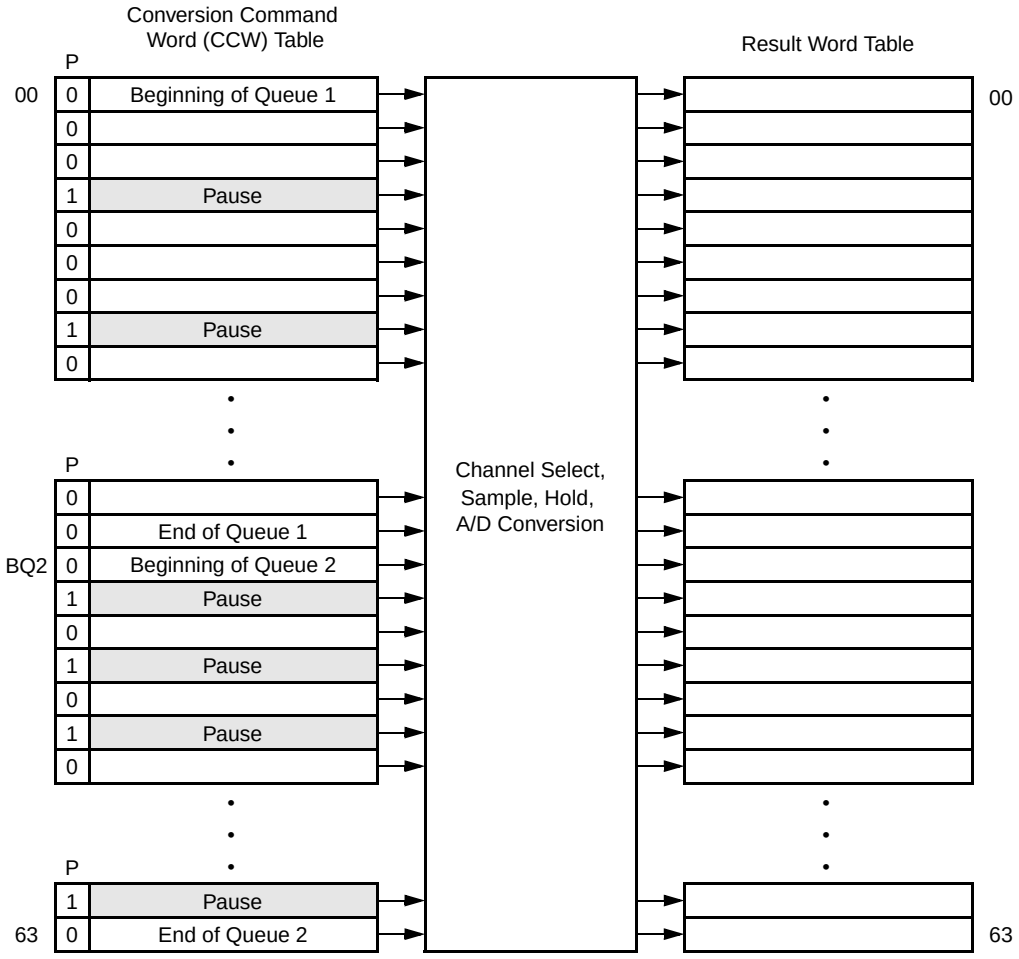


Figure 28-22. QADC Queue Operation with Pause

Trigger events which occur during the execution of a subqueue are ignored, but the trigger overrun flag is set. When a continuous-scan mode is selected, a trigger event occurring after the completion of the last subqueue (after the queue completion flag is set), causes the execution to continue with the first subqueue, starting with the first CCW in the queue.

When the QADC encounters a CCW with the pause bit set, the queue enters the paused state after completing the conversion specified in the CCW with the pause bit. The pause flag is set in QASR0, and a pause interrupt may be requested. The status of the queue is shown to be paused, indicating completion of a subqueue. The QADC then waits for another trigger event to again begin execution of the next subqueue.

28.8.1.2 Queue Priority Schemes

Because there are two conversion command queues and only one A/D converter, a priority scheme determines which conversion occurs. Each queue has a variety of trigger events that are intended to initiate conversions, and they can occur asynchronously in relation to each other and other conversions in progress. For example, a queue can be idle awaiting a trigger event; a trigger event can have occurred, but the first conversion has not started; a conversion can be in progress; a pause condition can exist awaiting another trigger event to continue the queue; and so on.

The following paragraphs and figures outline the prioritizing criteria used to determine which conversion occurs in each overlap situation.

NOTE

Each situation in Figure 28-23 through Figure 28-33 is labeled S1 through S19. In each diagram, time is shown increasing from left to right. The execution of queue 1 and queue 2 (Q1 and Q2) is shown as a string of rectangles representing the execution time of each CCW in the queue. In most of the situations, there are four CCWs (labeled C1 to C4) in both queue 1 and queue 2. In some of the situations, CCW C2 is presumed to have the pause bit set, to show the similarities of pause and end-of-queue as terminations of queue execution.

Trigger events are described in Table 28-22.

Table 28-22. Trigger Events

Trigger	Events
T1	Events that trigger queue 1 execution (external trigger, software-initiated single-scan enable bit, or completion of the previous continuous loop)
T2	Events that trigger queue 2 execution (external trigger, software-initiated single-scan enable bit, timer period/interval expired, or completion of the previous continuous loop)

When a trigger event causes a CCW execution in progress to be aborted, the aborted conversion is shown as a ragged end of a shortened CCW rectangle.

The situation diagrams also show when key status bits are set.
Table 28-23 describes the status bits.

Table 28-23. Status Bits

Bit	Function
CF flag	Set when the end of the queue is reached
PF flag	Set when a queue completes execution up through a pause bit
Trigger overrun error (TOR)	Set when a new trigger event occurs before the queue is finished servicing the previous trigger event

Below the queue execution flows are three sets of blocks that show the status information that is made available to the user. The first two rows of status blocks show the condition of each queue as:

- Idle
- Active
- Pause
- Suspended (queue 2 only)
- Trigger pending

The third row of status blocks shows the 4-bit QS status register field that encodes the condition of the two queues. Two transition status cases, QS = 0011 and QS = 0111, are not shown because they exist only very briefly between stable status conditions.

The first three examples in Figure 28-23 through Figure 28-25 (S1, S2, and S3) show what happens when a new trigger event is recognized before the queue has completed servicing the previous trigger event on the same queue.

In situation S1 (Figure 28-23), one trigger event is being recognized on each queue while that queue is still working on the previously recognized trigger event. The trigger overrun error status bit is set, and the premature trigger event is otherwise ignored. A trigger event that occurs before the servicing of the previous trigger event is through does not disturb the queue execution in progress.

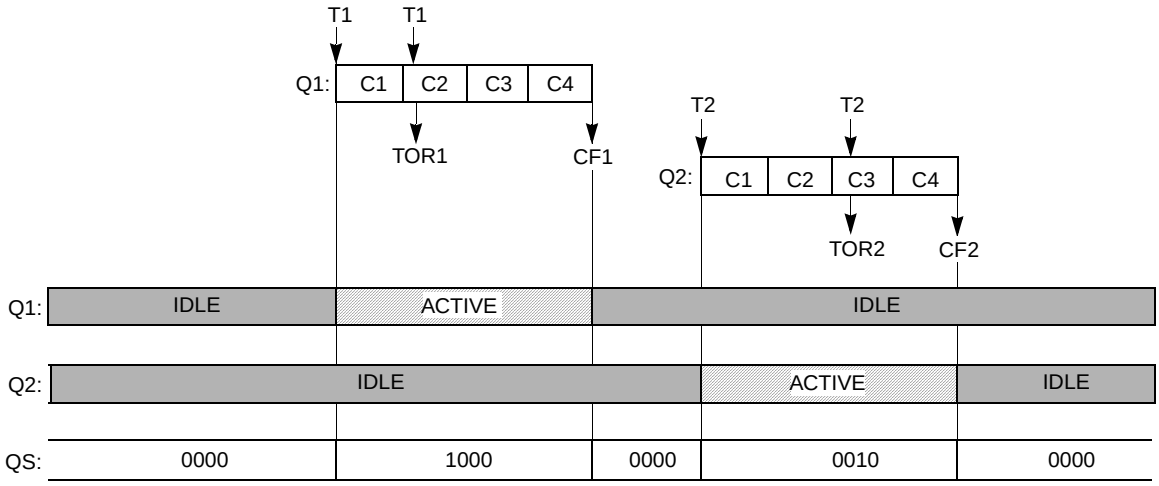


Figure 28-23. CCW Priority Situation 1

In situation S2 (Figure 28-24), more than one trigger event is recognized before servicing of a previous trigger event is complete. The trigger overrun bit is again set, but the additional trigger events are otherwise ignored. After the queue is complete, the first newly detected trigger event causes queue execution to begin again. When the trigger event rate is high, a new trigger event can be seen very soon after completion of the previous queue, leaving little time to retrieve the previous results. Also, when trigger events are occurring at a high rate for queue 1, the lower priority queue 2 channels may not get serviced at all.

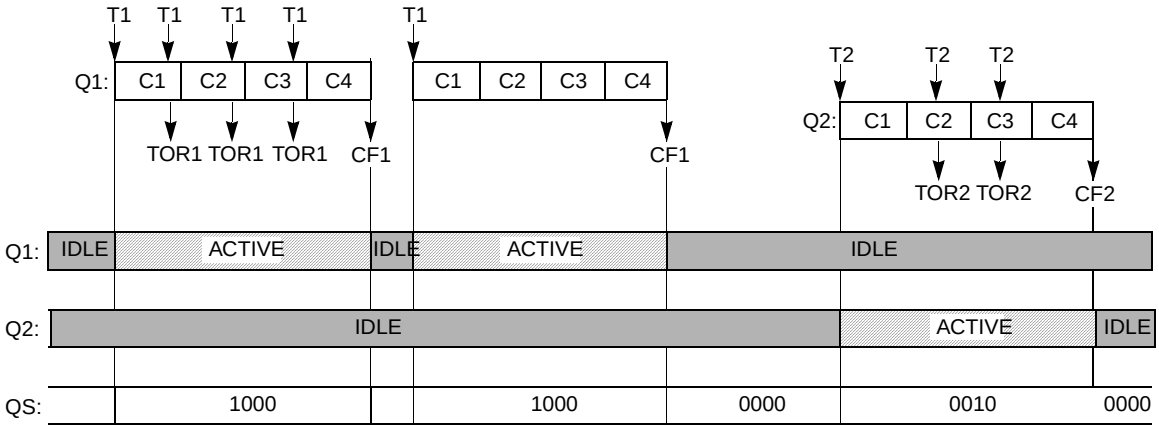


Figure 28-24. CCW Priority Situation 2

Situation S3 (Figure 28-25) shows that when the pause feature is used, the trigger overrun error status bit is set the same way and that queue execution continues unchanged.

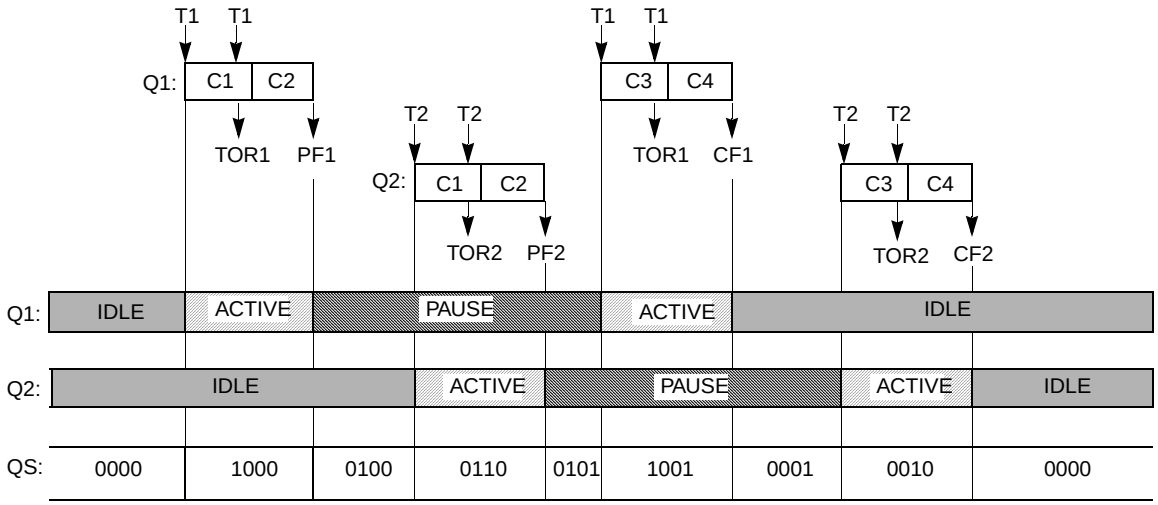


Figure 28-25. CCW Priority Situation 3

The next two situations consider trigger events that occur for the lower priority queue 2, while queue 1 is actively being serviced.

Situation S4 (Figure 28-26) shows that a queue 2 trigger event is recognized while queue 1 is active is saved, and as soon as queue 1 is finished, queue 2 servicing begins.

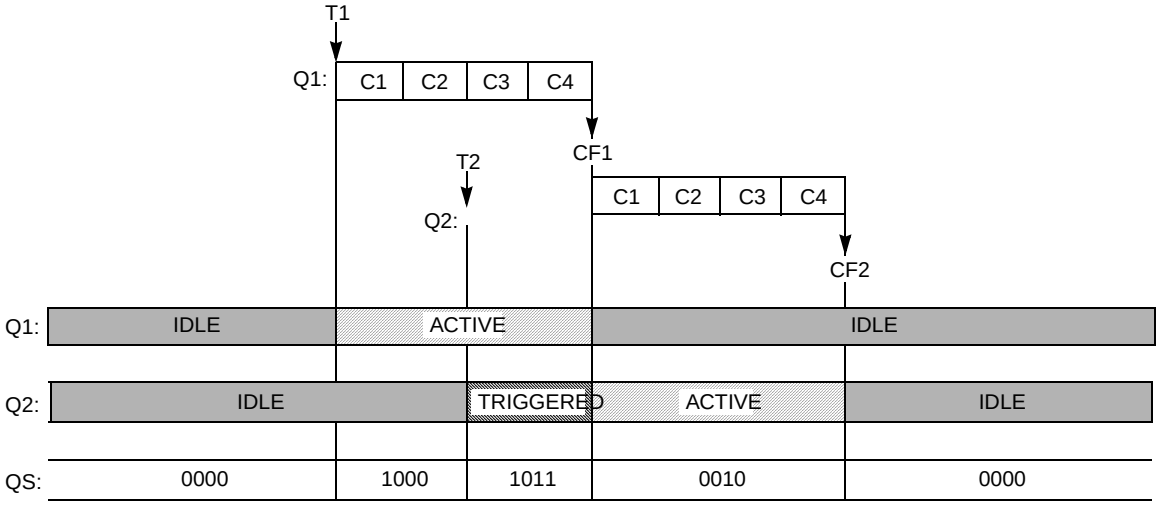


Figure 28-26. CCW Priority Situation 4

Situation S5 (Figure 28-27) shows that when multiple queue 2 trigger events are detected while queue 1 is busy, the trigger overrun error bit is set, but queue 1 execution is not disturbed. Situation S5 also shows that the effect of queue 2 trigger events during queue 1 execution is the same when the pause feature is used for either queue.

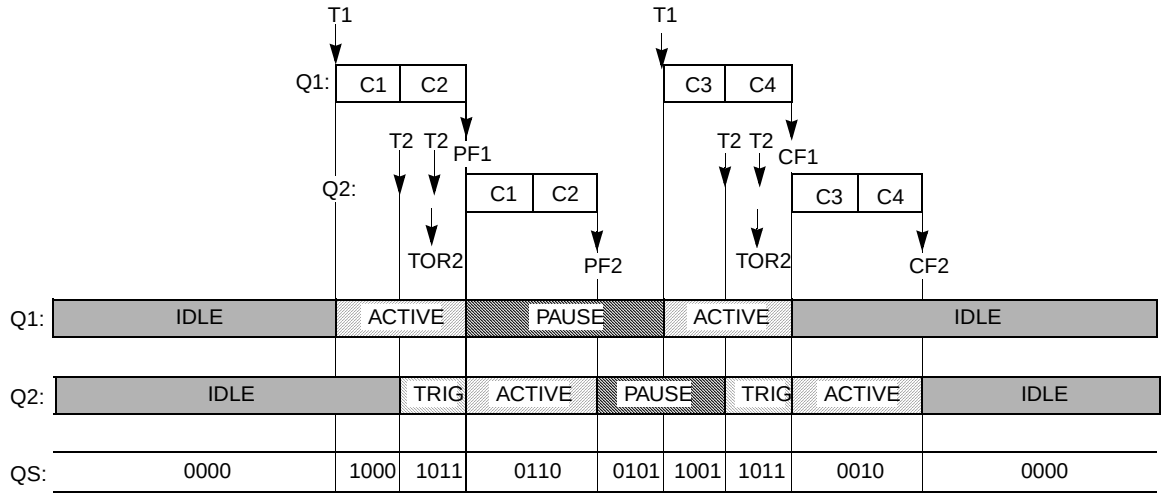


Figure 28-27. CCW Priority Situation 5

The remaining situations, S6 through S11, show the impact of a queue 1 trigger event occurring during queue 2 execution. Because queue 1 has higher priority, the conversion taking place in queue 2 is aborted so that there is no variable latency time in responding to queue 1 trigger events.

In situation 6 (Figure 28-28), the conversion initiated by the second CCW in queue 2 is aborted just before the conversion is complete, so that queue 1 execution can begin. Queue 2 is considered suspended. After queue 1 is finished, queue 2 starts over with the first CCW, when the RESUME control bit is set to 0. Situation S7 (Figure 28-29) shows that when pause operation is not used with queue 2, queue 2 suspension works the same way.

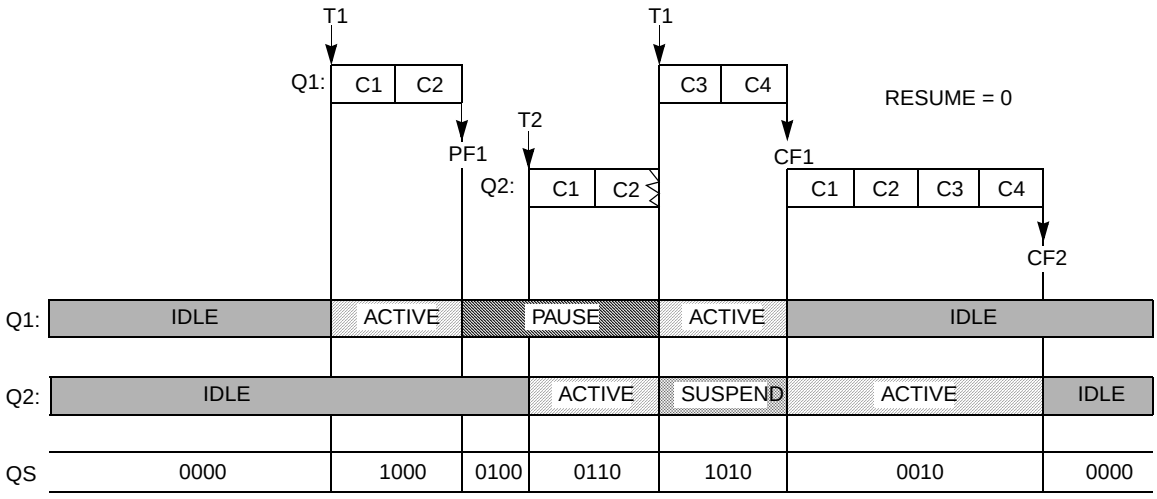


Figure 28-28. CCW Priority Situation 6

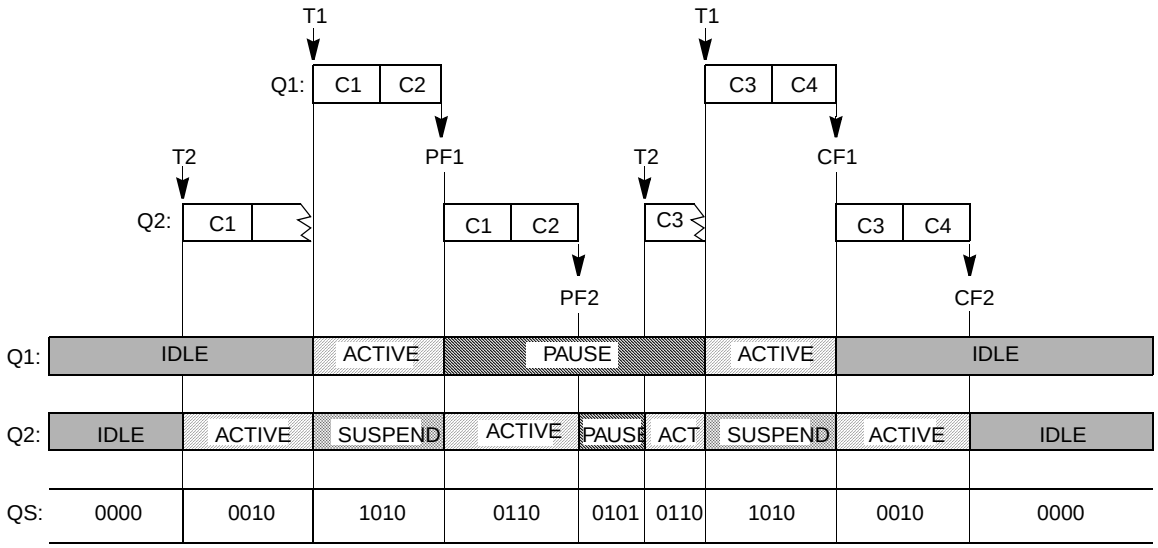


Figure 28-29. CCW Priority Situation 7

Situations S8 and S9 (Figure 28-30 and Figure 28-31) repeat the same two situations with the RESUME bit set to a 1. When the RESUME bit is set, following suspension, queue 2 resumes execution with the aborted CCW, not the first CCW, in the queue.

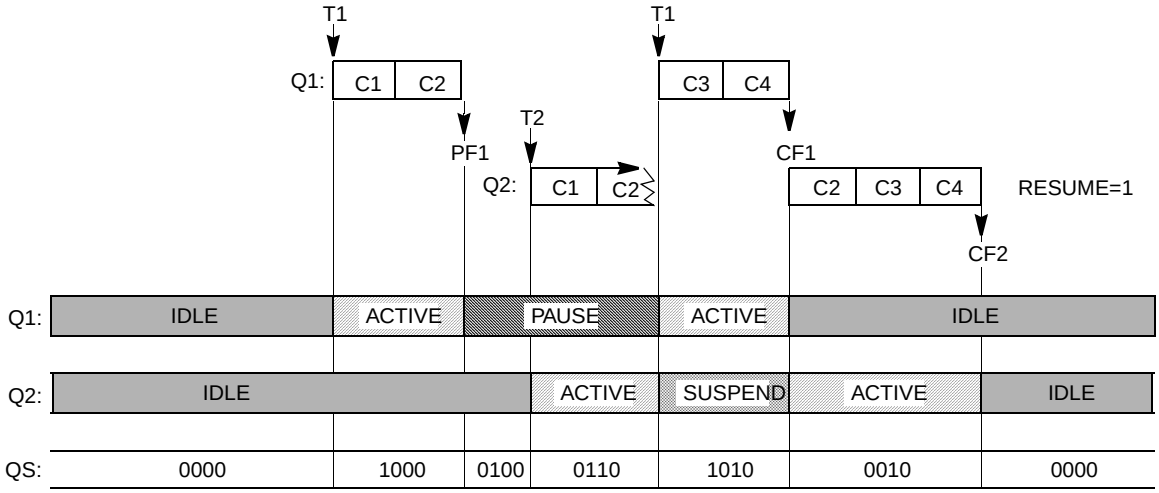


Figure 28-30. CCW Priority Situation 8

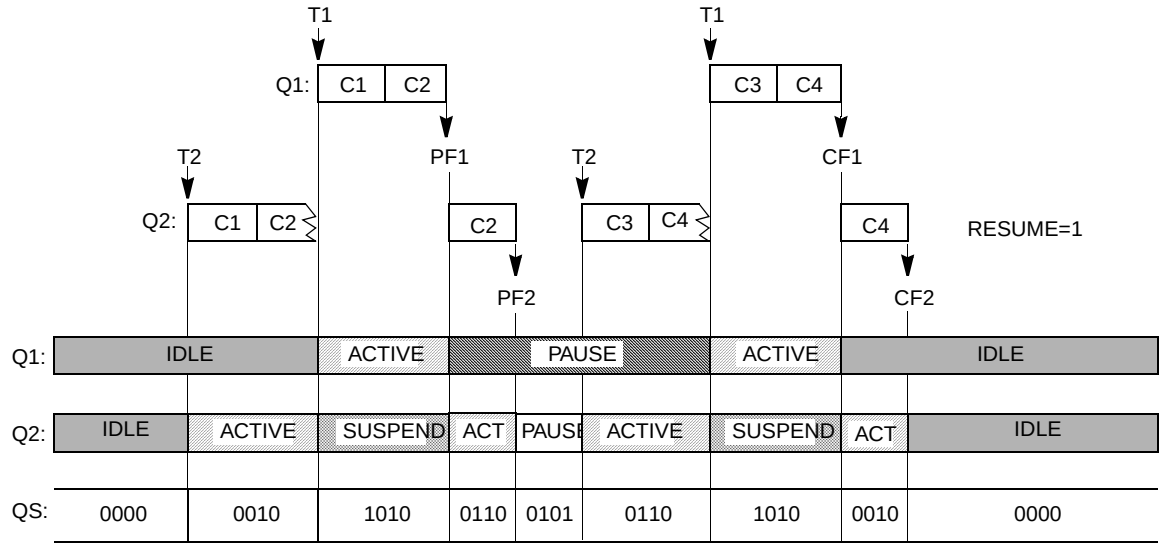


Figure 28-31. CCW Priority Situation 9

Situations S10 and S11 (Figure 28-32 and Figure 28-33) show that when an additional trigger event is detected for queue 2 while the queue is suspended, the trigger overrun error bit is set, the same as if queue 2 were being executed when a new trigger event occurs. Trigger overrun on queue 2 thus allows the user to know that queue 1 is taking up so much QADC time that queue 2 trigger events are being lost.

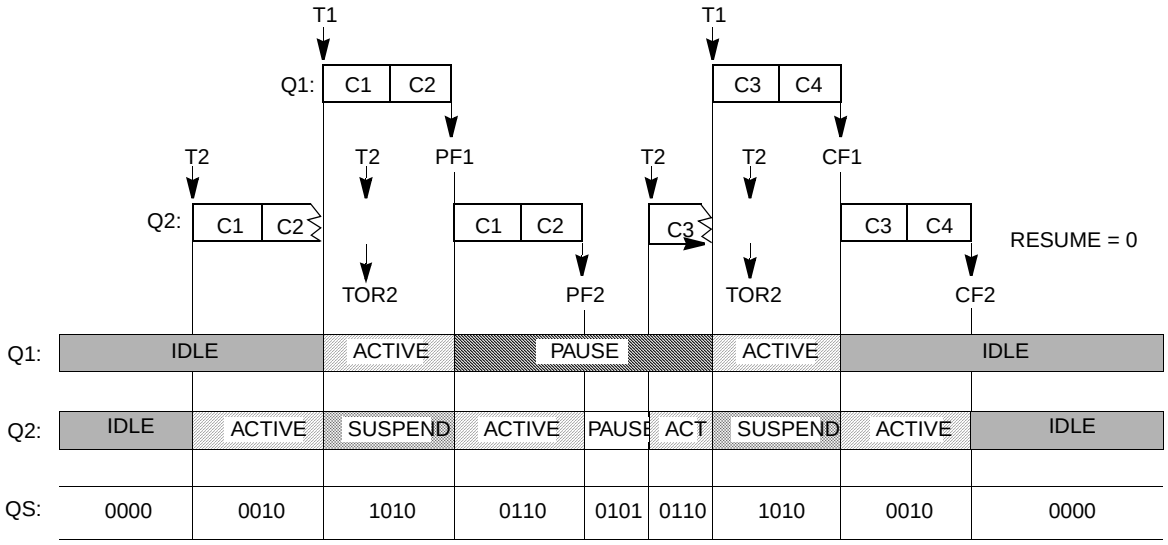


Figure 28-32. CCW Priority Situation 10

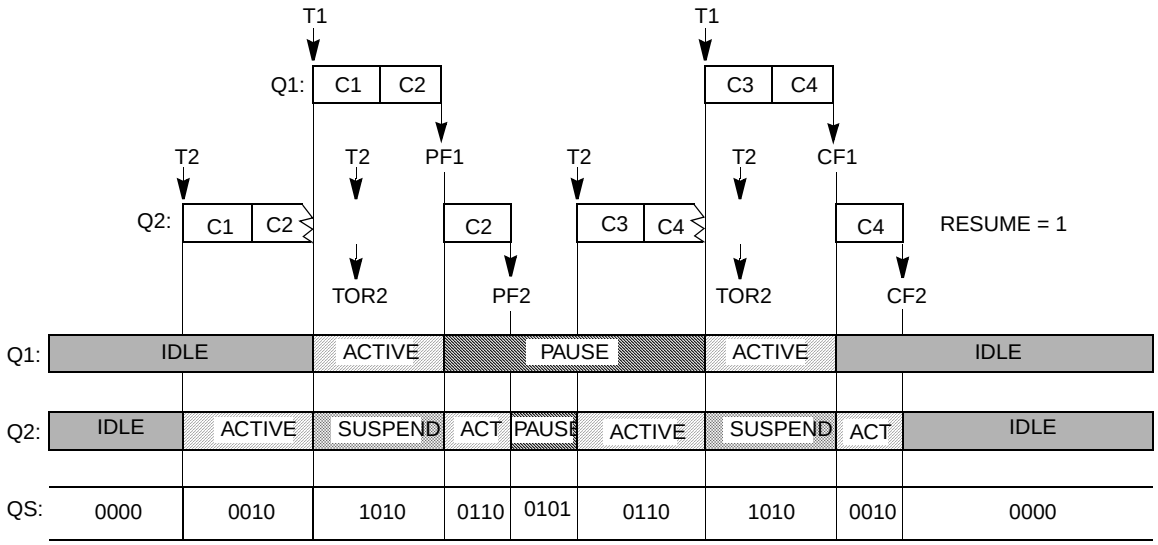


Figure 28-33. CCW Priority Situation 11

The previous situations cover normal overlap conditions that arise with asynchronous trigger events on the two queues. An additional conflict to consider is that the freeze condition can arise while the QADC is actively executing CCWs. The conventional use for the debug mode is for software/hardware debugging. When the CPU enters background debug mode, peripheral modules can cease operation. When freeze is detected, the QADC completes the conversion in progress, unlike the abort that occurs when queue 1 suspends queue 2. After the freeze condition is removed, the QADC continues queue execution with the next CCW in sequence.

Trigger events that occur during freeze are not captured. When a trigger event is pending for queue 2 before freeze begins, that trigger event is remembered when the freeze is passed. Similarly, when freeze occurs while queue 2 is suspended, after freeze, queue 2 resumes execution as soon as queue 1 is finished.

Situations 12 through 19 (Figure 28-34 to Figure 28-41) show examples of all of the freeze situations.

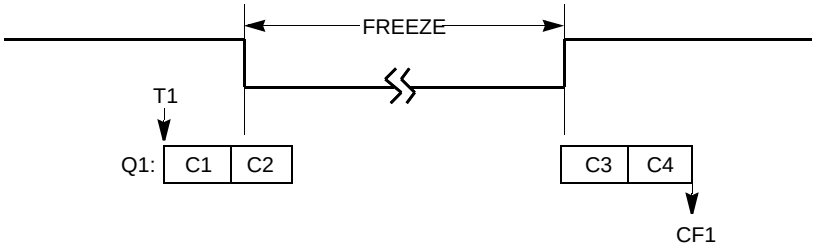


Figure 28-34. CCW Freeze Situation 12

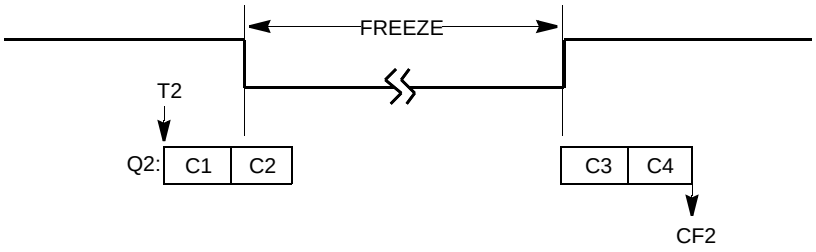


Figure 28-35. CCW Freeze Situation 13

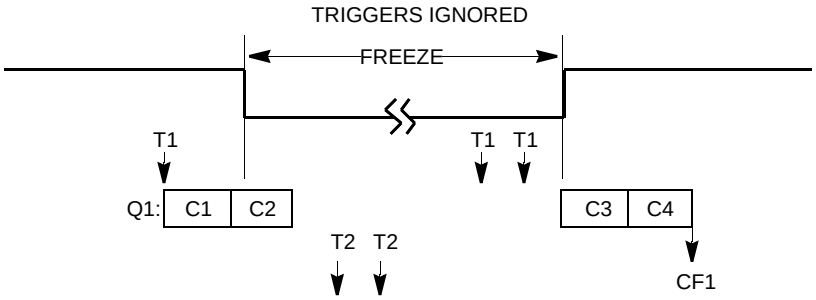


Figure 28-36. CCW Freeze Situation 14

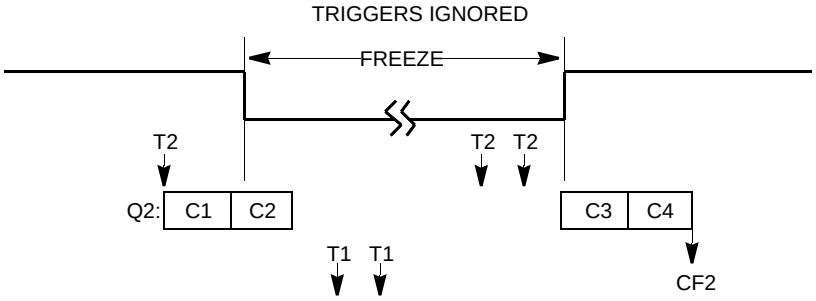


Figure 28-37. . CCW Freeze Situation 15

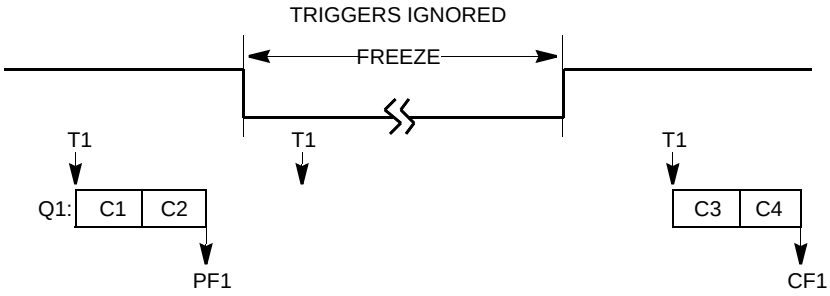


Figure 28-38. CCW Freeze Situation 16

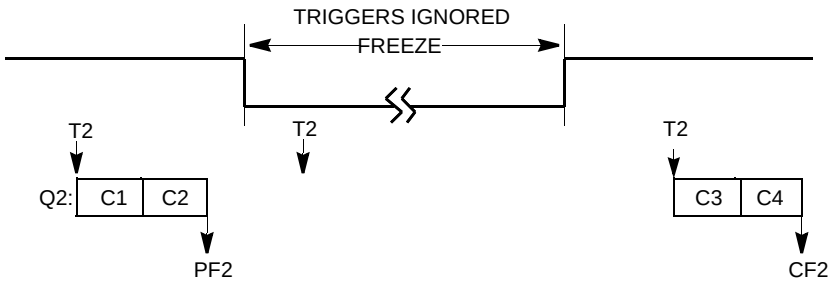


Figure 28-39. CCW Freeze Situation 17

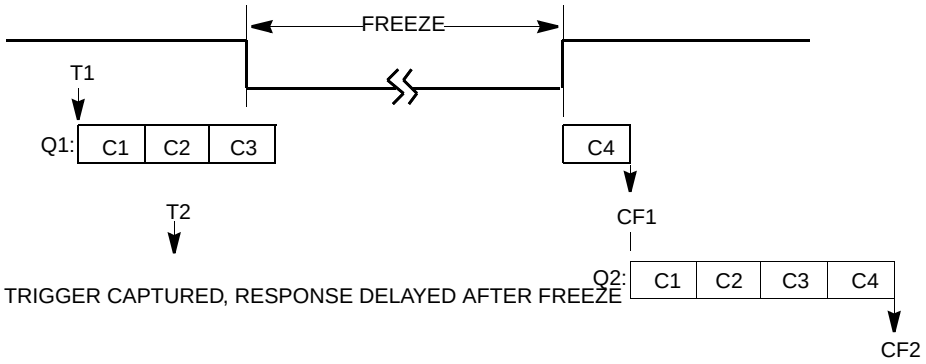


Figure 28-40. CCW Freeze Situation 18

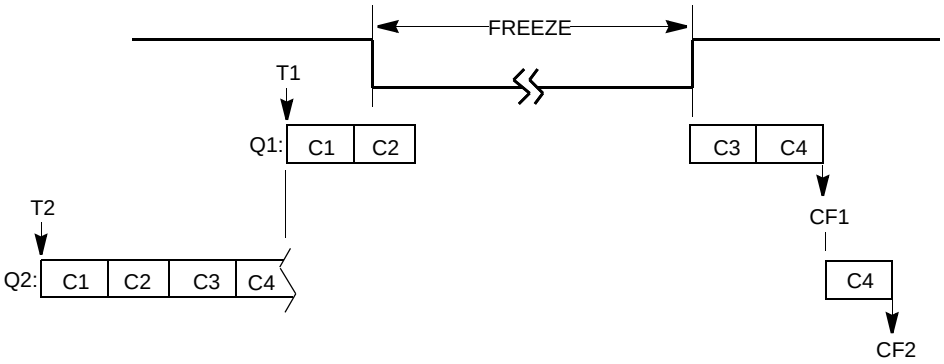


Figure 28-41. CCW Freeze Situation 19

28.8.2 Boundary Conditions

The queue operation boundary conditions are:

- The first CCW in a queue specifies channel 63, the end-of-queue (EOQ) code. The queue becomes active and the first CCW is read. The end-of-queue is recognized, the completion flag is set, and the queue becomes idle. A conversion is not performed.
- BQ2 (beginning of queue 2) is set at the end of the CCW table (63) and a trigger event occurs on queue 2. The end-of-queue condition is recognized, a conversion is performed, the completion flag is set, and the queue becomes idle.
- BQ2 is set to CCW0 and a trigger event occurs on queue 1. After reading CCW0, the end-of-queue condition is recognized, the completion flag is set, and the queue becomes idle. A conversion is not performed.

- BQ2 (beginning of queue 2) is set beyond the end of the CCW table (64–127) and a trigger event occurs on queue 2. The end-of-queue condition is recognized immediately, the completion flag is set, and the queue becomes idle. A conversion is not performed.

NOTE

Multiple end-of-queue conditions may be recognized simultaneously, although there is no change in QADC behavior. For example, if BQ2 is set to CCW0, CCW0 contains the EOQ code, and a trigger event occurs on queue 1, the QADC reads CCW0 and detects both end-of-queue conditions. The completion flag is set and queue 1 becomes idle.

Boundary conditions also exist for combinations of pause and end-of-queue. One case is when a pause bit is in one CCW and an end-of-queue condition is in the next CCW. The conversion specified by the CCW with the pause bit set completes normally. The pause flag is set. However, because the end-of-queue condition is recognized, the completion flag is also set and the queue status becomes idle, not paused. Examples of this situation include:

- The pause bit is set in CCW5 and the channel 63 (EOQ) code is in CCW6.
- The pause is in CCW63.
- During queue 1 operation, the pause bit is set in CCW20 and BQ2 points to CCW21.

Another pause and end-of-queue boundary condition occurs when the pause and an end-of-queue condition occur in the same CCW. Both the pause and end-of-queue conditions are recognized simultaneously. The end-of-queue condition has precedence so a conversion is not performed for the CCW and the pause flag is not set. The QADC sets the completion flag and the queue status becomes idle. Examples of this situation are:

- The pause bit is set in CCW10 and EOQ is programmed into CCW10.
- During queue 1 operation, the pause bit set in CCW32, which is also BQ2.

28.8.3 Scan Modes

The QADC queuing mechanism allows application software to utilize different requirements for automatically scanning input channels.

In single-scan mode, a single pass through a sequence of conversions defined by a queue is performed. In continuous-scan mode, multiple passes through a sequence of conversions defined by a queue are executed.

The possible modes are:

- Disabled mode and reserved mode
- Software-initiated single-scan mode
- Externally triggered single-scan mode
- Externally gated single-scan mode
- Interval timer single-scan mode
- Software-initiated continuous-scan mode
- Externally triggered continuous-scan mode
- Externally gated continuous-scan mode
- Periodic timer continuous-scan mode

The following paragraphs describe single-scan and continuous-scan operations.

28.8.4 Disabled Mode

When disabled mode is selected, the queue is not active. Trigger events cannot initiate queue execution. When both queue 1 and queue 2 are disabled, there is no possibility of encountering wait states when accessing CCW table and result RAM. When both queues are disabled, it is safe to change the QCLK prescaler values.

28.8.5 Reserved Mode

Reserved mode is available for future mode definitions. When reserved mode is selected, the queue is not active. The behavior is the same as disabled mode.

28.8.6 Single-Scan Modes

A single-scan queue operating mode is used to execute a single pass through a sequence of conversions defined by a queue. By programming the MQ1 field in QACR1 or the MQ2 field in QACR2, these modes can be selected:

- Software-initiated single-scan mode
- Externally triggered single-scan mode
- Externally gated single-scan mode
- Interval timer single-scan mode

NOTE

Queue 2 cannot be programmed for externally gated single-scan mode.

In all single-scan queue operating modes, queue execution is enabled by writing the single-scan enable bit to a 1 in the queue's control register. The single-scan enable bits, SSE1 and SSE2, are provided for queue 1 and queue 2, respectively.

Until a queue's single-scan enable bit is set, any trigger events for that queue are ignored. The single-scan enable bit may be set to a 1 during the same write cycle that selects the single-scan queue operating mode. The single-scan enable bit can be written only to 1, but will always read 0. Once set, writing the single-scan enable bit to 0 has no effect. Only the QADC can clear the single-scan enable bit. The completion flag, completion interrupt, or queue status is used to determine when the queue has completed.

After the single-scan enable bit is set, a trigger event causes the QADC to begin execution with the first CCW in the queue. The single-scan enable bit remains set until the queue is completed. After the queue reaches completion, the QADC resets the single-scan enable bit to 0. Writing the single-scan enable bit to a 1 or a 0 before the queue scan is complete has no effect; however, if the queue operating mode is changed, the new queue operating mode and the value of the single-scan enable bit are recognized immediately. The conversion in progress is aborted, and the new queue operating mode takes effect.

In software-initiated single-scan mode, writing a 1 to the single-scan enable bit causes the QADC to generate a trigger event internally, and queue execution begins immediately. In the other single-scan queue operating modes, once the single-scan enable bit is written, the selected trigger event must occur before the queue can start. The single-scan enable bit allows the entire queue to be scanned once. A trigger overrun is captured if a trigger event occurs during queue execution in an edge-sensitive external trigger mode or a periodic/interval timer mode.

In the interval timer single-scan mode, the next expiration of the timer is the trigger event for the queue. After queue execution is complete, the queue status is shown as idle. The queue can be restarted by setting the single-scan enable bit to 1. Queue execution begins with the first CCW in the queue.

28.8.6.1 Software-Initiated Single-Scan Mode

Software can initiate the execution of a scan sequence for queue 1 or 2 by selecting software-initiated single-scan mode and writing the single-scan enable bit in QACR1 or QACR2. A trigger event is generated internally and the QADC immediately begins execution of the first CCW in the queue. If a pause occurs, another trigger event is generated internally, and then execution continues without pausing.

The QADC automatically performs the conversions in the queue until an end-of-queue condition is encountered. The queue remains idle until the single-scan enable bit is again set. While the time to internally generate and act on a trigger event is very short, the queue status field can be read as momentarily indicating that the queue is paused. The trigger overrun flag is never set while in software-initiated single-scan mode.

The software-initiated single-scan mode is useful when:

- Complete control of queue execution is required
- There is a need to easily alternate between several queue sequences

28.8.6.2 Externally Triggered Single-Scan Mode

The externally triggered single-scan mode is available on both queue 1 and queue 2. Both rising and falling edge triggered modes are available. A scan must be enabled by setting the single-scan enable bit for the queue.

The first external trigger edge causes the queue to be executed one time. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. After the queue is completed, the QADC clears the single-scan enable bit. The single-scan enable bit can be written again to allow another scan of the queue to be initiated by the next external trigger edge.

The externally triggered single-scan mode is useful when the input trigger rate can exceed the queue execution rate. Analog samples can be taken in sync with an external event, even though application software does not require data taken from every edge. Externally triggered single-scan mode can be enabled to get one set of data and, at a later time, be enabled again for the next set of samples.

When a pause bit is encountered during externally triggered single-scan mode, another trigger event is required for queue execution to continue. Software involvement is not required for queue execution to continue from the paused state.

28.8.6.3 Externally Gated Single-Scan Mode

The QADC provides external gating for queue 1 only. When externally gated single-scan mode is selected, the input level on the associated external trigger signal enables and disables queue execution. The polarity of the external gate signal is fixed so that only a high level opens the gate and a low level closes the gate. Once the gate is open, each CCW is read and the indicated conversions are performed until the gate is closed. Queue scan must be enabled by setting the single-scan enable bit for queue 1. If a pause is encountered, the pause flag does not set, and execution continues without pausing.

While the gate is open, queue 1 executes one time. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. When queue 1 completes, the QADC sets the completion flag (CF1) and clears the single-scan enable bit. Set the single-scan enable bit again to allow another scan of queue 1 to be initiated during the next open gate.

If the gate closes before queue 1 completes execution, the current CCW completes, execution of queue 1 stops, the single-scan enable bit is cleared, and the PF1 bit is set. The CWPQ1 field can be read to determine the last valid conversion in the queue. The single-scan enable bit must be set again and the PF1

bit should be cleared before another scan of queue 1 is initiated during the next open gate. The start of queue 1 is always the first CCW in the CCW table.

Because the gate level is only sampled after each conversion during queue execution, closing the gate for a period less than a conversion time interval does not guarantee the closure will be captured.

28.8.6.4 Interval Timer Single-Scan Mode

Both queues can use the periodic/interval timer in a single-scan queue operating mode. The timer interval can range from 2^7 to 2^{17} QCLK cycles in binary multiples. When the interval timer single-scan mode is selected and the single-scan enable bit is set in QACR1 or QACR2, the timer begins counting. When the time interval elapses, an internal trigger event is generated to start the queue and the QADC begins execution with the first CCW.

The QADC automatically performs the conversions in the queue until a pause or an end-of-queue condition is encountered. When a pause occurs, queue execution stops until the timer interval elapses again, and queue execution continues. When queue execution reaches an end-of-queue situation, the single-scan enable bit is cleared. Set the single-scan enable bit again to allow another scan of the queue to be initiated by the interval timer.

The interval timer generates a trigger event whenever the time interval elapses. The trigger event may cause queue execution to continue following a pause or may be considered a trigger overrun. Once queue execution is completed, the single-scan enable bit must be set again to allow the timer to count again.

Normally, only one queue is enabled for interval timer single-scan mode, and the timer will reset at the end-of-queue. However, if both queues are enabled for either single-scan or continuous interval timer mode, the end-of-queue condition will not reset the timer while the other queue is active. In this case, the timer will reset when both queues have reached end-of-queue. See Section 28.8.9, “Periodic/Interval Timer for a definition of interval timer reset conditions.

The interval timer single-scan mode can be used in applications that need coherent results. For example:

- When it is necessary that all samples are guaranteed to be taken during the same scan of the analog signals
- When the interrupt rate in the periodic timer continuous-scan mode would be too high
- In sensitive battery applications, where the interval timer single-scan mode uses less power than the software-initiated continuous-scan mode

28.8.7 Continuous-Scan Modes

A continuous-scan queue operating mode is used to execute multiple passes through a sequence of conversions defined by a queue. By programming the MQ1 field in QACR1 or the MQ2 field in QACR2, these modes can be selected:

- Software-initiated continuous-scan mode
- Externally triggered continuous-scan mode
- Externally gated continuous-scan mode
- Periodic timer continuous-scan mode

NOTE

Queue 2 cannot be programmed for externally gated continuous-scan mode.

When a queue is programmed for a continuous-scan mode, the single-scan enable bit in the queue control register does not have any meaning or effect. As soon as the queue operating mode is programmed, the selected trigger event can initiate queue execution.

In the case of software-initiated continuous-scan mode, the trigger event is generated internally and queue execution begins immediately. In the other continuous-scan queue operating modes, the selected trigger event must occur before the queue can start. A trigger overrun is captured if a trigger event occurs during queue execution in the externally triggered continuous-scan mode or the periodic timer continuous-scan mode.

After queue execution is complete, the queue status is shown as idle. Because the continuous-scan queue operating modes allow the entire queue to be scanned multiple times, software involvement is not needed for queue execution to continue from the idle state. The next trigger event causes queue execution to begin again, starting with the first CCW in the queue.

NOTE

In continuous-scan modes, all samples are guaranteed to be taken during one pass through the queue (coherently), except when a queue 1 trigger event halts queue 2 execution. The time between consecutive conversions has been designed to be consistent. However, for queues that end with a CCW containing the EOQ code (channel 63), the time between the last queue conversion and the first queue conversion requires one additional CCW fetch cycle. Continuous samples are not coherent at this boundary.

In addition, the time from trigger to first conversion cannot be guaranteed, because it is a function of clock synchronization, programmable trigger events, queue priorities, and so on.

28.8.7.1 Software-Initiated Continuous-Scan Mode

When software-initiated continuous-scan mode is selected, the trigger event is generated automatically by the QADC. Queue execution begins immediately. If a pause is encountered, another trigger event is generated internally, and execution continues without pausing. When the end-of-queue is reached, another internal trigger event is generated and queue execution restarts at the beginning of the queue.

While the time to internally generate and act on a trigger event is very short, the queue status field can be read as momentarily indicating that the queue is idle. The trigger overrun flag is never set while in software-initiated continuous-scan mode.

The software-initiated continuous-scan mode keeps the result registers updated more frequently than any of the other queue operating modes. The result table can always be read to get the latest converted value for each channel. The channels scanned are kept up to date by the QADC without software involvement.

The software-initiated continuous-scan mode may be chosen for either queue, but is normally used only with queue 2. When software-initiated continuous-scan mode is chosen for queue 1, that queue operates continuously and queue 2, being lower in priority, never gets executed. The short interval of time between a queue 1 completion and the subsequent trigger event is not sufficient to allow queue 2 execution to begin.

The software-initiated continuous-scan mode is a useful choice with queue 2 for converting channels that do not need to be synchronized to anything or for slow-to-change analog channels. Interrupts are normally not used with the software-initiated continuous-scan mode. Rather, the latest conversion results can be read from the result table at any time. Once initiated, software action is not needed to sustain conversions of channel.

28.8.7.2 Externally Triggered Continuous-Scan Mode

The QADC provides external trigger signals for both queues. When externally triggered continuous-scan mode is selected, a transition on the associated external trigger signal initiates queue execution. The

polarity of the external trigger signal is programmable, so that a mode which begins queue execution on the rising or falling edge can be selected. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. When the next external trigger edge is detected, queue execution begins again automatically. Software involvement is not needed between trigger events.

When a pause bit is encountered in externally triggered continuous-scan mode, another trigger event is required for queue execution to continue. Software involvement is not needed for queue execution to continue from the paused state.

Some applications need to synchronize the sampling of analog channels to external events. There are cases when it is not possible to use software initiation of the queue scan sequence, because interrupt response times vary. Externally triggered continuous-scan mode is useful in these cases.

28.8.7.3 Externally Gated Continuous-Scan Mode

The QADC provides external gating for queue 1 only. When externally gated continuous-scan mode is selected, the input level on the associated external trigger signal enables and disables queue execution. The polarity of the external gate signal is fixed so that a high level opens the gate and a low level closes the gate. Once the gate is open, each CCW is read and the indicated conversions are performed until the gate is closed. When the gate opens again, queue execution automatically restarts at the beginning of the queue. Software involvement is not needed between trigger events. If a pause in a CCW is encountered, the pause flag does not set, and execution continues without pausing.

The purpose of externally gated continuous-scan mode is to continuously collect digitized samples while the gate is open and to have the most recent samples available. It is up to the programmer to ensure that the gate is not opened so long that an end-of-queue is reached.

In the event that the queue completes before the gate closes, the CF1 flag will set, and the queue will roll over to the beginning and continue conversions until the gate closes. If the gate remains open and the CF1 flag is not cleared, when the queue completes a second time the TOR1 flag will set and the queue will roll-over again. The queue will continue to execute until the gate closes or the mode is disabled.

If the gate closes before queue 1 completes execution, the QADC stops and sets the PF1 bit to indicate an incomplete queue. The CWPQ1 field can be read to determine the last valid conversion in the queue. If the gate opens again, execution of queue 1 restarts. The start of queue 1 is always the first CCW in the CCW table. The condition of the gate is only sampled after each conversion during queue execution, so closing the gate for a period less than a conversion time interval does not guarantee the closure will be captured.

28.8.7.4 Periodic Timer Continuous-Scan Mode

The QADC includes a dedicated periodic timer for initiating a scan sequence on queue 1 and/or queue 2. A programmable timer interval ranging from 2^7 to 2^{17} times the QCLK period in binary multiples can be selected. The QCLK period is prescaled down from the MCU clock.

When a periodic timer continuous-scan mode is selected, the timer begins counting. After the programmed interval elapses, the timer generated trigger event starts the appropriate queue. The QADC automatically performs the conversions in the queue until an end-of-queue condition or a pause is encountered. When a pause occurs, the QADC waits for the periodic interval to expire again, then continues with the queue. Once EOQ has been detected, the next trigger event causes queue execution to restart with the first CCW in the queue.

The periodic timer generates a trigger event whenever the time interval elapses. The trigger event may cause queue execution to continue following a pause or queue completion or may be considered a trigger overrun. As with all continuous-scan queue operating modes, software action is not needed between

trigger events. Because both queues may be triggered by the periodic/interval timer, see Section 28.8.9, "Periodic/Interval Timer for a summary of periodic/interval timer reset conditions.

28.8.8 QADC Clock (QCLK) Generation

Figure 28-42 is a block diagram of the QCLK subsystem. The QCLK provides the timing for the A/D converter state machine which controls the timing of the conversion. The QCLK is also the input to a 17-stage binary divider which implements the periodic/interval timer. To retain the specified analog conversion accuracy, the QCLK frequency (f_{QCLK}) must be within the tolerance specified in Chapter 33, "Electrical Characteristics".

Before using the QADC, the prescaler must be initialized with values that put the QCLK within the specified range. Though most applications initialize the prescaler once and do not change it, write operations to the prescaler fields are permitted.

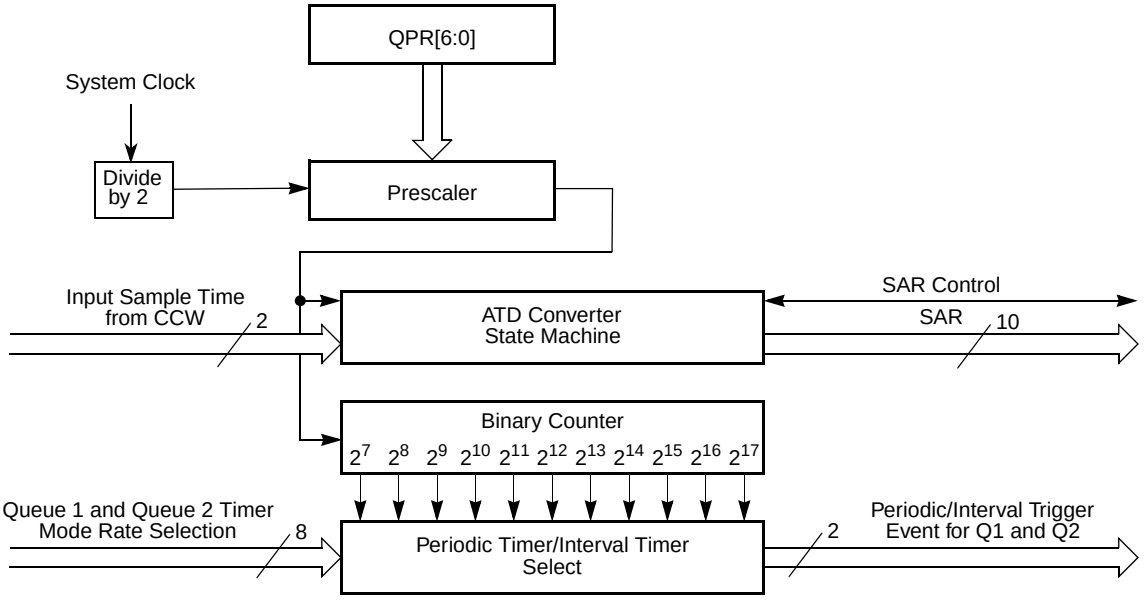


Figure 28-42. QADC Clock Subsystem Functions

CAUTION

A change in the prescaler value while a conversion is in progress is likely to corrupt the result. Therefore, any prescaler write operation should be done only when both queues are in the disabled modes.

To accommodate the wide range of the system clock frequency, QCLK is generated by a programmable prescaler which divides the system clock. To allow the A/D conversion time to be maximized across the spectrum of system clock frequencies, the QADC prescaler permits the QCLK frequency to be software selectable. The frequency of QCLK is set with the QPR field in QACR0.

28.8.9 Periodic/Interval Timer

The QADC periodic/interval timer can be used to generate trigger events at a programmable interval, initiating execution of queue 1 and/or queue 2. The periodic/interval timer stays reset under these conditions:

- Both queue 1 and queue 2 are programmed to any mode which does not use the periodic/interval timer.
- System reset is asserted.
- Stop mode is enabled.
- Debug mode is enabled.

NOTE

Interval timer single-scan mode does not start the periodic/interval timer until the single-scan enable bit is set.

These conditions will cause a pulsed reset of the periodic/interval timer during use:

- A queue 1 operating mode change to a mode which uses the periodic/interval timer, even if queue 2 is already using the timer
- A queue 2 operating mode change to a mode which uses the periodic/interval timer, provided queue 1 is not in a mode which uses the periodic/interval timer
- Roll over of the timer

During stop mode, the periodic/interval timer is held in reset. Because stop mode causes QACR1 and QACR2 to be reset to 0, a valid periodic or interval timer mode must be written after leaving stop mode to release the timer from reset.

When QADC debug mode is entered and a periodic or interval timer mode is selected, the timer counter is reset after the conversion in progress completes. When the periodic or interval timer mode has been enabled (the timer is counting), but a trigger event has not been issued, debug mode takes effect immediately, and the timer is held in reset. Removal of the QADC debug condition restarts the counter from the beginning. Refer to Section 28.3.1, “Debug Mode for more information.

28.8.10 Conversion Command Word Table

The conversion command word (CCW) table is 64 half-word (128 byte) long RAM with 10 bits of each entry implemented. The CCW table is written by the user and is not modified by the QADC. Each CCW requests the conversion of one analog channel to a digital result. The CCW specifies the analog channel number, the input sample time, and whether the queue is to pause after the current CCW. The 10 implemented bits of the CCW can be read and written. The remaining six bits are unimplemented and read as 0s; write operations have no effect. Each location in the CCW table corresponds to a location in the result word table. When a conversion is completed for a CCW entry, the 10-bit result is written in the corresponding result word entry.

The beginning of queue 1 is the first location in the CCW table. The first location of queue 2 is specified by the beginning of queue 2 pointer field (BQ2) in QACR2. To dedicate the entire CCW table to queue 1, place queue 2 in disabled mode and write BQ2 to 64 or greater. To dedicate the entire CCW table to queue 2, place queue 1 in disabled mode and set BQ2 to the first location in the CCW table (CCW0).

Figure 28-43 illustrates the operation of the queue structure.

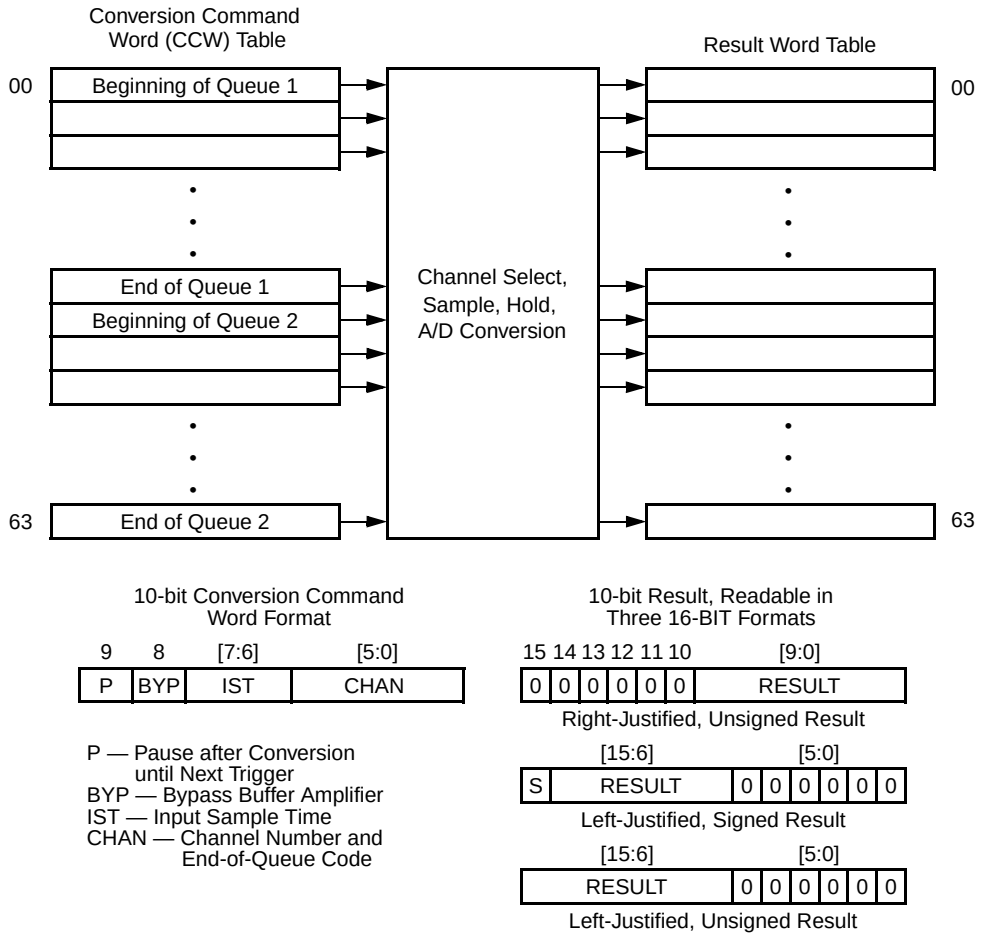


Figure 28-43. QADC Conversion Queue Operation

To prepare the QADC for a scan sequence, write to the CCW table to specify the desired channel conversions. The criteria for queue execution is established by selecting the queue operating mode. The queue operating mode determines what type of trigger event starts queue execution. A trigger event refers to any of the ways that cause the QADC to begin executing the CCWs in a queue or subqueue. An external trigger is only one of the possible trigger events.

A scan sequence may be initiated by:

- A software command
- Expiration of the periodic/interval timer
- An external trigger signal
- An external gated signal (queue 1 only)

The queue can be scanned in single pass or continuous fashion. When a single-scan mode is selected, the scan must be engaged by setting the single-scan enable bit. When a continuous-scan mode is selected, the queue remains active in the selected queue operating mode after the QADC completes each queue scan sequence.

During queue execution, the QADC reads each CCW from the active queue and executes conversions in three stages:

- Initial sample
- Final sample

- Resolution

During initial sample, a buffered version of the selected input channel is connected to the sample capacitor at the input of the sample buffer amplifier.

During the final sample period, the sample buffer amplifier is bypassed, and the multiplexer input charges the sample capacitor directly. Each CCW specifies a final input sample time of 2, 4, 8, or 16 QCLK cycles. When an analog-to-digital conversion is complete, the result is written to the corresponding location in the result word table. The QADC continues to sequentially execute each CCW in the queue until the end of the queue is detected or a pause bit is found in a CCW.

When the pause bit is set in the current CCW, the QADC stops execution of the queue until a new trigger event occurs. The pause status flag bit is set, and an interrupt may optionally be requested. After the trigger event occurs, the paused state ends, and the QADC continues to execute each CCW in the queue until another pause is encountered or the end of the queue is detected.

An end-of-queue condition occurs when:

- The CCW channel field is programmed with 63 to specify the end of the queue.
- The end-of-queue 1 is implied by the beginning of queue 2, which is specified by the BQ2 field in QACR2.
- The physical end of the queue RAM space defines the end of either queue.

When any of the end-of-queue conditions is recognized, a queue completion flag is set, and if enabled, an interrupt is requested. These situations prematurely terminate queue execution:

- Queue 1 is higher in priority than queue 2. When a trigger event occurs on queue 1 during queue 2 execution, the execution of queue 2 is suspended by aborting the execution of the CCW in progress, and queue 1 execution begins. When queue 1 execution is complete, queue 2 conversions restart with the first CCW entry in queue 2 or the first CCW of the queue 2 subqueue being executed when queue 2 was suspended. Alternately, conversions can restart with the aborted queue 2 CCW entry. The RESUME bit in QACR2 selects where queue 2 begins after suspension. By choosing to re-execute all of the suspended queue 2 CCWs (RESUME = 0), all of the samples are guaranteed to have been taken during the same scan pass. However, a high trigger event rate for queue 1 can prevent completion of queue 2. If this occurs, execution of queue 2 can begin with the aborted CCW entry (RESUME = 1).
- Any conversion in progress for a queue is aborted when that queue's operating mode is changed to disabled. Putting a queue into the disabled mode does not power down the converter.
- Changing a queue's operating mode to another valid mode aborts any conversion in progress. The queue restarts at its beginning once an appropriate trigger event occurs.
- For low-power operation, the stop bit can be set to prepare the module for a loss of clocks. The QADC aborts any conversion in progress when stop mode is entered.
- When the QADC debug bit is set and the CPU enters background debug mode, the QADC freezes at the end of the conversion in progress. After leaving debug mode, the QADC resumes queue execution beginning with the next CCW entry. Refer to Section 28.3.1, "Debug Mode" for more information.

28.8.11 Result Word Table

The result word table is a 64 half-word (128 byte) long by 10-bit wide RAM. An entry is written by the QADC after completing an analog conversion specified by the corresponding CCW table entry. The result word table can be read or written, but in normal operation is only read to obtain analog conversions from the QADC. Unimplemented bits read as 0s and writes have no effect.

NOTE

Although the result RAM can be written, some write operations, like bit manipulation, may not operate as expected because the hardware cannot access a true 16-bit value.

While there is only one result word table, the half-word (16-bit) data can be accessed in three different data formats:

- Right justified with 0s in the higher order unused bits
- Left justified with the most significant bit inverted to form a sign bit, and 0s in the unused lower order bits
- Left justified with 0s in the lower order unused bits

The left justified, signed format corresponds to a half-scale, offset binary, two's complement data format. The address used to read the result table determines the data alignment format. All write operations to the result word table are right justified.

28.9 Signal Connection Considerations

The QADC requires accurate, noise-free input signals for proper operation. This section discusses the design of external circuitry to maximize QADC performance.

28.9.1 Analog Reference Signals

No A/D converter can be more accurate than its analog reference. Any noise in the reference can result in at least that much error in a conversion. The reference for the QADC, supplied by signals V_{RH} and V_{RL}, should be low-pass filtered from its source to obtain a noise-free, clean signal. In many cases, simple capacitive bypassing may suffice. In extreme cases, inductors or ferrite beads may be necessary if noise or RF energy is present. External resistance may introduce error in this architecture under certain conditions. Any series devices in the filter network should contain a minimum amount of DC resistance.

For accurate conversion results, the analog reference voltages must be within the limits defined by V_{DDA} and V_{SSA}, as explained in this subsection.

28.9.2 Analog Power Signals

The analog supply signals (V_{DDA} and V_{SSA}) define the limits of the analog reference voltages (V_{RH} and V_{RL}) and of the analog multiplexer inputs. Figure 28-44 is a diagram of the analog input circuitry.

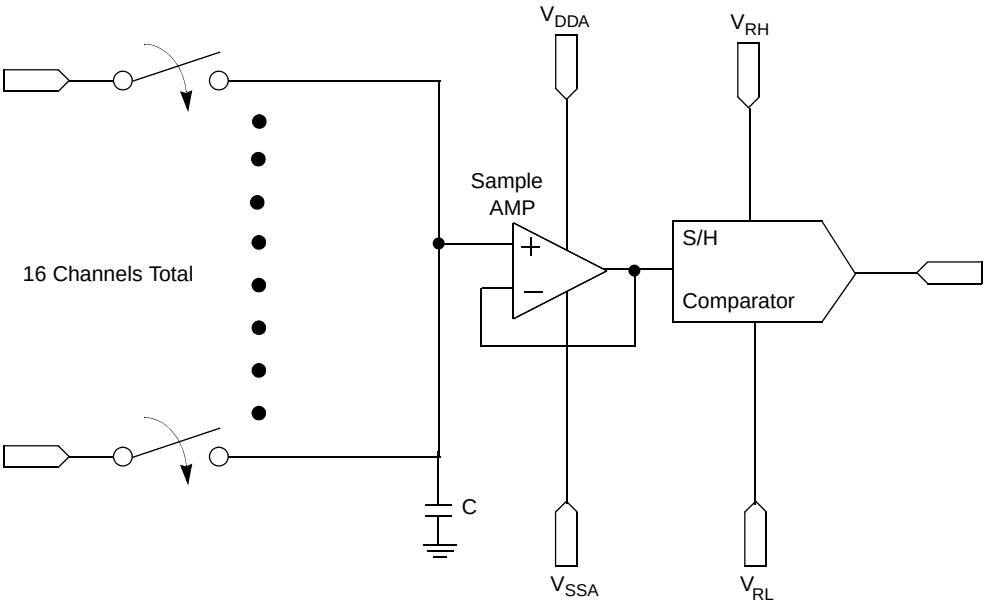


Figure 28-44. Equivalent Analog Input Circuitry

Because the sample amplifier is powered by V_{DDA} and V_{SSA}, it can accurately transfer input signal levels up to but not exceeding V_{DDA} and down to but not below V_{SSA}. If the input signal is outside of this range, the output from the sample amplifier is clipped.

In addition, V_{RH} and V_{RL} must be within the range defined by V_{DDA} and V_{SSA}. As long as V_{RH} is less than or equal to V_{DDA}, and V_{RL} is greater than or equal to V_{SSA}, and the sample amplifier has accurately transferred the input signal, resolution is ratiometric within the limits defined by V_{RL} and V_{RH}. If V_{RH} is greater than V_{DDA}, the sample amplifier can never transfer a full-scale value. If V_{RL} is less than V_{SSA}, the sample amplifier can never transfer a 0 value.

Figure 28-45 shows the results of reference voltages outside the range defined by V_{DDA} and V_{SSA}. At the top of the input signal range, V_{DDA} is 10 mV lower than V_{RH}. This results in a maximum obtainable 10-bit conversion value 0x03fe. At the bottom of the signal range, V_{SSA} is 15 mV higher than V_{RL}, resulting in a minimum obtainable 10-bit conversion value of 0x0003.

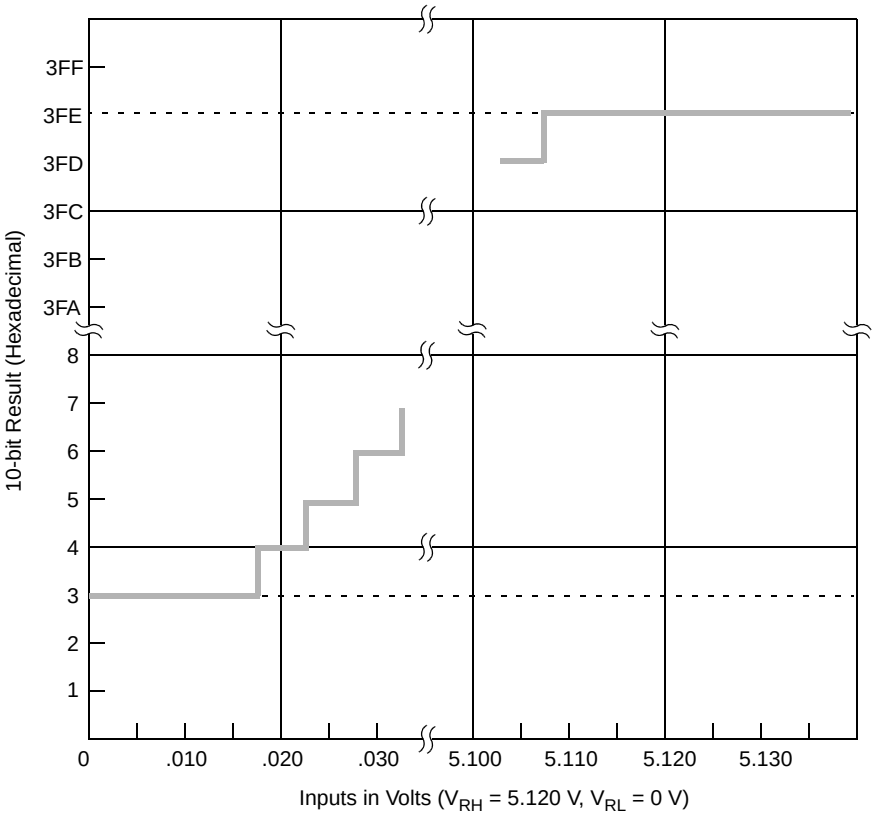


Figure 28-45. Errors Resulting from Clipping

28.9.3 Conversion Timing Schemes

This section contains some conversion timing examples. Figure 28-46 shows the timing for basic conversions where it is assumed that:

- Q1 begins with CCW0 and ends with CCW3.
- CCW0 has pause bit set.
- CCW1 does not have pause bit set.
- External trigger rising edge for Q1
- CCW4 = BQ2 and Q2 is disabled.
- Q1 RES shows relative result register updates.

Recall that when QS = 0, both queues are disabled; when QS = 8, queue 1 is active and queue 2 is idle; and when QS = 4; queue 1 is paused and queue 2 is disabled.

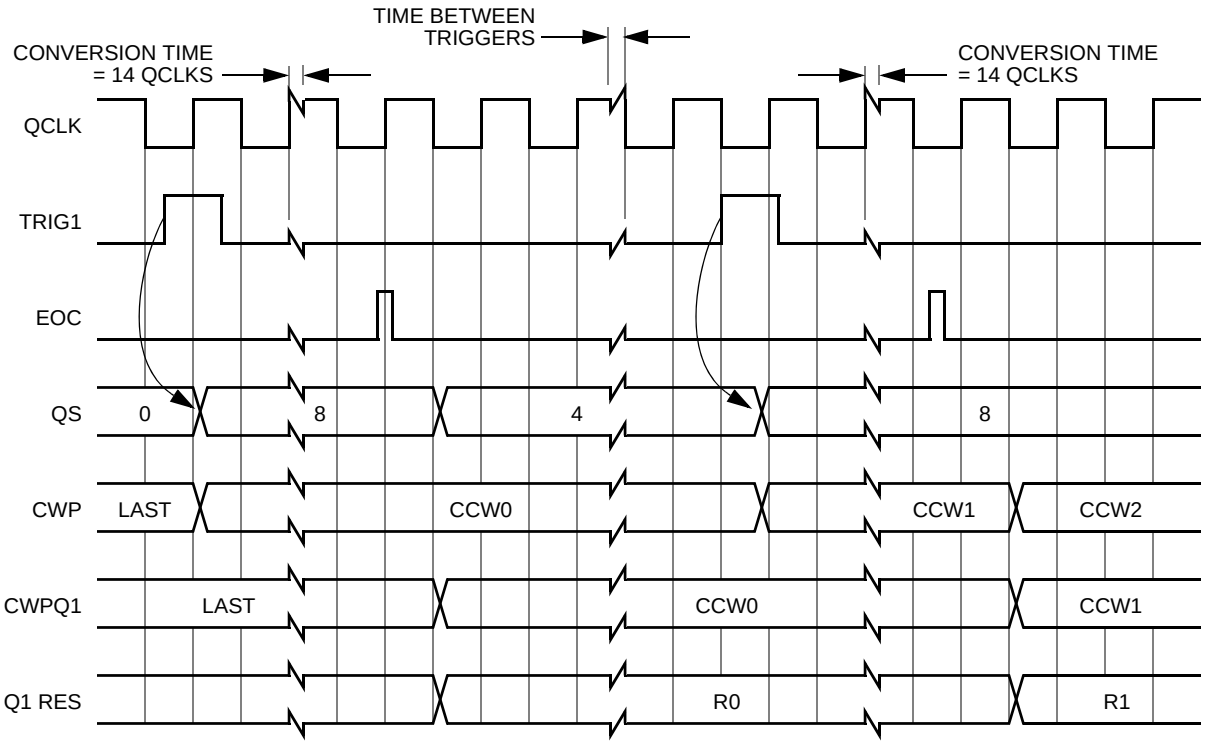


Figure 28-46. External Positive Edge Trigger Mode Timing with Pause

A time separator is provided between the triggers and the end of conversion (EOC). The relationship to QCLK displayed is not guaranteed.

CWPQ1 and CWPQ2 typically lag CWP and only match CWP when the associated queue is inactive. Another way to view CWPQ1 and CWPQ2 is that these registers update when EOC triggers the write to the result register.

For the CCW with the pause bit set (CCW0), CWP does not increment until triggered. For the CCW with the pause bit clear (CCW1), the CWP increments with the EOC.

The conversion results Q1 RESx show the result associated with CCWx, such that R0 represents the result associated with CCW0.

Figure 28-47 shows the timing for conversions in externally gated single-scan with same assumptions in example 1 except:

- No pause bits set in any CCW
- Externally gated single scan mode for Q1
- Single scan enable bit (SSE1) is set.

When the gate closes and opens again, the conversions start with the first CCW in Q1.

When the gate closes, the active conversion completes before the queue goes idle.

When Q1 completes, both the CF1 bit sets and the SSE bit clears.

In this mode, the PF1 bit sets to reflect that a gate closing occurred before the queue completed.

Figure 28-48 shows the timing for conversions in externally gated continuous scan mode with the same assumptions as in Figure 28-47.

At the end of Q1, the completion flag CF1 sets and the queue restarts. If the queue starts a second time and completes, the trigger overrun flag TOR1 sets.

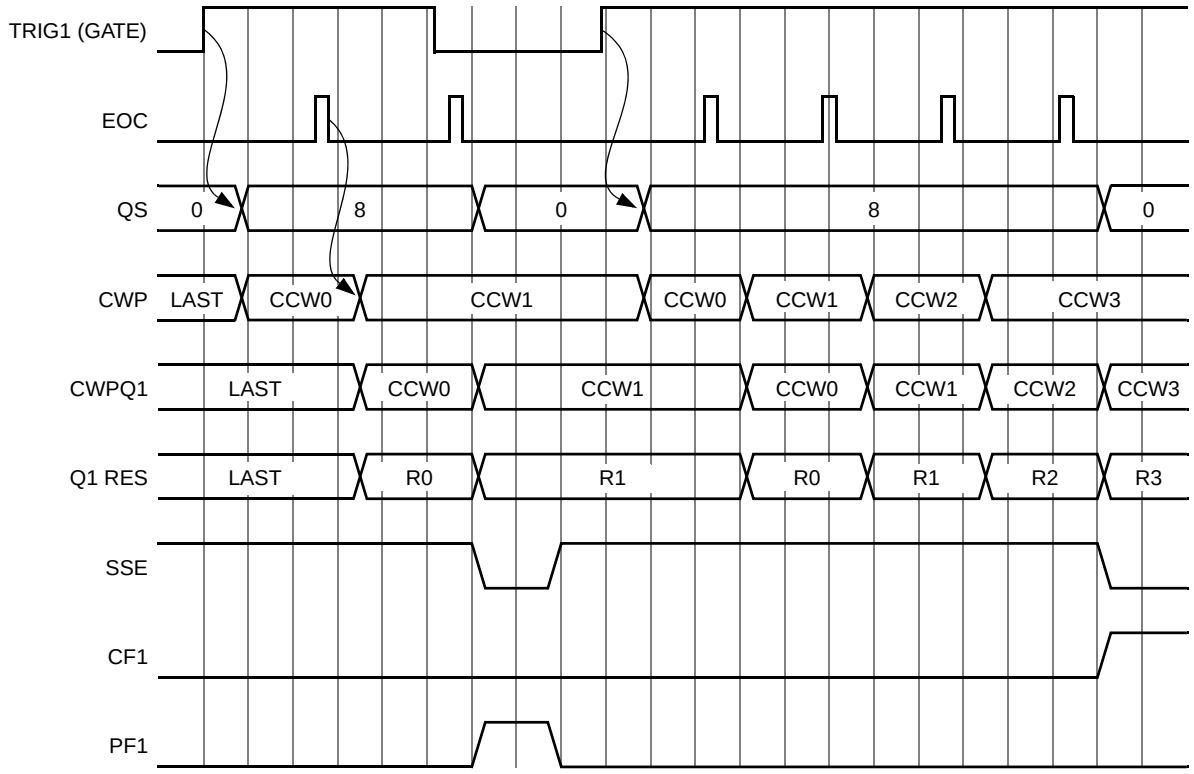


Figure 28-47. Gated Mode, Single Scan Timing

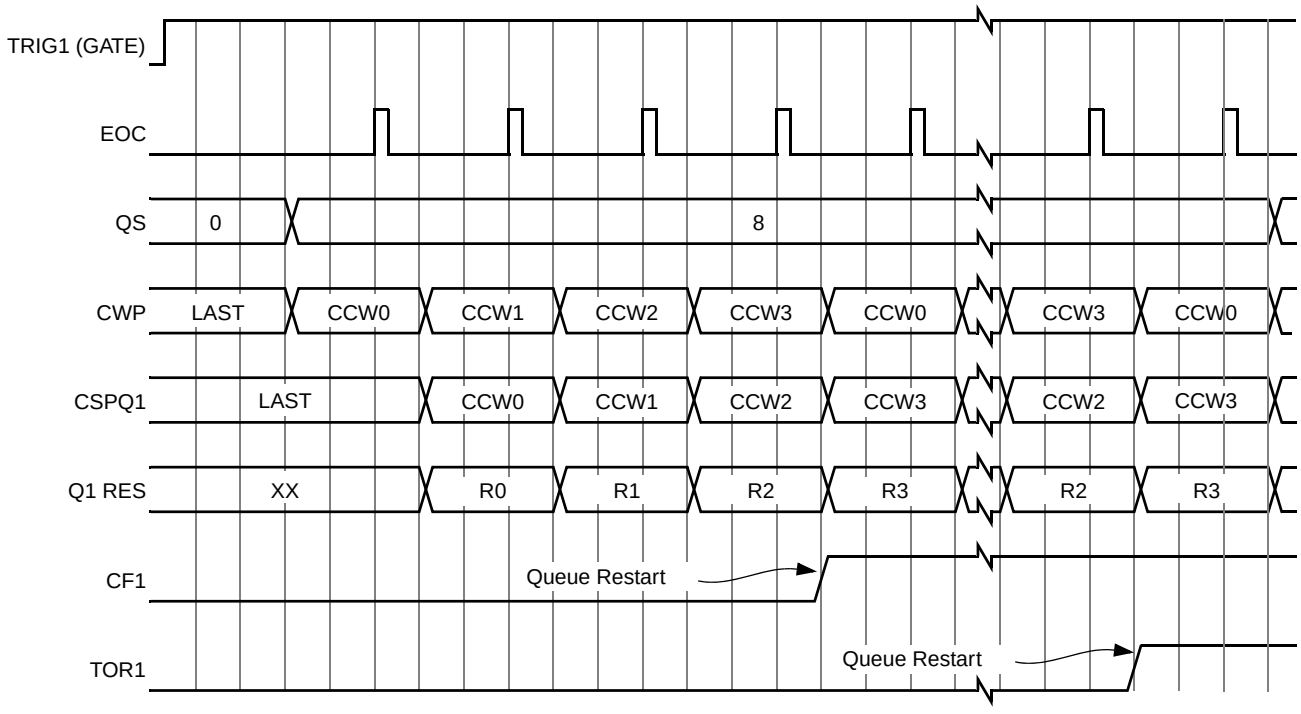


Figure 28-48. Gated Mode, Continuous Scan Timing

28.9.4 Analog Supply Filtering and Grounding

Two important factors influencing performance in analog integrated circuits are supply filtering and grounding. Generally, digital circuits use bypass capacitors on every V_{DD}/V_{SS} signal pair. This applies to analog subsystems and submodules also. Equally important as bypassing is the distribution of power and ground.

Analog supplies should be isolated from digital supplies as much as possible. This necessity stems from the higher performance requirements often associated with analog circuits. Therefore, deriving an analog supply from a local digital supply is not recommended. However, if for cost reasons digital and analog power are derived from a common regulator, filtering of the analog power is recommended in addition to the bypassing of the supplies already mentioned. For example, an RC low pass filter could be used to isolate the digital and analog supplies when generated by a common regulator. If multiple high precision analog circuits are locally employed (for example, two A/D converters), the analog supplies should be isolated from each other as sharing supplies introduces the potential for interference between analog circuits.

Grounding is the most important factor influencing analog circuit performance in mixed signal systems (or in standalone analog systems). Close attention must be paid not to introduce additional sources of noise into the analog circuitry. Common sources of noise include ground loops, inductive coupling, and combining digital and analog grounds together inappropriately.

The problem of how and when to combine digital and analog grounds arises from the large transients which the digital ground must handle. If the digital ground is not able to handle the large transients, the associated current can return to ground through the analog ground. It is this excess current overflowing into the analog ground which causes performance degradation by developing a differential voltage between the true analog ground and the microcontroller’s ground pins. The end result is that the ground observed by the analog circuit is no longer true ground and thus skews converter performance.

Two similar approaches to improving or eliminating the problems associated with grounding excess transient currents involve star-point ground systems. One approach is to star-point the different grounds at the power supply origin, thus keeping the ground isolated. Refer to Figure 28-49.

Another approach is to star-point the different grounds near the analog ground signal on the microcontroller by using small traces for connecting the non-analog grounds to the analog ground. The small traces are meant only to accommodate dc differences, not ac transients.

NOTE

This star-point scheme still requires adequate grounding for digital and analog subsystems in addition to the star-point ground.

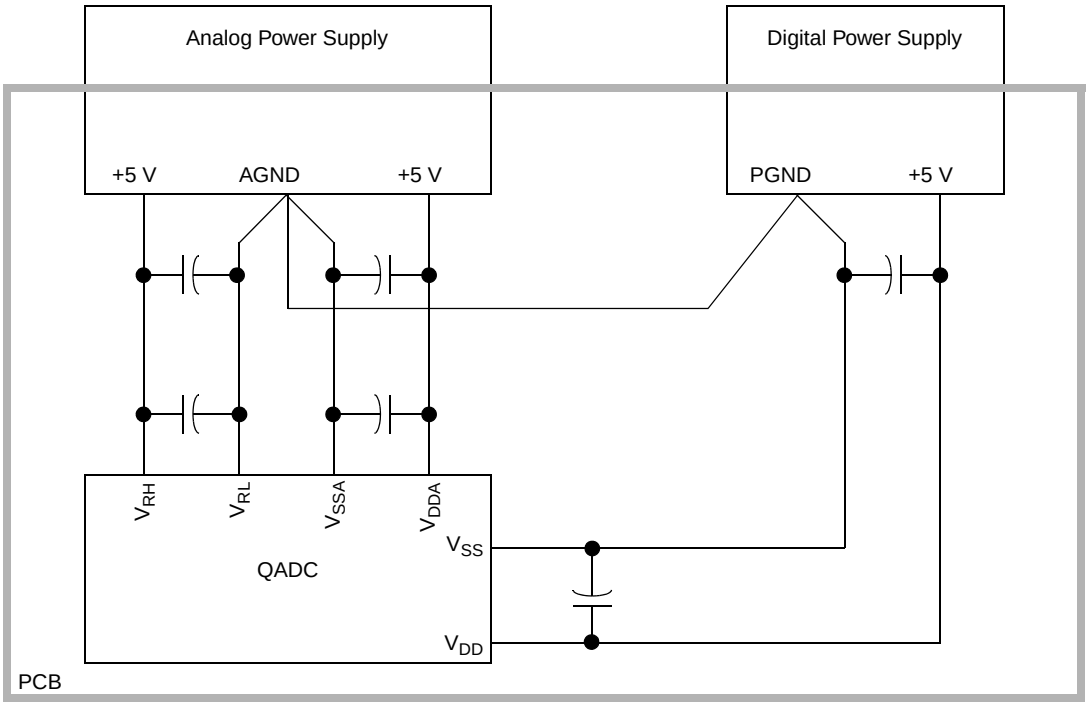


Figure 28-49. Star-Ground at the Point of Power Supply Origin

Other suggestions for PCB layout in which the QADC is employed include:

- Analog ground must be low impedance to all analog ground points in the circuit.
- Bypass capacitors should be as close to the power pins as possible.
- The analog ground should be isolated from the digital ground. This can be done by cutting a separate ground plane for the analog ground.
- Non-minimum traces should be utilized for connecting bypass capacitors and filters to their corresponding ground/power points.
- Minimum distance for trace runs when possible.

28.9.5 Accommodating Positive/Negative Stress Conditions

Positive or negative stress refers to conditions which exceed nominally defined operating limits. Examples include applying a voltage exceeding the normal limit on an input (for example, voltages outside of the suggested supply/reference ranges) or causing currents into or out of the pin which exceed normal limits. QADC specific considerations are voltages greater than V_{DDA} or less than V_{SSA} applied to an analog input which cause excessive currents into or out of the input. Refer to Chapter 33, “Electrical Characteristics” for more information on exact magnitudes.

Either stress conditions can potentially disrupt conversion results on neighboring inputs. Parasitic devices, associated with CMOS processes, can cause an immediate disruptive influence on neighboring pins. Common examples of parasitic devices are diodes to substrate and bipolar devices with the base terminal tied to substrate (V_{SS}/V_{SSA} ground). Under stress conditions, current injected on an adjacent signal can cause errors on the selected channel by developing a voltage drop across the selected channel’s impedances.

Figure 28-50 shows an active parasitic bipolar NPN transistor when an input signal is subjected to negative stress conditions. Figure 28-51 shows positive stress conditions can activate a similar PNP transistor.

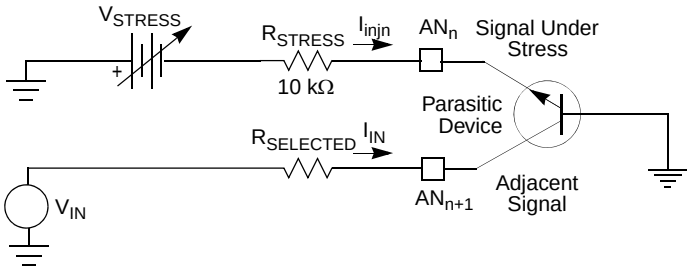


Figure 28-50. Input Signal Subjected to Negative Stress

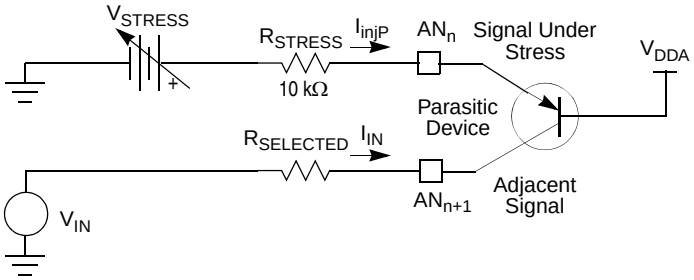


Figure 28-51. Input Signal Subjected to Positive Stress

The current into the signal (I_{INJN} or I_{INJP}) under negative or positive stress is determined by these equations:

$$I_{INJN} = \frac{-(V_{Stress} - V_{BE})}{R_{Stress}} \quad \textit{Eqn. 28-1}$$

$$I_{INJP} = \frac{V_{Stress} - V_{EB} - V_{DDA}}{R_{Stress}} \quad \textit{Eqn. 28-2}$$

Where:

V_{Stress} = Adjustable voltage source

V_{EB} = Parasitic PNP emitter/base voltage

V_{BE}= Parasitic NPN base/emitter voltage

R_{Stress}= Source impedance (10 kΩ resistor in Figure 28-50 and Figure 28-51 on stressed channel)

$R_{Selected}$ = Source impedance on channel selected for conversion

The current into (I_{In}) the neighboring pin is determined by the K_N (current coupling ratio) of the parasitic bipolar transistor (K_N ‹‹ 1). The I_{In} can be expressed by this equation:

$I_{In} = – K_N * I_{INJ}$

Where:

I_{INJ} is either I_{INJN} or I_{INJP}.

A method for minimizing the impact of stress conditions on the QADC is to strategically allocate QADC inputs so that the lower accuracy inputs are adjacent to the inputs most likely to see stress conditions.

Also, suitable source impedances should be selected to meet design goals and minimize the effect of stress conditions.

28.9.6 Analog Input Considerations

The source impedance of the analog signal to be measured and any intermediate filtering should be considered whether external multiplexing is used or not. Figure 28-52 shows the connection of eight typical analog signal sources to one QADC analog input signal through a separate multiplexer chip. Also, an example of an analog signal source connected directly to a QADC analog input channel is displayed.

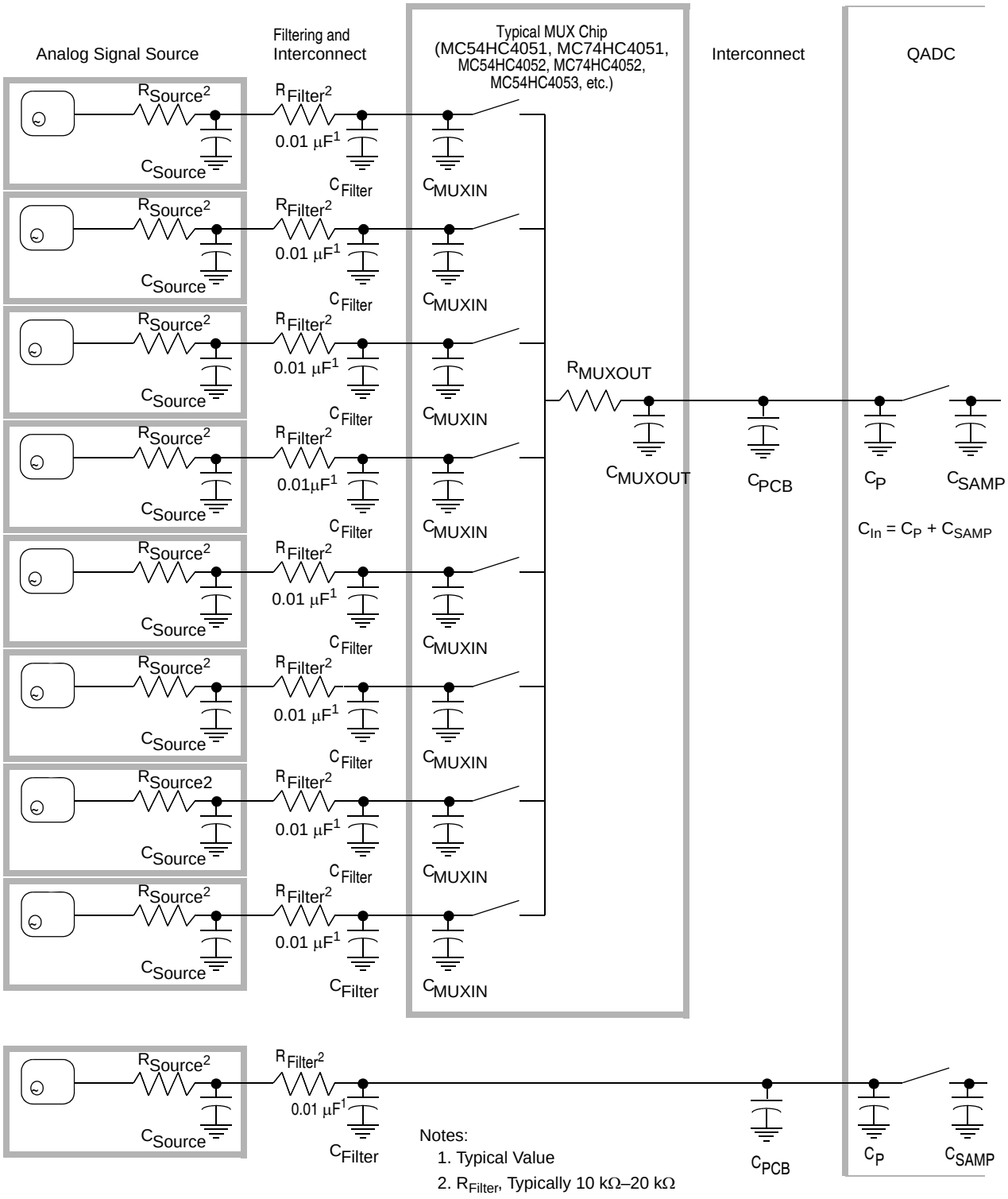


Figure 28-52. External Multiplexing of Analog Signal Sources

28.9.7 Analog Input Pins

Analog inputs should have low AC impedance at the pins. Low AC impedance can be realized by placing a capacitor with good high frequency characteristics at the input signal of the device. Ideally, that capacitor should be as large as possible (within the practical range of capacitors that still have good high-frequency characteristics). This capacitor has two effects:

- It helps attenuate any noise that may exist on the input.
- It sources charge during the sample period when the analog signal source is a high-impedance source.

Series resistance can be used with the capacitor on an input signal to implement a simple RC filter. The maximum level of filtering at the input pins is application dependent and is based on the bandpass characteristics required to accurately track the dynamic characteristics of an input. Simple RC filtering at the pin may be limited by the source impedance of the transducer or circuit supplying the analog signal to be measured. (See Section 28.9.7.2, “Error Resulting from Leakage.”) In some cases, the size of the capacitor at the pin may be very small.

Figure 28-53 is a simplified model of an input channel. Refer to this model in the following discussion of the interaction between the external circuitry and the circuitry inside the QADC.

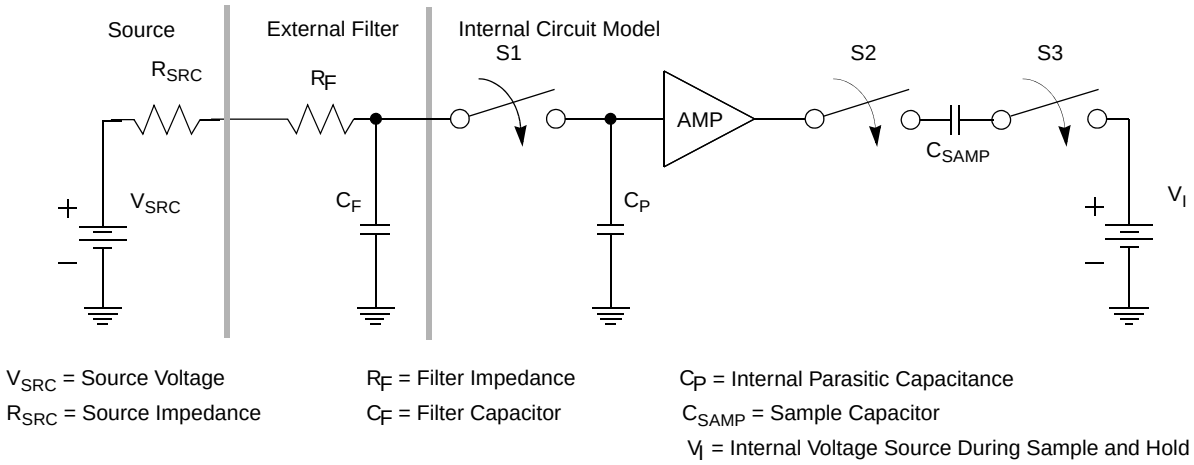


Figure 28-53. Electrical Model of an A/D Input Signal

In Figure 28-53, R_F, R_{SRC}, and C_F comprise the external filter circuit. C_P is the internal parasitic capacitor. C_{SAMP} is the capacitor array used to sample and hold the input voltage. V_I is an internal voltage source used to provide charge to C_{SAMP} during sample phase.

The following paragraphs provide a simplified description of the interaction between the QADC and the user's external circuitry. This circuitry is assumed to be a simple RC low-pass filter passing a signal from a source to the QADC input signal. These paragraphs make the following assumptions:

- The external capacitor is perfect (no leakage, no significant dielectric absorption characteristics, etc.).
- All parasitic capacitance associated with the input signal is included in the value of the external capacitor.
- Inductance is ignored.
- The “on” resistance of the internal switches is 0 ohms and the “off” resistance is infinite.

28.9.7.1 Settling Time for the External Circuit

The values for R_{SRC}, R_F, and C_F in the user's external circuitry determine the length of time required to charge C_F to the source voltage level (V_{SRC}). At time t = 0, V_{SRC} changes in Figure 28-53 while S1 is open, disconnecting the internal circuitry from the external circuitry. Assume that the initial voltage across C_F is 0. As C_F charges, the voltage across it is determined by the equation, where t is the total charge time:

$$VCF = VSRC\ (1 – e–t/(RF + RSRC)\ CF)$$

As t approaches infinity, V_{CF} will equal V_{SRC}. (This assumes no internal leakage.) With 10-bit resolution, 1/2 of a count is equal to 1/2048 full-scale value. Assuming worst case (V_{SRC} = full scale), Table 28-24 shows the required time for C_F to charge to within 1/2 of a count of the actual source voltage during 10-bit conversions. Table 28-24 is based on the RC network in Figure 28-53.

NOTE

The following times are completely independent of the A/D converter architecture (assuming the QADC is not affecting the charging).

Table 28-24. External Circuit Settling Time to 1/2 LSB

Filter Capacitor (CF)	Source Resistance ($R_F + R_{SRC}$)			
	100 Ω	**1 kΩ**	**10 kΩ**	**100 kΩ**
1 µF	760 µs	7.6 ms	76 ms	760 ms
0.1 µF	76 µs	760 µs	7.6 ms	76 ms
0.01 µF	7.6 µs	76 µs	760 µs	7.6 ms
0.001 µF	760 ns	7.6 µs	76 µs	760 µs
100 pF	76 ns	760 ns	7.6 µs	76 µs

The external circuit described in Table 28-24 is a low-pass filter. Measurements of an AC component of the external signal must take the characteristics of this filter into account.

28.9.7.2 Error Resulting from Leakage

A series resistor limits the current to a signal; therefore, input leakage acting through a large source impedance can degrade A/D accuracy. The maximum input leakage current is specified in Chapter 33, “Electrical Characteristics”. Input leakage is greater at higher operating temperatures. In the temperature range from 125°C to 50°C, the leakage current is halved for every 8°C to 12°C reduction in temperature.

Assuming $V_{RH}–V_{RL}$ = 5.12 V, 1 count (with 10-bit resolution) corresponds to 5 mV of input voltage. A typical input leakage of 200 nA acting through 10 kΩ of external series resistance results in an error of 0.4 count (2.0 mV). If the source impedance is 100 kΩ and a typical leakage of 100 nA is present, an error of 2 counts (10 mV) is introduced.

In addition to internal junction leakage, external leakage (for example, if external clamping diodes are used) and charge sharing effects with internal capacitors also contribute to the total leakage current. Table 28-25 illustrates the effect of different levels of total leakage on accuracy for different values of source impedance. The error is listed in terms of 10-bit counts.

CAUTION

Leakage below 200 nA is obtainable only within a limited temperature range.

Table 28-25. Error Resulting from Input Leakage (I_{Off})

Source Impedance	Leakage Value (10-Bit Conversions)			
	100 nA	200 nA	500 nA	1000 nA
1 kΩ	—	—	0.1 counts	0.2 counts
10 kΩ	0.2 counts	0.4 counts	1 counts	2 counts
100 kΩ	2 counts	4 count	10 counts	20 counts

28.10 Interrupts

The four interrupt lines are outputs of the module and have no priority or arbitration within the module.

28.10.1 Interrupt Operation

QADC inputs can be monitored by polling or by using interrupts. When interrupts are not needed, the completion flag and the pause flag for each queue can be monitored in the status register (QASR0). In other words, flag bits can be polled to determine when new results are available.

Table 28-26 shows the status flag and interrupt enable bits which correspond to queue 1 and queue 2 activity.

Table 28-26. QADC Status Flags and Interrupt Sources

Queue	Queue Activity	Status Flag	Interrupt Enable Bit
Queue 1	Result written for last CCW in queue 1	CF1	CIE1
	Result written for a CCW with pause bit set in queue 1	PF1	PIE1
Queue 2	Result written for last CCW in queue 2	CF2	CIE2
	Result written for a CCW with pause bit set in queue 2	PF2	PIE2

If interrupts are enabled for an event, the QADC requests interrupt service when the event occurs. Using interrupts does not require continuously polling the status flags to see if an event has taken place; however, status flags must be cleared after an interrupt is serviced, in order to remove the interrupt request

In both polled and interrupt-driven operating modes, status flags must be re-enabled after an event occurs. Flags are re-enabled by clearing the appropriate QASR0 bits in a particular sequence. QASR0 must first be read, then 0s must be written to the flags that are to be cleared. If a new event occurs between the time that the register is read and the time that it is written, the associated flag is not cleared.

28.10.2 Interrupt Sources

The QADC includes four sources of interrupt requests, each of which is separately enabled. Each time the result is written for the last conversion command word (CCW) in a queue, the completion flag for the corresponding queue is set, and when enabled, an interrupt is requested. In the same way, each time the

result is written for a CCW with the pause bit set, the queue pause flag is set, and when enabled, an interrupt is requested. Refer to Table 28-26.

The pause and complete interrupts for queue 1 and queue 2 have separate interrupt vector levels, so that each source can be separately serviced.

Chapter 29
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.

29.1 Features

Module features include:

- Seven sources of reset:
  - External
  - Power-on reset (POR)
  - Watchdog timer
  - 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

29.2 Block Diagram

Figure 29-1 illustrates the reset controller and is explained in the following sections.

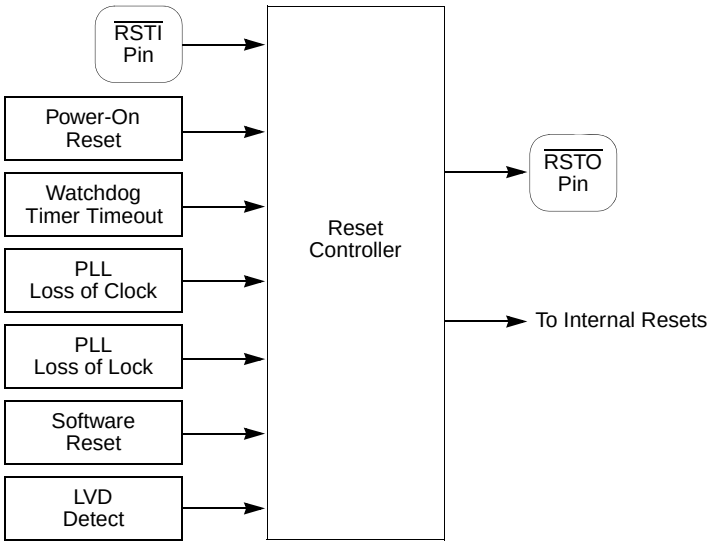


Figure 29-1. Reset Controller Block Diagram

29.3 Signals

Table 29-1 provides a summary of the reset controller signal properties. The signals are described in the following paragraphs.

Table 29-1. 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.

29.3.1 $\overline{\text{RSTI}}$

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

29.3.2 $\overline{\text{RSTO}}$

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.

29.4 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 29-2 for the memory map and the following paragraphs for a description of the registers.

Table 29-2. Reset Controller Memory Map

IPSBAR Offset	Bits 7:0	Access[1]
0x11_0000	RCR	S/U
0x11_0001	RSR	S/U
0x11_0002	Reserved[2]	—
0x11_0003	Reserved[2]	—

1 S/U = supervisor or user mode access.

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

29.4.1 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 29-2. Reset Control Register (RCR)

Table 29-3. 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 $\overline{\text{RSTO}}$ pin. 1 Assert $\overline{\text{RSTO}}$ pin 0 Negate $\overline{\text{RSTO}}$ pin CAUTION: External logic driving reset configuration data during reset needs to be considered when asserting the $\overline{\text{RSTO}}$ 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).

Table 29-3. RCR Field Descriptions (continued)

Bit(s)	Name	Description
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

29.4.2 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	WDR	POR	EXT	LOC	LOL
Reset	Reset Dependent							
R/W	R							
Address	IPSBAR + 0x11_0001							

Figure 29-3. Reset Status Register (RSR)

Table 29-4. 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	WDR	Watchdog timer reset flag. Indicates that the last reset was caused by a watchdog timer timeout. 1 Last reset caused by watchdog timer timeout 0 Last reset not caused by watchdog timer timeout

Table 29-4. RSR Field Descriptions (continued)

Bit(s)	Name	Description
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 $\overline{\text{RSTI}}$ 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

29.5 Functional Description

29.5.1 Reset Sources

Table 29-5 defines the sources of reset and the signals driven by the reset controller.

Table 29-5. Reset Source Summary

Source	Type
Power on	Asynchronous
External $\overline{\text{RSTI}}$ pin (not stop mode)	Synchronous
External $\overline{\text{RSTI}}$ pin (during stop mode)	Asynchronous
Watchdog timer	Synchronous
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.

29.5.1.1 Power-On Reset

At power up, the reset controller asserts $\overline{\text{RSTO}}$. $\overline{\text{RSTO}}$ 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, $\overline{\text{RSTO}}$ is negated and the part begins operation.

29.5.1.2 External Reset

Asserting the external $\overline{\text{RSTI}}$ 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 $\overline{\text{RSTO}}$ for approximately 512 cycles after $\overline{\text{RSTI}}$ 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 $\overline{\text{RSTI}}$ in stop mode causes an external reset to be recognized.

29.5.1.3 Watchdog Timer Reset

A watchdog timer timeout causes timer reset request to be recognized and latched. The bus monitor is enabled and the current bus cycle is completed. 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 begins operation.

29.5.1.4 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 $\overline{\text{RSTO}}$ for approximately 512 cycles after the PLL has acquired lock. The part then exits reset and begins operation.

29.5.1.5 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.

29.5.1.6 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.

29.5.1.7 LVD Reset

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

29.5.2 Reset Control Flow

The reset logic control flow is shown in Figure 29-4. 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.

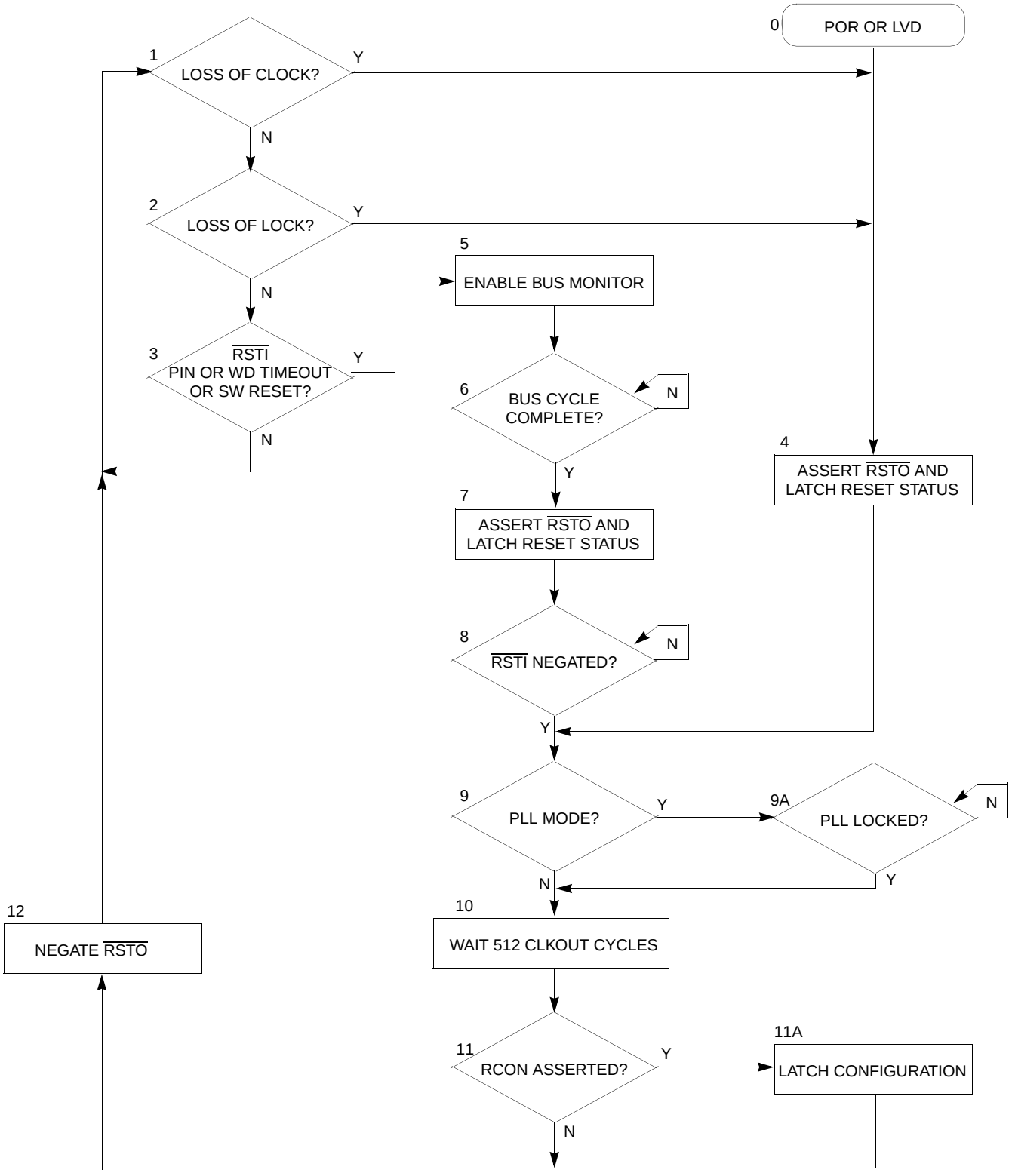


Figure 29-4. Reset Control Flow

29.5.2.1 Synchronous Reset Requests

In this discussion, the reference in parentheses refer to the state numbers in Figure 29-4. All cycle counts given are approximate.

If the external RSTI signal is asserted by an external device for at least four rising CLKOUT edges (3), if the watchdog timer times out, or 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), RSTO is asserted (7). The reset control logic waits until the 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 RCON signal level (11, 11A) before negating RSTO (12).

If the external 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 RSTI signal to be negated before continuing.

29.5.2.2 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 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 RCON pin level (11, 11A) before negating 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.

29.5.2.3 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.

29.5.3 Concurrent Resets

This section describes the concurrent resets. As in the previous discussion references in parentheses refer to the state numbers in Figure 29-4.

29.5.3.1 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 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 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 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).

29.5.3.2 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 $\overline{\text{RSTI}}$ 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 30
Debug Support

This chapter describes the Revision A enhanced hardware debug support.

30.1 Overview

The debug module is shown in Figure 30-1.

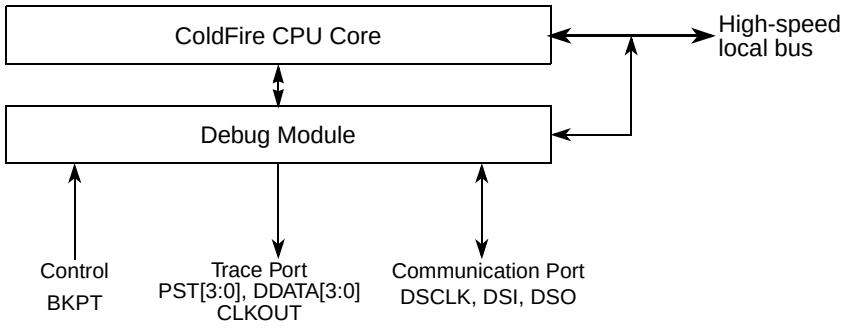


Figure 30-1. Processor/Debug Module Interface

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 30.3, “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 and registers. The external emulator uses a three-pin, serial, full-duplex channel. See Section 30.5, “Background Debug Mode (BDM),” and Section 30.4, “Programming Model.”
- Real-time debug support—BDM requires the processor to be halted, which many real-time embedded applications cannot do. 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. External development systems can access saved data because the hardware supports concurrent operation of the processor and BDM-initiated commands. See Section 30.6, “Real-Time Debug Support.”

NOTE

Enabling Flash security disables BDM communications.

30.2 Signal Description

Table 30-1 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 30.8, "Freescale-Recommended BDM Pinout."

Table 30-1. 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 CLKOUT edges.) Clocks the serial communication port to the debug module during packet transfers. Maximum frequency is 1/5 the processor status clock (CLKOUT) 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 ($\overline{\text{BKPT}}$)	Input used to request a manual breakpoint. Assertion of $\overline{\text{BKPT}}$ 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.
CLKOUT	See Figure 30-2. CLKOUT indicates when the development system should sample PST and DDATA values.
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.
Processor Status (PST[3:0])	These output signals report the processor status. Table 30-2 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.

Figure 30-2 shows CLKOUT timing with respect to PST and DDATA.

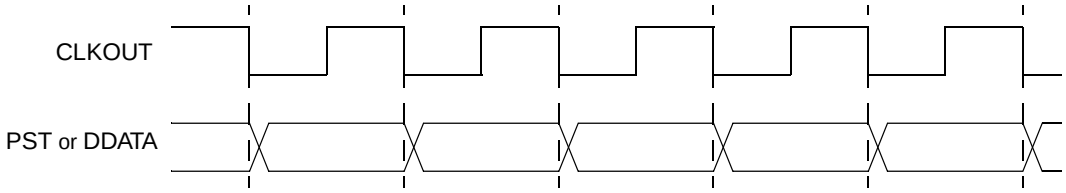


Figure 30-2. CLKOUT Timing

30.3 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, especially when

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 30.3.1, "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 speed is 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 30-2 shows the encoding of these signals.

Table 30-2. 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 30.3.1, "Begin Execution of Taken Branch (PST = 0x5)."
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 CLKOUT 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.
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.

Table 30-2. Processor Status Encoding (continued)

PST[3:0]		Definition
Hex	Binary	
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 30.5.1, "CPU Halt")

30.3.1 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 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 30-3 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).

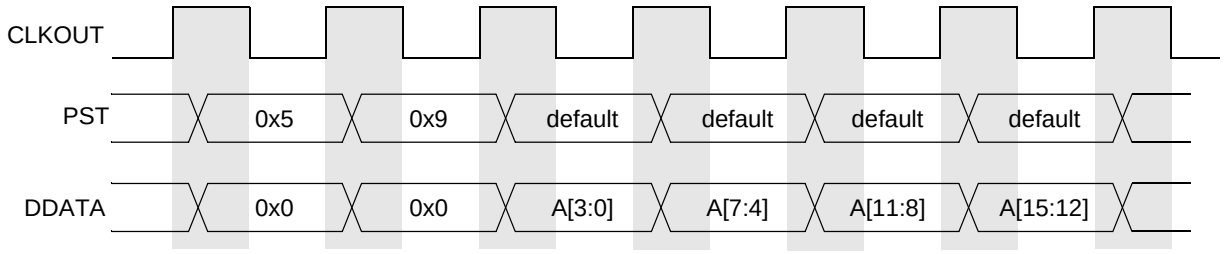


Figure 30-3. Example JMP Instruction Output on PST/DDATA

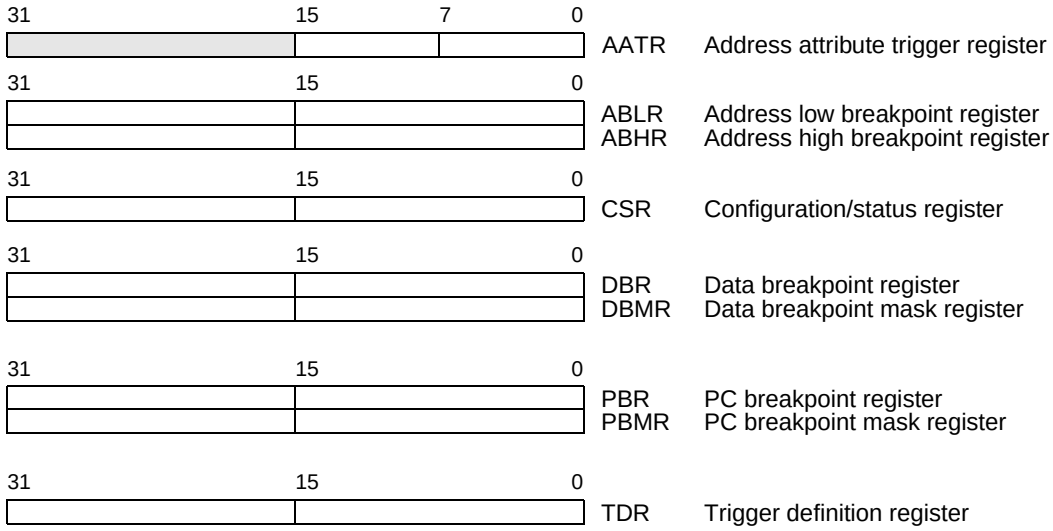
PST 0x5 indicates a taken branch and the marker value 0x9 indicates a 2-byte address. Thus, the subsequent 4 nibbles of DDATA display the lower 2 bytes of address register A0 in least-to-most-significant nibble order. The PST output after the JMP instruction completes depends on the

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.

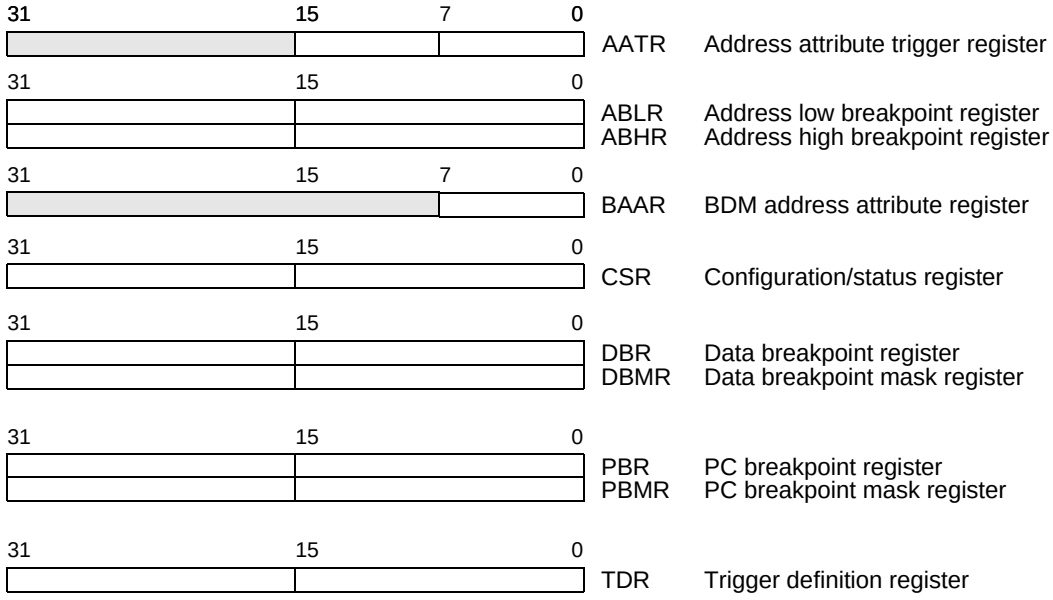
30.4 Programming Model

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 device is using the WDEBUG instruction to access debug module registers or the resulting behavior is undefined.

These registers, shown in Figure 30-4, are treated as 32-bit quantities, regardless of the number of implemented bits.



Note: Each debug register is accessed as a 32-bit register; shaded fields above are not used (don't care). All debug control registers are writable from the external development system or the CPU via 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.



Note: Each debug register is accessed as a 32-bit register; shaded fields above are not used (don't care). All debug control registers are writable from the external development system or the CPU via 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.

Figure 30-4. Debug Programming Model

These registers are accessed through the BDM port by the commands, WDMREG and RDMREG, described in Section 30.5.3.3, "Command Set Descriptions." These commands contain a 5-bit field, DRc, that specifies the register, as shown in Table 30-3.

Table 30-3. BDM/Breakpoint Registers

DRc[4–0]	Register Name	Abbreviation	Initial State	Page
0x00	Configuration/status register	CSR	0x00010_0000	p. 30-10
0x01–0x05	Reserved	—	—	—
0x06	Address attribute trigger register	AATR	0x0000_0005	p. 30-7
0x07	Trigger definition register	TDR	0x0000_0000	p. 30-14
0x08	Program counter breakpoint register	PBR	—	p. 30-13
0x09	Program counter breakpoint mask register	PBMR	—	p. 30-13
0x0A–0x0B	Reserved	—	—	—
0x0C	Address breakpoint high register	ABHR	—	p. 30-9
0x0D	Address breakpoint low register	ABLR	—	p. 30-9
0x0E	Data breakpoint register	DBR	—	p. 30-12
0x0F	Data breakpoint mask register	DBMR	—	p. 30-12

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.

30.4.1 Revision A Shared Debug Resources

In the Revision A implementation of the debug module, certain hardware structures are shared between BDM and breakpoint functionality as shown in Table 30-4.

Table 30-4. Rev. A 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, a BDM command to access memory overwrites an address breakpoint in ABHR. A BDM write command overwrites the data breakpoint in DBR.

30.4.2 Address Attribute Trigger Register (AATR)

The AATR, shown in Figure 30-5, 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).

	15	14 13	12 11	10 8	7	6 5	4 3	2 0
Field	RM	SZM	TTM	TMM	R	SZ	TT	TM
Reset	0000_0000_0000_0101							
R/W	Write only. 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)							

Figure 30-5. Address Attribute Trigger Register (AATR)

Table 30-5 describes AATR fields.

Table 30-5. AATR Field Descriptions

Bits	Name	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.
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 indicates an external or DMA access (for backward compatibility). These bits affect the TM bits.

Table 30-5. AATR Field Descriptions (continued)

Bits	Name	Description
2–0	TM	Transfer modifier. Compared with the local bus transfer modifier signals, which give supplemental information for each transfer type. TT = 00 (normal mode): 000 Explicit cache line push 001 User data access 010 User code access 011 Reserved 100 Reserved 101 Supervisor data access 110 Supervisor code access 111 Reserved TT = 10 (emulator mode): 0xx–100 Reserved 101 Emulator mode data access 110 Emulator mode code access 111 Reserved TT = 11 (acknowledge/CPU space transfers): 000 CPU space access 001–111 Interrupt acknowledge levels 1–7 These bits also define the TM encoding for BDM memory commands (for backward compatibility).

30.4.3 Address Breakpoint Registers (ABLR, ABHR)

The ABLR and ABHR, shown in Figure 30-6, 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

	31 … 0
Field	Address
Reset	—
R/W	Write only. 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]	0x0D (ABLR); 0x0C (ABHR)

Figure 30-6. Address Breakpoint Registers (ABLR, ABHR)

Table 30-6 describes ABLR fields.

Table 30-6. ABLR Field Description

Bits	Name	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 30-7 describes ABHR fields.

Table 30-7. ABHR Field Description

Bits	Name	Description
31–0	Address	High address. Holds the 32-bit address marking the upper bound of the address breakpoint range.

30.4.4 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.

	31–28	27	26	25	24	23–20	19–17	16
Field	BSTAT	FOF	TRG	HALT	BKPT	HRL	—	IPW
Reset	0000_0000_0000_0000							
R/W[1]	R						—	R/W

	15	14	13	12–11	10	9–8	7	6	5	4	3–0
Field	MAP	TRC	EMU	DDC	UHE	BTB	—	NPL	IPI	SSM	—
Reset	0000_0000_0000_0000										
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	—
DRc[4–0]	0000_0000_0000_0000										

Figure 30-7. Configuration/Status Register (CSR)

Table 30-8 describes CSR fields.

Table 30-8. CSR Field Descriptions

Bit	Name	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. 0000 Initial debug functionality (Revision A) (This is the only valid value for this device.)
19–17	—	Reserved, should be cleared.
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 30.6.1.1, "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 30-2. 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 30-8. CSR Field Descriptions (continued)

Bit	Name	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 30.3.1, "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.

30.4.5 Data Breakpoint/Mask Registers (DBR, DBMR)

The DBR, shown in Figure 30-8, specifies data patterns used as part of the trigger into debug mode. DBR bits are masked by setting corresponding DBMR bits, as defined in TDR.

	31 … 0
Field	Data (DBR); Mask (DBMR)
Reset	Uninitialized
R/W	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), 0x0F (DBMR)

Figure 30-8. Data Breakpoint/Mask Registers (DBR/DBMR)

Table 30-9 describes DBR fields.

Table 30-9. DBR Field Descriptions

Bits	Name	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 30-10 describes DBMR fields.

Table 30-10. DBMR Field Descriptions

Bits	Name	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 30-11 shows relationships between processor address, access size, and location within the 32-bit data bus.

Table 30-11. 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]

30.4.6 Program Counter Breakpoint/Mask Registers (PBR, PBMR)

The PBR 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 30-9 shows the PC breakpoint register.

	31	0
Field	Program Counter	
Reset	—	
R/W	Write. 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 30.5.3.3, “Command Set Descriptions.”	
DRc[4–0]	0x08 (PBR)	

Figure 30-9. Program Counter Breakpoint Register (PBR)

Table 30-12 describes PBR fields.

Table 30-12. PBR Field Descriptions

Bits	Name	Description
31–0	Address	PC breakpoint address. The 32-bit address to be compared with the PC as a breakpoint trigger.

Figure 30-9 shows PBMR.

	31	0
Field	Mask	
Reset	—	
R/W	Write. 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	

Figure 30-10. Program Counter Breakpoint Mask Register (PBMR)

Table 30-13 describes PBMR fields.

Table 30-13. PBMR Field Descriptions

Bits	Name	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. Set PBMR bits cause PBR bits to be ignored.

30.4.7 Trigger Definition Register (TDR)

The TDR, shown in Table 30-11, 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].

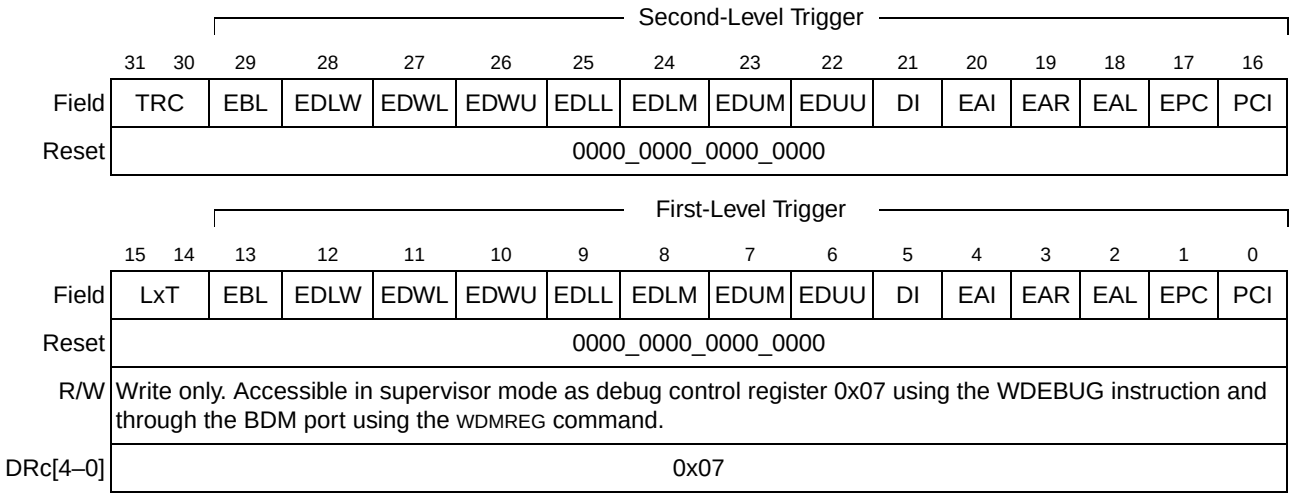


Figure 30-11. Trigger Definition Register (TDR)

Table 30-14 describes TDR fields.

Table 30-14. TDR Field Descriptions

Bits	Name	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
15–14	LxT	Level-x trigger. This is a Rev. B function only. The Level-x Trigger bit 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. TDR[15] 0 Level-2 trigger = PC_condition & Address_range & Data_condition 1 Level-2 trigger = PC_condition | (Address_range & Data_condition) TDR[14] 0 Level-1 trigger = PC_condition & Address_range & Data_condition 1 Level-1 trigger = PC_condition | (Address_range & Data_condition)
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 30-14. TDR Field Descriptions (continued)

Bits	Name	Description	
28–22/ 12–6	ED*x*	Setting an ED*x* bit enables the corresponding data breakpoint condition based on the size and placement on the processor's local data bus. Clearing all ED*x* bits disables data breakpoints.	
28/12		EDLW	Data longword. Entire processor's local data bus.
27/11		EDWL	Lower data word.
26/10		EDWU	Upper data word.
25/9		EDLL	Lower lower data byte. Low-order byte of the low-order word.
24/8		EDLM	Lower middle data byte. High-order byte of the low-order word.
23/7		EDUM	Upper middle data byte. Low-order byte of the high-order word.
22/6		EDUU	Upper upper data byte. High-order byte of the high-order word.
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	EA*x*	Enable address bits. Setting an EA bit enables the corresponding address breakpoint. Clearing all three bits disables the breakpoint.	
20/4		EAI	Enable address breakpoint inverted. Breakpoint is based outside the range between ABLR and ABHR.
19/3		EAR	Enable address breakpoint range. The breakpoint is based on the inclusive range defined by ABLR and ABHR.
18/2		EAL	Enable address breakpoint low. The breakpoint is based on the address in the ABLR.
17/1	EPC	Enable PC breakpoint. If set, this bit enables the 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.	

30.5 Background Debug Mode (BDM)

The ColdFire Family implements a low-level system debugger in the microprocessor hardware. Communication with the development system is handled through a dedicated, high-speed serial command interface. The ColdFire architecture implements the BDM controller in a dedicated hardware module. 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.

30.5.1 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 30.6.1, "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.

CSR[27–24] indicates the halt source, showing the highest priority source for multiple halt conditions.

30.5.2 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 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 30-1. 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 CLKOUT 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 30-12, all state transitions are enabled on a rising edge of CLKOUT when DSCLK is high; that is, DSI is sampled and DSO is driven.

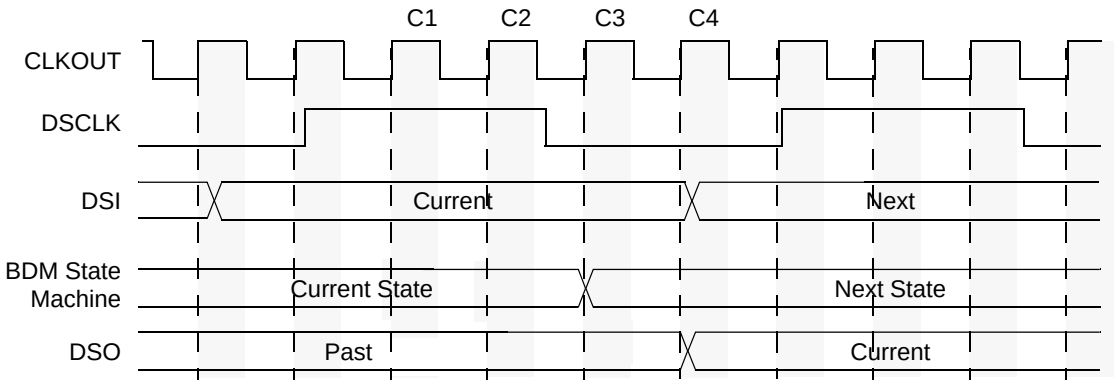


Figure 30-12. BDM Serial Interface Timing

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, DSDO 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.

30.5.2.1 Receive Packet Format

The basic receive packet, Figure 30-13, consists of 16 data bits and 1 status bit

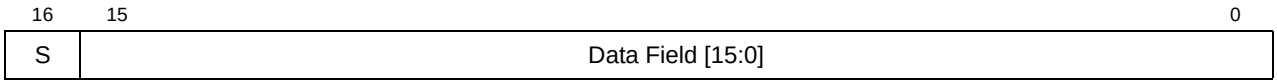


Figure 30-13. Receive BDM Packet

Table 30-15 describes receive BDM packet fields.

Table 30-15. Receive BDM Packet Field Description

Bits	Name	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. S DataMessage 0 xxxxValid data transfer 0 0xFFFFStatus OK 1 0x0000Not ready with response; come again 1 0x0001Error—Terminated bus cycle; data invalid 1 0xFFFFIllegal command
15–0	D	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.

30.5.2.2 Transmit Packet Format

The basic transmit packet, Figure 30-14, consists of 16 data bits and 1 control bit.

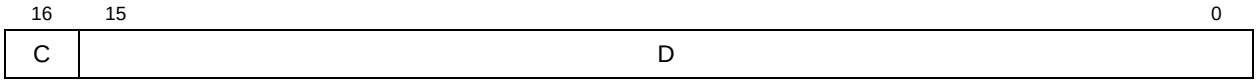


Figure 30-14. Transmit BDM Packet

Table 30-16 describes transmit BDM packet fields.

Table 30-16. Transmit BDM Packet Field Description

Bits	Name	Description
16	C	Control. This bit is reserved. Command and data transfers initiated by the development system should clear C.
15–0	D	Data bits 15–0. Contains the data to be sent from the development system to the debug module.

30.5.3 BDM Command Set

Table 30-17 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 30-17. 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	30.5.3.3.1	0x218 {A/D, Reg[2:0]}
Write A/D register	WAREG/ WDREG	Write the data operand to the specified address or data register.	Halted	30.5.3.3.2	0x208 {A/D, Reg[2:0]}
Read memory location	READ	Read the data at the memory location specified by the longword address.	Steal	30.5.3.3.3	0x1900—byte 0x1940—word 0x1980—longword
Write memory location	WRITE	Write the operand data to the memory location specified by the longword address.	Steal	30.5.3.3.4	0x1800—byte 0x1840—word 0x1880—longword
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	30.5.3.3.5	0x1D00—byte 0x1D40—word 0x1D80—longword
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	30.5.3.3.6	0x1C00—byte 0x1C40—word 0x1C80—longword
Resume execution	GO	The pipeline is flushed and refilled before resuming instruction execution at the current PC.	Halted	30.5.3.3.7	0x0C00
No operation	NOP	Perform no operation; may be used as a null command.	Parallel	30.5.3.3.8	0x0000
Read control register	RCREG	Read the system control register.	Halted	30.5.3.3.9	0x2980
Write control register	WCREG	Write the operand data to the system control register.	Halted	30.5.3.3.10	0x2880
Read debug module register	RDMREG	Read the debug module register.	Parallel	30.5.3.3.11	0x2D {0x4[2] DRc[4:0]}
Write debug module register	WDMREG	Write the operand data to the debug module register.	Parallel	30.5.3.3.12	0x2C {0x4[2] 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 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**.**

30.5.3.1 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 30-15.

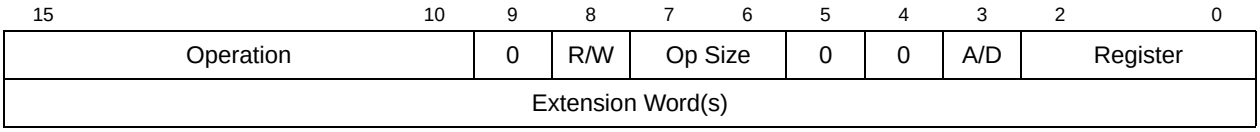


Figure 30-15. BDM Command Format

Table 30-18 describes BDM fields.

Table 30-18. BDM Field Descriptions

Bit	Name	Description
15–10	Operation	Specifies the command. These values are listed in Table 30-17.
9	0	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. Operand Size Bit Values 00 Byte 8 bits 01 Word 16 bits 10 Longword 32 bits 11 Reserved —
5–4	00	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.

30.5.3.1.1 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.

30.5.3.2 Command Sequence Diagrams

The command sequence diagram in Figure 30-16 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.

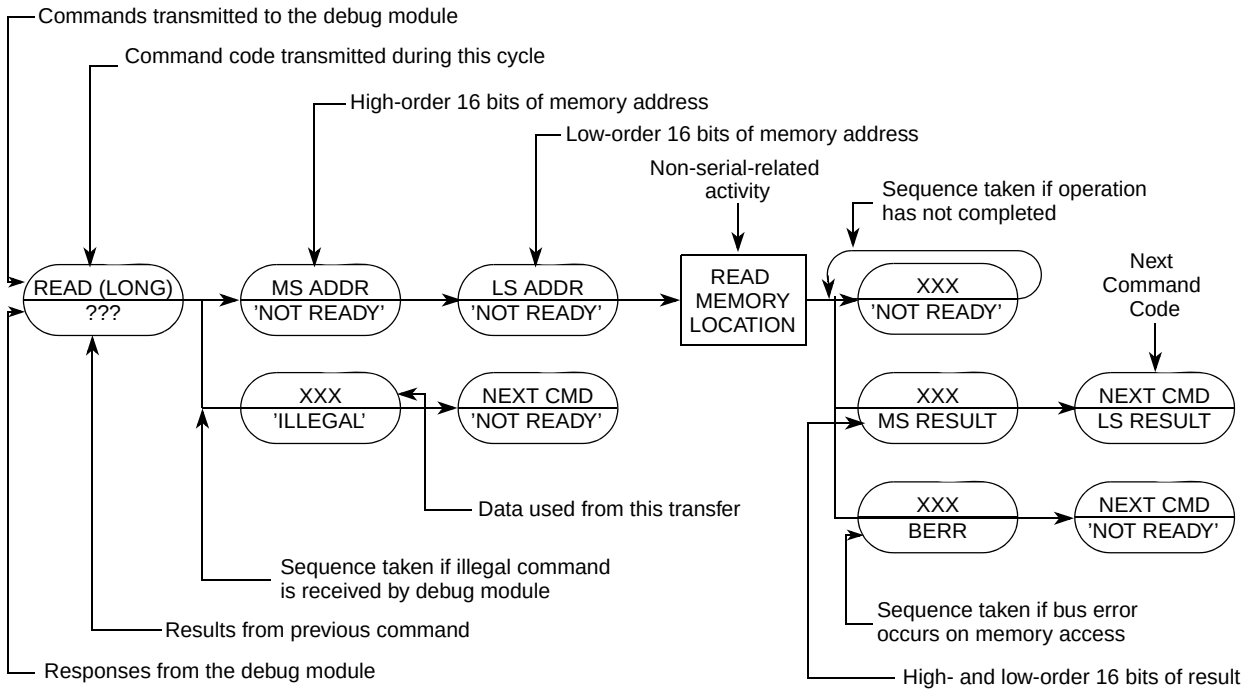


Figure 30-16. Command Sequence Diagram

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.

30.5.3.3 Command Set Descriptions

The following sections describe the commands summarized in Table 30-17.

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 30.5.2, "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.

30.5.3.3.1 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:

	15–12	11–8	7–4	3	2–0
Command	0x2	0x1	0x8	A/D	Register
Result	D[31:16]				
	D[15:0]				

Figure 30-17. RAREG/RDREG Command Format

Command Sequence:

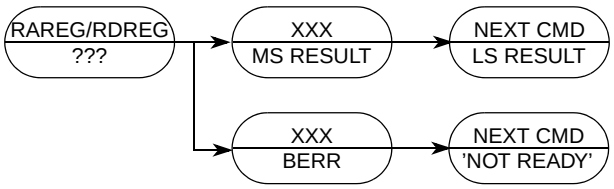


Figure 30-18. RAREG/RDREG Command Sequence

Operand Data: None

Result Data: The contents of the selected register are returned as a longword value, most-significant word first.

30.5.3.3.2 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:

15–12	11–8	7–4	3	2–0
0x2	0x0	0x8	A/D	Register
D[31:16]				
D[15:0]				

Figure 30-19. WAREG/WDREG Command Format

Command Sequence

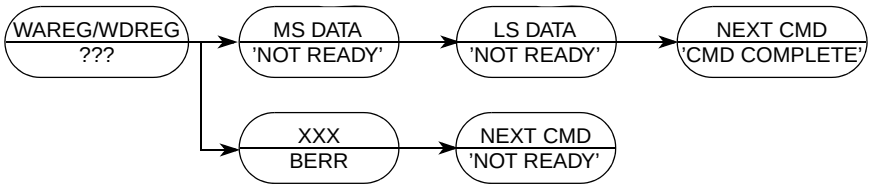


Figure 30-20. WAREG/WDREG Command Sequence

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.

30.5.3.3.3 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 12	11 8	7 4	3 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 30-21. READ Command/Result Formats

Command Sequence:

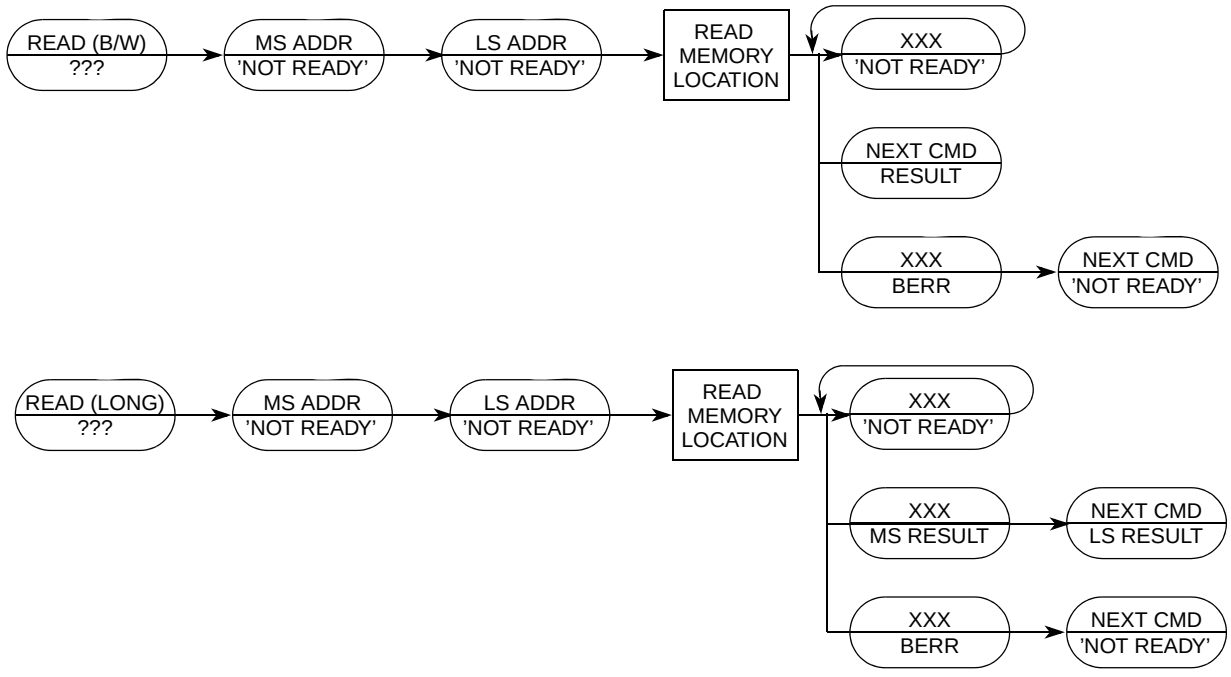


Figure 30-22. READ Command Sequence

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.

30.5.3.3.4 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:

	15–12	11–8	7–4	3–1
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 30-23. WRITE Command Format

Command Sequence:

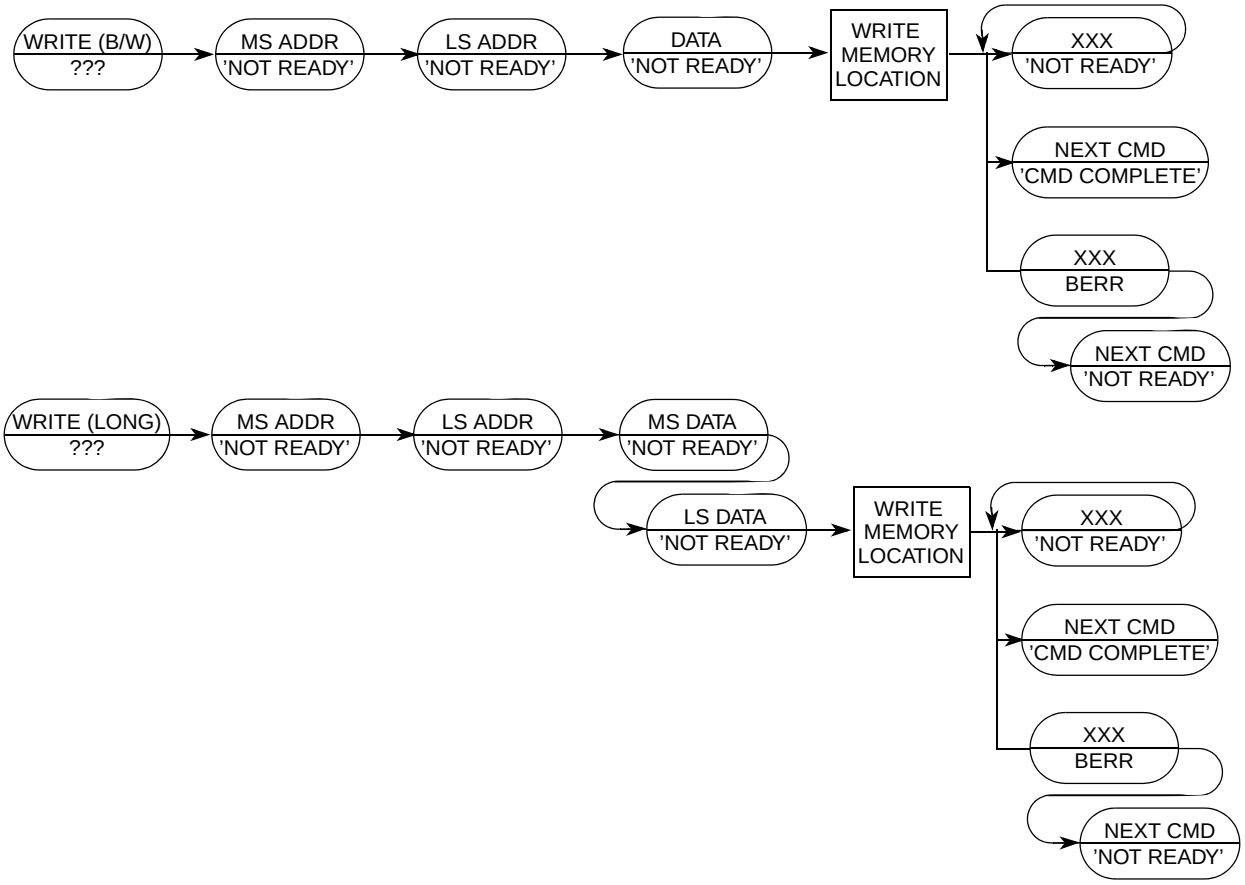


Figure 30-24. WRITE Command Sequence

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.

30.5.3.3.5 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 30-25. DUMP Command/Result Formats

Command Sequence:

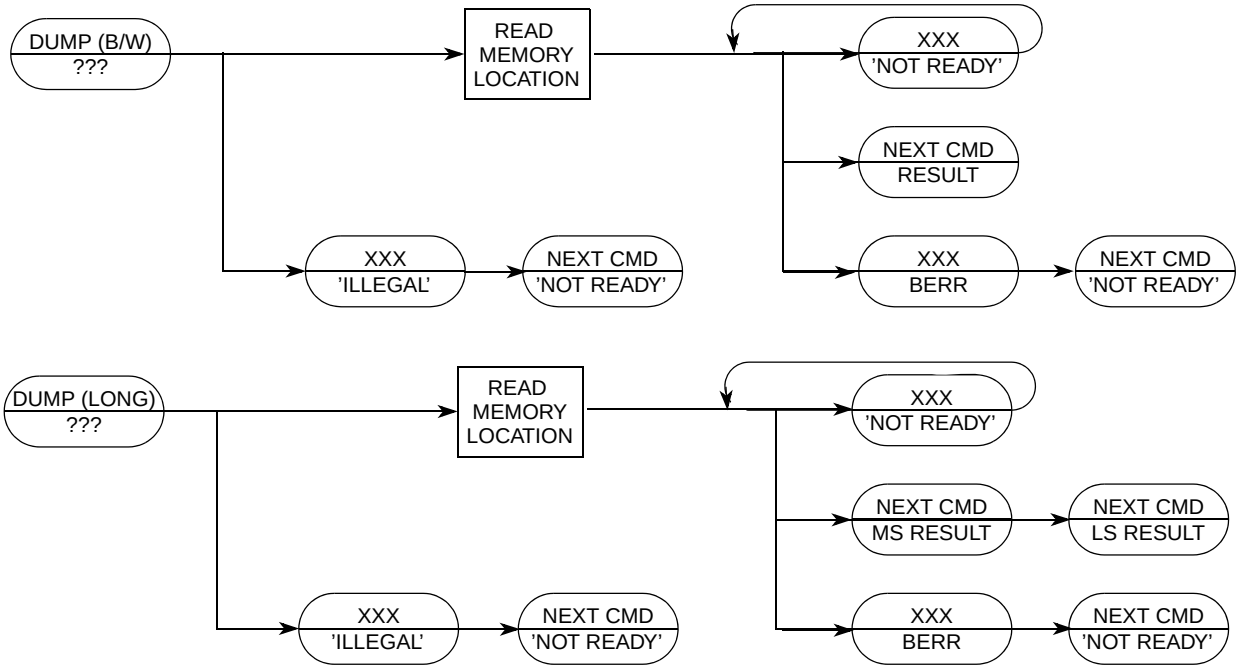


Figure 30-26. DUMP Command Sequence

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.

30.5.3.3.6 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 12	11 8	7 4	3 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 30-27. FILL Command Format

Command Sequence:

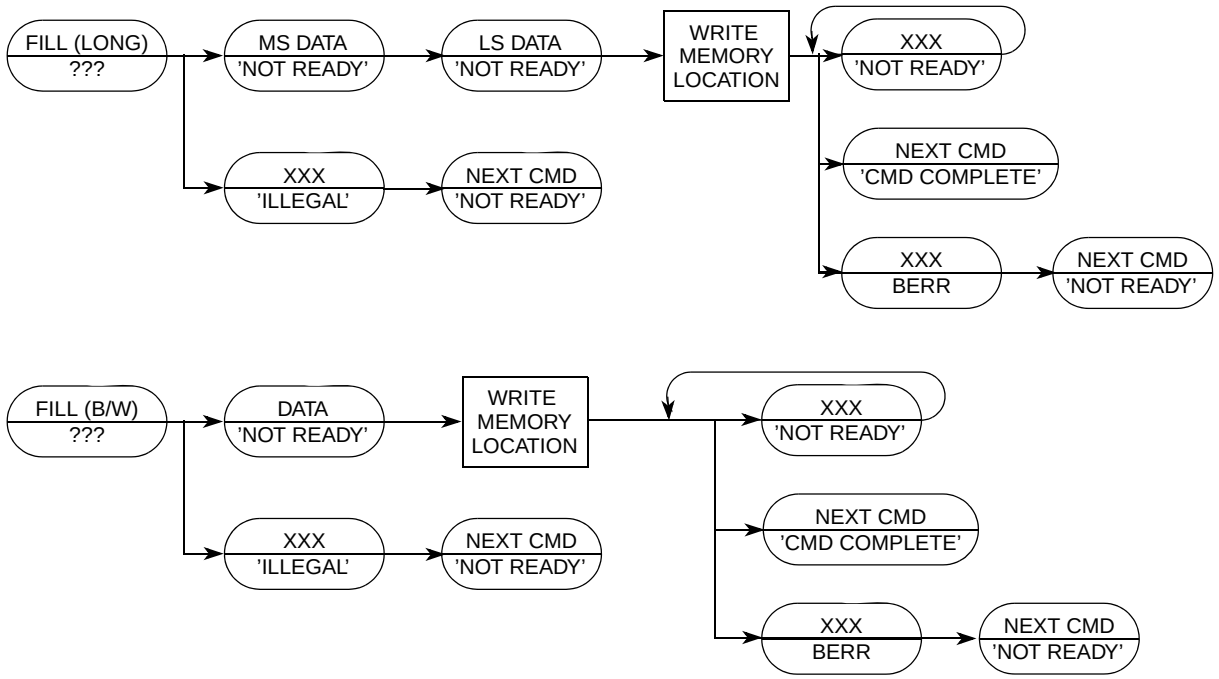


Figure 30-28. FILL Command Sequence

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.

30.5.3.3.7 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.

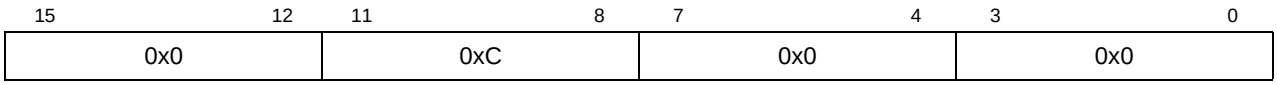


Figure 30-29. GO Command Format

Command Sequence:

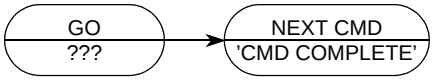


Figure 30-30. GO Command Sequence

Operand Data: None

Result Data: The command-complete response (0xFFFF) is returned during the next shift operation.

30.5.3.3.8 No Operation (NOP)

NOP performs no operation and may be used as a null command where required.

Command Formats:

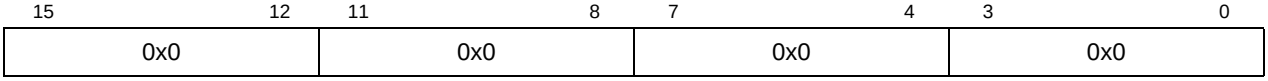


Figure 30-31. NOP Command Format

Command Sequence:

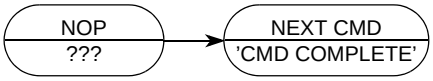


Figure 30-32. NOP Command Sequence

Operand Data: None

Result Data: The command-complete response, 0xFFFF (with S cleared), is returned during the next shift operation.

30.5.3.3.9 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 12	11 8	7 4	3 0
Command	0x2	0x9	0x8	0x0
	0x0	0x0	0x0	0x0
	0x0	Rc		
Result	D[31:16]			
	D[15:0]			

Figure 30-33. RCREG Command/Result Formats

Rc encoding:

Table 30-19. Control Register Map

Rc	Register Definition
0x002	Cache Control Register (CACR)
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)
0x807	MAC Accumulator 0,1 Extension Bytes (ACCEXT01)
0x808	MAC Accumulator 2,3 Extension Bytes (ACCEXT23)
0x809	MAC Accumulator 1 (ACC1)
0x80A	MAC Accumulator 2 (ACC2)
0x80B	MAC Accumulator 3 (ACC3)
0x80E	Status Register (SR)
0x80F	Program Register (PC)
0xC04	Flash Base Address Register 0 (FLASHBAR) (Not on MCF5280)
0xC05	RAM Base Address Register (RAMBAR)

Command Sequence:

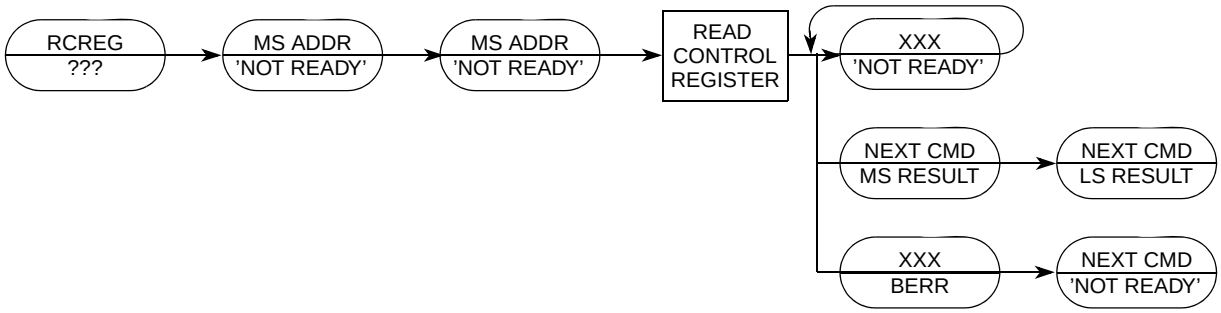


Figure 30-34. RCREG Command Sequence

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 processor's Version 2 ColdFire 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.

BDM Accesses of the EMAC Registers

The presence of rounding logic in the output datapath of the EMAC requires that special care be taken during any BDM-initiated reads and writes of its programming model. In particular, it is necessary that any result rounding modes be disabled during the read/write process so the exact bit-wise contents of the EMAC registers are accessed.

As an example, any BDM read of an accumulator register (ACCn) must be preceded by two commands accessing the MAC status register. Specifically, the following BDM sequence is required:

```
BdmReadACCn (

    rcreg    macsr;            // read macsr contents & save

    wcreg    #0,macsr;         // disable all rounding modes
```

```
    rcreg    accn;             // read the desired accumulator
    wcreg    #saved_data,macsr;// restore the original macsr
)
```

Likewise, the following BDM sequence is needed to write an accumulator register:

```
BdmWriteACCn (
    rcreg    macsr;            // read macsr contents & save
    wcreg    #0,macsr;         // disable all rounding modes
    wcreg    #data,accn;       // write the desired accumulator
    wcreg    #saved_data,macsr;// restore the original macsr
)
```

Additionally, it is required that writes to the accumulator extension registers be performed after the corresponding accumulators are updated. This is needed since a write to any accumulator alters the contents of the corresponding extension register.

For more information on saving and restoring the complete EMAC programming model, see Section 3.3.1.2, “Saving and Restoring the EMAC Programming Model”.

30.5.3.3.10 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:

	15–12	11–8	7–4	3–0
Command	0x2	0x8	0x8	0x0
	0x0	0x0	0x0	0x0
	0x0	Rc		
Result	D[31:16]			
	D[15:0]			

Figure 30-35. WCREG Command/Result Formats

Command Sequence:

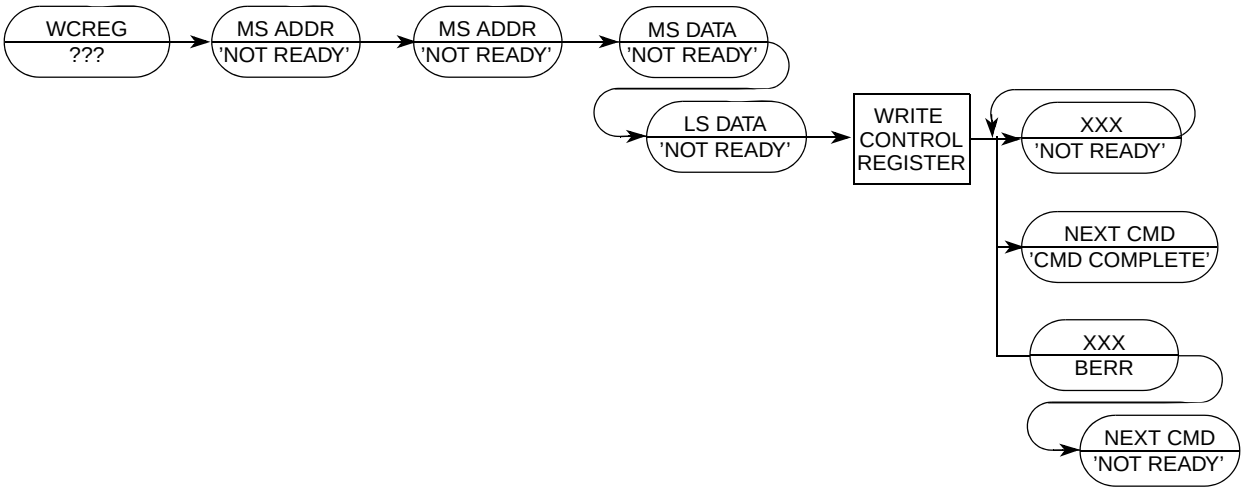


Figure 30-36. WCREG Command Sequence

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.

30.5.3.3.11 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:

	15–12	11–8	7–5	4–0
Command	0x2	0xD	100	DRc
Result	D[31:16]			
	D[15:0]			

Figure 30-37. RDMREG Command/Result Formats

Table 30-20 shows the definition of DRc encoding.

Table 30-20. Definition of DRc Encoding—Read

DRc[4:0]	Debug Register Definition	Mnemonic	Initial State	Page
0x00	Configuration/Status	CSR	0x0	p. 30-10
0x01–0x1F	Reserved	—	—	—

Command Sequence:

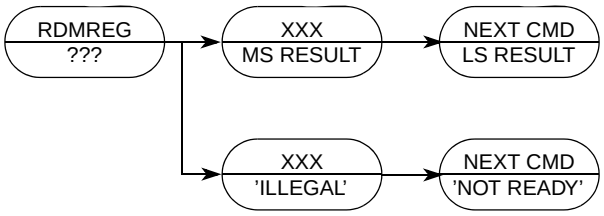


Figure 30-38. RDMREG Command Sequence

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.

30.5.3.3.12 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:

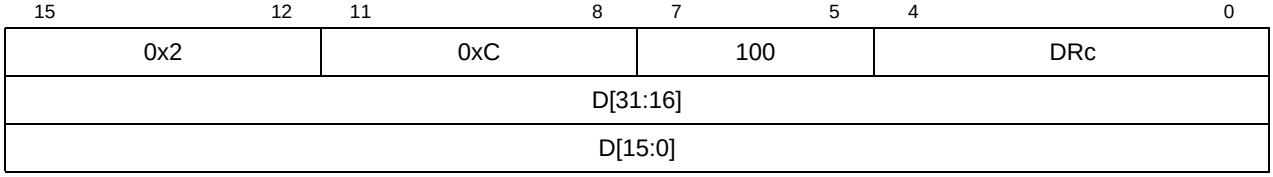


Figure 30-39. WDMREG BDM Command Format

Table 30-3 shows the definition of the DRc write encoding.

Command Sequence:

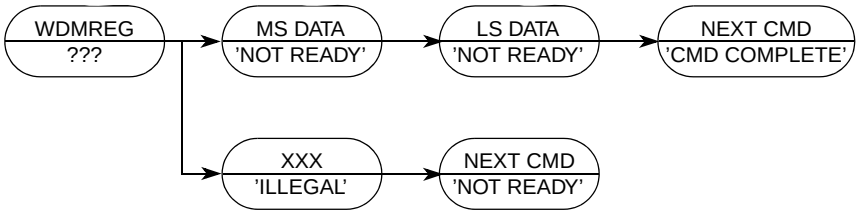


Figure 30-40. WDMREG Command Sequence

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.

30.6 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 three types of breakpoints—PC with 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.

30.6.1 Theory of Operation

Breakpoint hardware can be configured to respond to triggers in several ways. The response desired is programmed into TDR. As shown in Table 30-21, 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 30-21. DDATA[3:0]/CSR[BSTAT] Breakpoint Response

DDATA[3:0]/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 this 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.

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.

30.6.1.1 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 30.5.1, “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).

30.6.2 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).

Note that the debug module requires the use of the internal bus to perform BDM commands. In Revision A, if the processor is executing a tight loop that is contained within a single aligned longword, the processor may never grant the internal bus to the debug module, for example:

```
        align4
label1: nop
        bra.b label1
```

or

```
        align4
label2: bra.w label2
```

The processor grants the internal bus if these loops are forced across two longwords.

30.7 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}.

30.7.1 User Instruction Set

Table 30-22 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 30-22. 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

Table 30-22. PST/DDATA Specification for User-Mode Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
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}
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

Table 30-22. PST/DDATA Specification for User-Mode Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
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
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}

Table 30-22. PST/DDATA Specification for User-Mode Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
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}
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 30-23 shows the PST/DDATA specification for multiply-accumulate instructions.

Table 30-23. 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
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}

30.7.2 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 30-24.

Table 30-24. 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

Table 30-24. PST/DDATA Specification for Supervisor-Mode Instructions

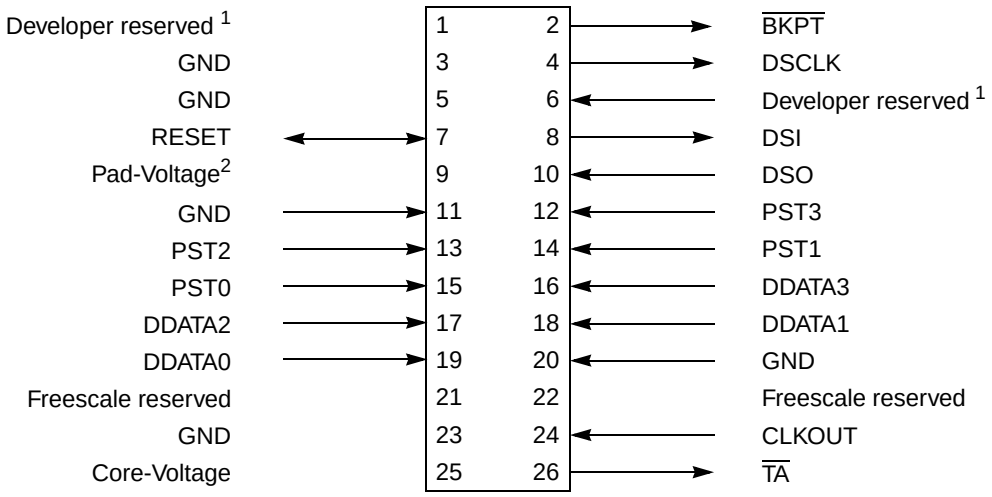
Instruction	Operand Syntax	PST/DDATA
rte		PST = 0x7, {PST = 0xB, DD = source operand}, {PST = 3},{ PST =0xB, DD =source operand}, PST = 0x5, {[PST = 0x9AB], DD = target address}
stldsr.w	#imm	PST = 0x1, {PST = 0xA, DD = destination operand, PST = 0x3}
stop	#imm	PST = 0x1, PST = 0xE
wdebug	<ea>y	PST = 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.

30.8 Freescale-Recommended BDM Pinout

The ColdFire BDM connector, Figure 30-41, is a 26-pin Berg connector arranged 2 x 13.



[1]Pins reserved for BDM developer use.
[2]Supplied by target

Figure 30-41. Recommended BDM Connector

Chapter 31
IEEE 1149.1 Test Access Port (JTAG)

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, TRST.

Figure 31-1 shows the block diagram of the JTAG module.

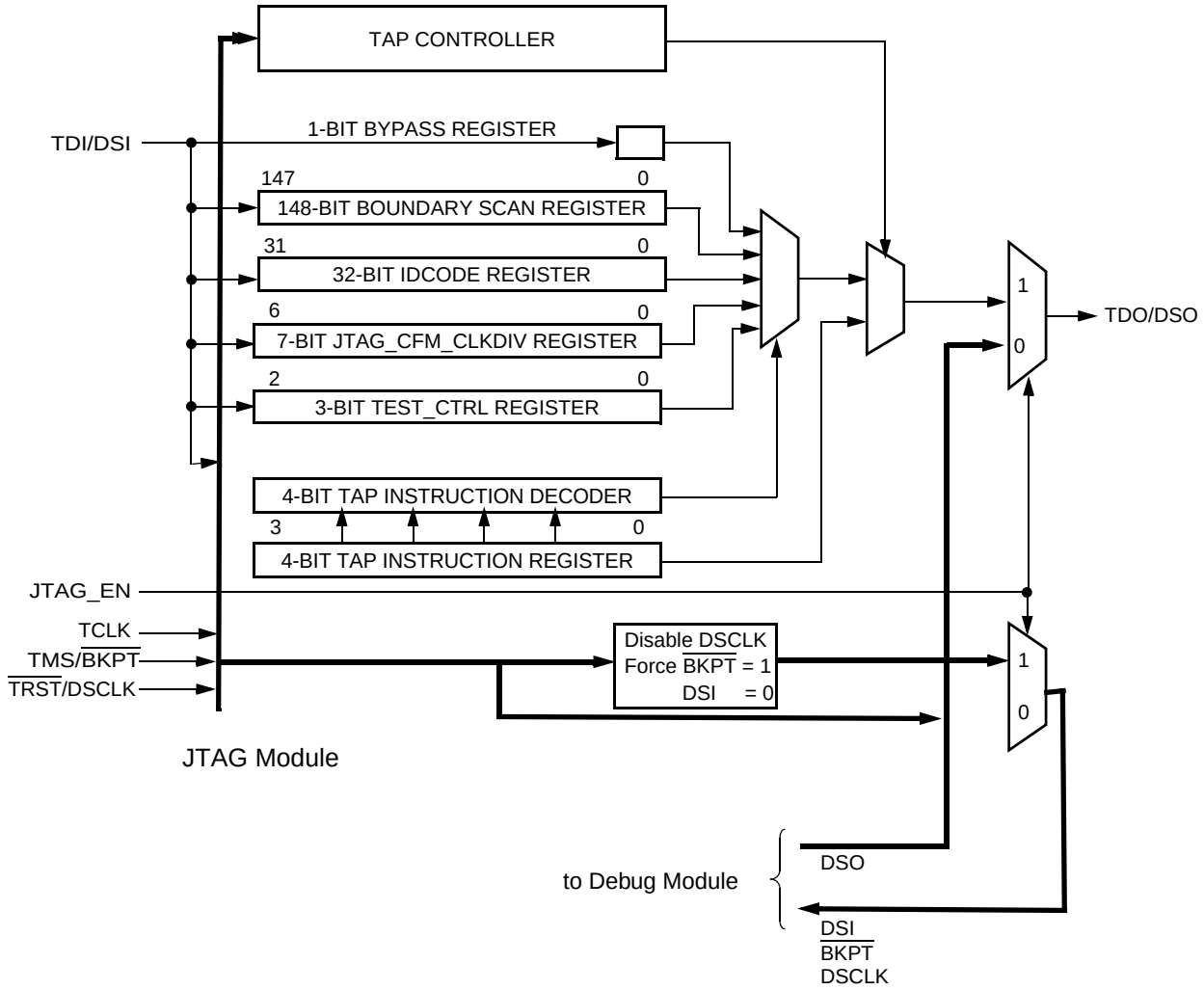


Figure 31-1. JTAG Block Diagram

31.1 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

31.2 Modes of Operation

The JTAG_EN pin can select between the following modes of operation:

- JTAG mode
- BDM - background debug mode (For more information, refer to Section 30.5, “Background Debug Mode (BDM)).”

31.3 External Signal Description

The JTAG module has five input and one output external signals, as described in Table 31-1.

31.3.1 Detailed Signal Description

Table 31-1. 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	—

31.3.1.1 JTAG_EN — JTAG Enable

The JTAG_EN pin selects between Debug module and JTAG. If JTAG_EN is low, the Debug module is selected; if it is high, the JTAG is selected. Table 31-2 summarizes the pin function selected depending upon JTAG_EN logic state.

Table 31-2. Pin Function Selected

	JTAG_EN = 0	JTAG_EN = 1	Pin Name

Table 31-2. Pin Function Selected

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 31-3, in order to disable the corresponding module.

Table 31-3. 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.

31.3.1.2 TCLK — Test Clock Input

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.

31.3.1.3 TMS/$\overline{\text{BKPT}}$ — Test Mode Select / Breakpoint

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.

31.3.1.4 TDI/DSI — Test Data Input / Development Serial Input

The TDI pin is the LSB-first data and instruction input. TDI is sampled on the rising edge of TCLK. The TDI pin has an internal pull-up resistor.

The DSI pin provides data input for the debug module serial communication port.

31.3.1.5 $\overline{\text{TRST}}$/DSCLK — Test Reset / Development Serial Clock

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.

31.3.1.6 TDO/DSO — Test Data Output / Development Serial Output

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.

31.4 Memory Map/Register Definition

31.4.1 Memory Map

The JTAG module registers are not memory mapped and are only accessible through the TDO/DSO pin.

31.4.2 Register Descriptions

All registers are shift-in and parallel load.

31.4.2.1 Instruction Shift Register (IR)

The JTAG module uses a 4-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.

31.4.2.2 IDCODE Register

The IDCODE is a read-only register; its value is chip dependent. For more information, see Section 31.5.3.2, “IDCODE Instruction.”

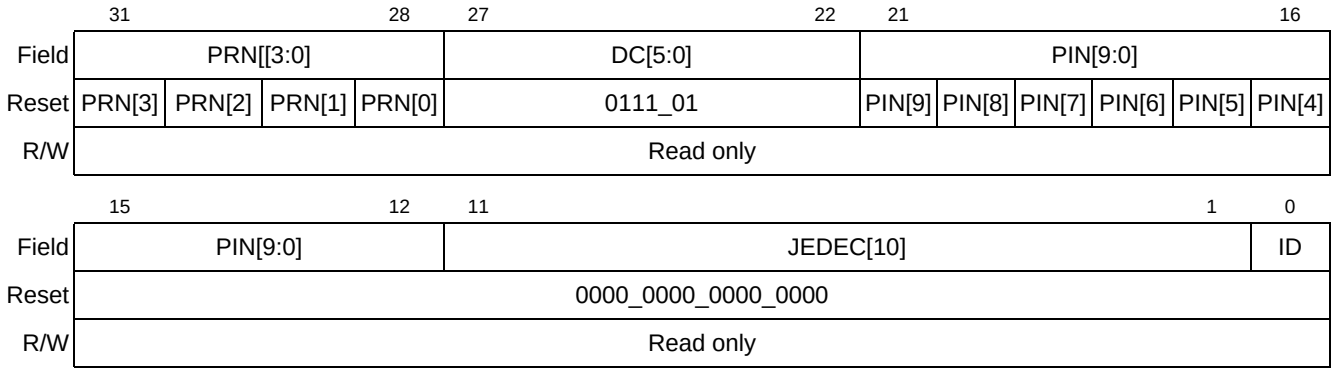


Figure 31-2. IDCODE Register

Table 31-4. IDCODE Register Field Descriptions

Bits	Name	Description
31–28	PRN	Part revision number. Indicate the revision number of the project.
27–22	DC	Design center.
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.

31.4.2.3 Bypass Register

The bypass register is a single-bit shift register path from TDI to TDO when the BYPASS instruction is selected.

31.4.2.4 JTAG_CFM_CLKDIV Register

The JTAG_CFM_CLKDIV register is a 7-bit clock divider for the CFM that is used with the LOCKOUT_RECOVERY instruction. It controls the period of the clock used for timed events in the CFM erase algorithm. The JTAG_CFM_CLKDIV register must be loaded before the lockout sequence can begin.

31.4.2.5 TEST_CTRL Register

The TEST_CTRL register is a 3-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.

31.4.2.6 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, and clock signals.

31.5 Functional Description

31.5.1 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.

31.5.2 TAP Controller

The TAP controller is a state machine that changes state based on the sequence of logical values on the TMS pin. Figure 31-3 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 31-3 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.

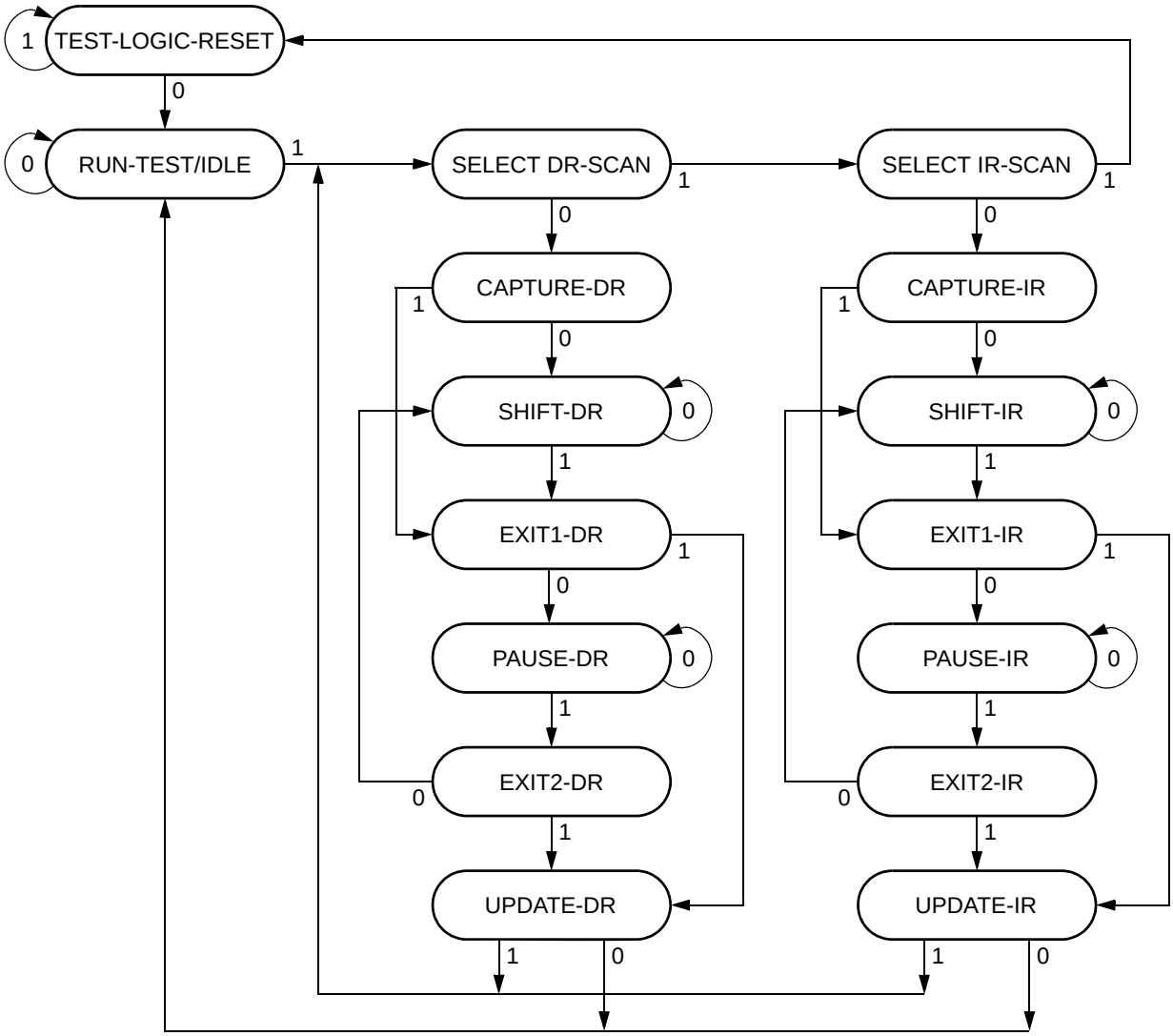


Figure 31-3. TAP Controller State Machine Flow

31.5.3 JTAG Instructions

Table 31-5 describes public and private instructions.

Table 31-5. JTAG Instructions

Instruction	IR[3:0]	Instruction Summary
EXTEST	0000	Selects boundary scan register while applying fixed values to output pins and asserting functional reset
IDCODE	0001	Selects IDCODE register for shift
SAMPLE/PRELOAD	0010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
TEST_LEAKAGE[1,2]	0101	Selects bypass register while tri-stating all output pins and assert to high the jtag_leakage signal
ENABLE_TEST_CTRL	0110	Selects TEST_CTRL register
HIGHZ	1001	Selects bypass register while tri-stating all output pins and asserting functional reset
LOCKOUT_RECOVERY	1011	Allows for the erase of the TFM flash when the part is secure
CLAMP	1100	Selects bypass while applying fixed values to output pins and asserting functional reset
BYPASS	1111	Selects bypass register for data operations
Reserved	all others	Decoded to select bypass register[3]

[1]Instruction for manufacturing purposes only
[2]TRST pin assertion or power-on reset is required to exit this instruction.
[3]Freescale reserves the right to change the decoding of the unused opcodes in the future.

31.5.3.1 External Test Instruction (EXTEST)

The 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.

31.5.3.2 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.

31.5.3.3 SAMPLE/PRELOAD Instruction

The SAMPLE/PRELOAD instruction has two functions:

- SAMPLE - obtain a sample of the system data and control signals present at the MCU input pins and just before the boundary scan cell at the output pins. This sampling occurs on the rising edge of TCLK in the capture-DR state when the IR contains the $2 opcode. The sampled data is

accessible by shifting it through the boundary scan register to the TDO output by using the shift-DR state. Both the data capture and the shift operation are transparent to system operation.

NOTE

External synchronization is required to achieve meaningful results because there is no internal synchronization between TCLK and the system clock.

- 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.

31.5.3.4 TEST_LEAKAGE Instruction

The TEST_LEAKAGE instruction forces the jtag_leakage output signal to high. It is intended to tri-state all output pad buffers and disable all of the part's pad input buffers except TEST and TRST. The jtag_leakage signal is asserted at the rising edge of TCLK when the TAP controller transitions from update-IR to run-test/idle state. Once asserted, the part disables the TCLK, TMS, and TDI inputs into JTAG and forces these JTAG inputs to logic 1. The TAP controller remains in the run-test/idle state until the TRST input is asserted (logic 0).

31.5.3.5 ENABLE_TEST_CTRL Instruction

The ENABLE_TEST_CTRL instruction selects a 3-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. It allows the control chip to test functions independent of the JTAG TAP controller state.

31.5.3.6 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.

31.5.3.7 LOCKOUT_RECOVERY Instruction

If a user inadvertently enables security on a MCU, the LOCKOUT_RECOVERY instruction allows the disabling of security by the complete erasure of the internal flash contents including the configuration field. This does not compromise security as the entire contents of the user's secured code stored in flash gets erased before security is disabled on the MCU on the next reset or power-up sequence.

The LOCKOUT_RECOVERY instruction selects a 7-bit shift register for connection as a shift path between the TDI pin and the TDO pin. When the user transitions the TAP controller to the UPDATE-DR state, the 7-bit shift register is loaded into the 7-bit JTAG_TFM_CLKDIV register and this value is output to the TFM's clock divider circuit. When the user transitions the TAP controller to the RUN-TEST/IDLE state, the erase signal to the TFM asserts and the lockout sequence starts. The controller must remain in that state until the erase sequence has completed. Once the lockout recovery sequence has completed, the user must reset both the JTAG TAP controller and the MCU to return to normal operation.

31.5.3.8 CLAMP Instruction

The CLAMP instruction selects the 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.

31.5.3.9 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.

31.6 Initialization/Application Information

31.6.1 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 V_{DD}.
- The TMS, TDI, and $\overline{\text{TRST}}$ pins include on-chip pull-up resistors. For minimal power consumption in low-power stop mode, these three pins should be either connected to V_{DD} or left unconnected.

31.6.2 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 $\overline{\text{TRST}}$ be pulled up. $\overline{\text{TRST}}$ could be connected to ground. However, since there is a pull-up on $\overline{\text{TRST}}$, 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 $\overline{\text{TRST}}$.

Chapter 32
Mechanical Data

This chapter contains drawings showing the pinout and the packaging and mechanical characteristics of the MCF528*x* and MCF521*x*.

32.1 Pinout

Figure 32-1 is the pinout for the MCF5280, MCF5281, and MCF5282. The shaded cells illustrate the differences between the MCF5214 and MCF5216 pinout.

NOTE

On the MCF5280 device, which does not contain any flash memory, the VSSF and VDDF pins must be connected to VSS and VDD respectively.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	VSS	A15	A16	A18	A21	VPP	ETXD3	ETXCLK	ERXD3	ERXCLK	ECRS	VDDF	DDATA1	PST2	PST0	VSS
B	A14	A13	A17	A19	VSSF	A22	ETXD2	ERXER	ERXD2	EMDC	ECOL	VSSF	DDATA0	PST1	$\overline{\text{IRQ7}}$	$\overline{\text{IRQ6}}$
C	A12	A11	A10	A20	VDDF	A23	ETXD1	ERXDV	ERXD1	EMDIO	VPP	DDATA3	PST3	$\overline{\text{IRQ5}}$	$\overline{\text{IRQ4}}$	$\overline{\text{IRQ3}}$
D	A9	A8	A7	A6	VDDF	ETXEN	ETXD0	NC	ERXD0	ETXER	VDDF	DDATA2	NC	$\overline{\text{IRQ2}}$	$\overline{\text{IRQ1}}$	CANRX
E	A5	A4	A3	A2	VSS	VDD	VDD	VDD	VDD	VDD	VDD	VSS	CANTX	SDA	SCL	QSPI_DIN
F	A1	A0	D31	NC	VDD	VSS	VDD	VDD	VDD	VDD	VSS	VDD	QSPI_$\overline{\text{DOUT}}$	QSPI_CLK	QSPI_$\overline{\text{CS0}}$	QSPI_CS1
G	D30	D29	D28	D27	VDD	VDD	VSS	VSS	VSS	VSS	VDD	VDD	QSPI_CS2	QSPI_CS3	$\overline{\text{DRAMW}}$	$\overline{\text{SDRAM}}$_$\overline{\text{CS0}}$
H	D26	D25	D24	D23	VDD	VDD	VSS	VSS	VSS	VSS	VDD	VDD	$\overline{\text{SDRA}}$_$\overline{\text{CS1}}$	SCKE	$\overline{\text{SRAS}}$	$\overline{\text{SCAS}}$
J	D22	D21	D20	D19	VDD	VDD	VSS	VSS	VSS	VSS	VDD	VDD	DTOUT0	DTIN0	DTOUT1	DTIN1
K	D18	D17	D16	NC	VDD	VDD	VSS	VSS	VSS	VSS	VDD	VDD	DTOUT2	DTIN2	DTOUT3	DTIN3
L	D15	D14	D13	D12	VDD	VSS	VDD	VDD	VDD	VDD	VSS	VDD	$\overline{\text{CS0}}$	$\overline{\text{CS1}}$	$\overline{\text{CS2}}$	$\overline{\text{CS3}}$
M	D11	D10	D9	D8	VSS	VDD	VDD	VDD	VDD	VDD	VDD	VSS	NC	$\overline{\text{TIP}}$	$\overline{\text{TS}}$	SIZ0
N	D7	D6	D5	NC	D3	URXD0	CLKOUT	VDDPLL	NC	TEST	VSTBY	GPTB0	GPTA0	SIZ1	R/$\overline{\text{W}}$	$\overline{\text{OE}}$
P	D4	VDDH	AN55	VRH	VSSA	D0	UTXD1	VSSPLL	DSCLK	$\overline{\text{BKPT}}$	$\overline{\text{RSTO}}$	GPTB1	GPTA1	$\overline{\text{BS3}}$	$\overline{\text{TEA}}$	$\overline{\text{TA}}$
R	AN3	AN1	AN56	AN52	VDDA	D1	URXD1	XTAL	JTAG_EN	DSI	$\overline{\text{RSTI}}$	GPTB2	GPTA2	CLKMOD0	$\overline{\text{BS1}}$	$\overline{\text{BS0}}$
T	VSSA	AN2	AN0	AN53	VRL	D2	UTXD0	EXTAL	TCLK	DSO	$\overline{\text{RCON}}$	GPTB3	GPTA3	CLKMOD1	$\overline{\text{BS2}}$	VSS

Figure 32-1. MCF528*x* Pinout (256 MAPBGA)

Figure 32-2 is the pinout for the MCF5214 and MCF5216. The shaded cells illustrate the differences between the MCF5280, MCF5281, and MCF5282 pinout.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	VSS	A15	A16	A18	A21	VPP	PEL7	PEL0	PEL3	PAS4	NC	VDDF	DDATA1	PST2	PST0	VSS
B	A14	A13	A17	A19	VSSF	A22	PEL6	NC	PEL2	NC	NC	VSSF	DDATA0	PST1	$\overline{\text{IRQ7}}$	$\overline{\text{IRQ6}}$
C	A12	A11	A10	A20	VDDF	A23	PEL5	NC	PEL1	PAS5	VPP	DDATA3	PST3	$\overline{\text{IRQ5}}$	$\overline{\text{IRQ4}}$	$\overline{\text{IRQ3}}$
D	A9	A8	A7	A6	VDDF	NC	NC	NC	NC	PEL4	VDDF	DDATA2	NC	$\overline{\text{IRQ2}}$	$\overline{\text{IRQ1}}$	CANRX
E	A5	A4	A3	A2	VSS	VDD	VDD	VDD	VDD	VDD	VDD	VSS	CANTX	SDA	SCL	QSPI_DIN
F	A1	A0	D31	NC	VDD	VSS	VDD	VDD	VDD	VDD	VSS	VDD	QSPI_DOUT	QSPI_CLK	QSPI_CS0	QSPI_CS1
G	D30	D29	D28	D27	VDD	VDD	VSS	VSS	VSS	VSS	VDD	VDD	QSPI_CS2	QSPI_CS3	$\overline{\text{DRAMW}}$	$\overline{\text{SDRAM}}$_CS0
H	D26	D25	D24	D23	VDD	VDD	VSS	VSS	VSS	VSS	VDD	VDD	$\overline{\text{SDRA}}$_CS1	SCKE	$\overline{\text{SRAS}}$	$\overline{\text{SCAS}}$
J	D22	D21	D20	D19	VDD	VDD	VSS	VSS	VSS	VSS	VDD	VDD	DTOUT0	DTIN0	DTOUT1	DTIN1
K	D18	D17	D16	NC	VDD	VDD	VSS	VSS	VSS	VSS	VDD	VDD	DTOUT2	DTIN2	DTOUT3	DTIN3
L	D15	D14	D13	D12	VDD	VSS	VDD	VDD	VDD	VDD	VSS	VDD	$\overline{\text{CS0}}$	$\overline{\text{CS1}}$	$\overline{\text{CS2}}$	$\overline{\text{CS3}}$
M	D11	D10	D9	D8	VSS	VDD	VDD	VDD	VDD	VDD	VDD	VSS	NC	$\overline{\text{TIP}}$	$\overline{\text{TS}}$	SIZ0
N	D7	D6	D5	NC	D3	URXD0	CLKOUT	VDDPLL	NC	TEST	VSTBY	GPTB0	GPTA0	SIZ1	R/$\overline{\text{W}}$	$\overline{\text{OE}}$
P	D4	VDDH	AN55	VRH	VSSA	D0	UTXD1	VSSPLL	DSCLK	$\overline{\text{BKPT}}$	$\overline{\text{RSTO}}$	GPTB1	GPTA1	$\overline{\text{BS3}}$	$\overline{\text{TEA}}$	$\overline{\text{TA}}$
R	AN3	AN1	AN56	AN52	VDDA	D1	URXD1	XTAL	JTAG_EN	DSI	$\overline{\text{RSTI}}$	GPTB2	GPTA2	CLKMOD0	$\overline{\text{BS1}}$	$\overline{\text{BS0}}$
T	VSSA	AN2	AN0	AN53	VRL	D2	UTXD0	EXTAL	TCLK	DSO	$\overline{\text{RCON}}$	GPTB3	GPTA3	CLKMOD1	$\overline{\text{BS2}}$	VSS

Figure 32-2. MCF521*x* Pinout (256 MAPBGA)

Table 32-1 lists the MCF521*x* and MCF528*x* signals in pin number order for the 256 MAPBGA package. The shaded cells illustrate the difference signals between the two device families.

Table 32-1. Signal Description by Pin Number

MAPBGA Pin	Pin Functions			MAPBGA Pin	Pin Functions		
	Primary	Secondary	Tertiary		Primary	Secondary	Tertiary
A1	VSS	—	—	J1	D22	PB6	—
A2	A15	PG7	—	J2	D21	PB5	—
A3	A16	PF0	—	J3	D20	PB4	—
A4	A18	PF2	—	J4	D19	PB3	—
A5	A21	PF5	$\overline{\text{CS4}}$	J5	VDD	—	—
A6	VPP	—	—	J6	VDD	—	—
A7 (MCF521*x*)	PEL7	—	—	J7	VSS	—	—
A7 (MCF528*x*)	ETXCLK	PEH7	—				
A8 (MCF521*x*)	PEL0	—	—	J8	VSS	—	—
A8 (MCF528*x*)	ETXCLK	PEH7	—				
A9 (MCF521*x*)	PEL3	—	—	J9	VSS	—	—
A9 (MCF528*x*)	ERXD3	PEL3	—				
A10 (MCF521*x*)	PAS4	UTXD2	—	J10	VSS	—	—
A10 (MCF528*x*)	ERXCLK	PEH3	UTXD2				
A11 (MCF521*x*)	NC[1]	—	—	J11	VDD	—	—
A11 (MCF528*x*)	ECRS	PEH0	—				
A12	VDDF	—	—	J12	VDD	—	—
A13	DDATA1	PDD5	—	J13	DTOUT0	PTD0	$\overline{\text{UCTS1}}$/ $\overline{\text{UCTS0}}$
A14	PST2	PDD2	—	J14	DTIN0	PTD1	$\overline{\text{UCTS1}}$/$\overline{\text{UCTS0}}$
A15	PST0	PDD0	—	J15	DTOUT1	PTD2	$\overline{\text{URTS1}}$/$\overline{\text{URTS0}}$
A16	VSS	—	—	J16	DTIN1	PTD3	$\overline{\text{URTS1}}$/$\overline{\text{URTS0}}$
B1	A14	PG6	—	K1	D18	PB2	—
B2	A13	PG5	—	K2	D17	PB1	—
B3	A17	PF1	—	K3	D16	PB0	—
B4	A19	PF3	—	K4	NC	—	—
B5	VSSF	—	—	K5	VDD	—	—
B6	A22	PF6	$\overline{\text{CS5}}$	K6	VDD	—	—
B7 (MCF521*x*)	PEL6	—	—	K7	VSS	—	—
B7 (MCF528*x*)	ETXD2	PEL6	—				

Table 32-1. Signal Description by Pin Number (continued)

MAPBGA Pin	Pin Functions			MAPBGA Pin	Pin Functions		
	Primary	Secondary	Tertiary		Primary	Secondary	Tertiary
B8 (MCF521*x*)	NC	—	—	K8	VSS	—	—
B8 (MCF528*x*)	ERXER	PEL0	—				
B9 (MCF521*x*)	PEL2	—	—	K9	VSS	—	—
B9 (MCF528*x*)	ERXD2	PEL2	—				
B10 (MCF521*x*)	NC	—	—	K10	VSS	—	—
B10 (MCF528*x*)	EMDC	PAS4	UTXD2				
B11 (MCF521*x*)	NC	—	—	K11	VDD	—	—
B11 (MCF528*x*)	ECOL	PEH4	—				
B12	VSSF	—	—	K12	VDD	—	—
B13	DDATA0	PDD4	—	K13	DTOUT2	PTC0	$\overline{\text{UCTS1}}$/$\overline{\text{UCTS0}}$
B14	PST1	PDD1	—	K14	DTIN2	PTC1	$\overline{\text{UCTS1}}$/$\overline{\text{UCTS0}}$
B15	$\overline{\text{IRQ7}}$	PNQ7	—	K15	DTOUT3	PTC2	$\overline{\text{URTS1}}$/$\overline{\text{URTS0}}$
B16	$\overline{\text{IRQ6}}$	PNQ6	—	K16	DTIN3	PTC3	$\overline{\text{URTS1}}$/$\overline{\text{URTS0}}$
C1	A12	PG4	—	L1	D15	PC7	—
C2	A11	PG3	—	L2	D14	PC6	—
C3	A10	PG2	—	L3	D13	PC5	—
C4	A20	PF4	—	L4	D12	PC4	—
C5	VDDF	—	—	L5	VDD	—	—
C6	A23	PF7	$\overline{\text{CS6}}$	L6	VSS	—	—
C7 (MCF521*x*)	PEL5		—	L7	VDD	—	—
C7 (MCF528*x*)	ETXD1	PEL5	—				
C8 (MCF521*x*)	NC		—	L8	VDD	—	—
C8 (MCF528*x*)	ERXDV	PEH2	—				
C9 (MCF521*x*)	PEL1		—	L9	VDD	—	—
C9 (MCF528*x*)	ERXD1	PEL1	—				
C10 (MCF521*x*)	PAS5	URXD2		L10	VDD	—	—
C10 (MCF528*x*)	EMDIO	PAS5	URXD2				
C11	VPP	—	—	L11	VSS	—	—
C12	DDATA3	PDD7	—	L12	VDD	—	—
C13	PST3	PDD3	—	L13	$\overline{\text{CS0}}$	PJ0	—
C14	$\overline{\text{IRQ5}}$	PNQ5	—	L14	$\overline{\text{CS1}}$	PJ1	—
C15	$\overline{\text{IRQ4}}$	PNQ4	—	L15	$\overline{\text{CS2}}$	PJ2	—

Table 32-1. Signal Description by Pin Number (continued)

MAPBGA Pin	Pin Functions			MAPBGA Pin	Pin Functions		
	Primary	Secondary	Tertiary		Primary	Secondary	Tertiary
C16	$\overline{\text{IRQ3}}$	PNQ3	—	L16	$\overline{\text{CS3}}$	PJ3	—
D1	A9	PG1	—	M1	D11	PC3	—
D2	A8	PG0	—	M2	D10	PC2	—
D3	A7	PH7	—	M3	D9	PC1	—
D4	A6	PH6	—	M4	D8	PC0	—
D5	VDDF	—	—	M5	VSS	—	—
D6 (MCF521*x*)	NC	—	—	M6	VDD	—	—
D6 (MCF528*x*)	ETXEN	PEH6	—				
D7 (MCF521*x*)	NC	—	—	M7	VDD	—	—
D7 (MCF528*x*)	ETXD0	PEH5	—				
D8	NC	—	—	M8	VDD	—	—
D9 (MCF521*x*)	NC	—	—	M9	VDD	—	—
D9 (MCF528*x*)	ERXD0	PEH1	—				
D10 (MCF521*x*)	PEL4	—	—	M10	VDD	—	—
D10 (MCF528*x*)	ETXER	PEL4	—				
D11	VDDF	—	—	M11	VDD	—	—
D12	DDATA2	PDD6	—	M12	VSS	—	—
D13	NC	—	—	M13	NC	—	—
D14	$\overline{\text{IRQ2}}$	PNQ2	—	M14	$\overline{\text{TIP}}$	PE0	SYNCB
D15	$\overline{\text{IRQ1}}$	PNQ1	—	M15	$\overline{\text{TS}}$	PE1	SYNCA
D16	CANRX	PAS3	URXD2	M16	SIZ0	PE2	SYNCB
E1	A5	PH5	—	N1	D7	PD7	—
E2	A4	PH4	—	N2	D6	PD6	—
E3	A3	PH3	—	N3	D5	PD5	—
E4	A2	PH2	—	N4	NC	—	—
E5	VSS	—	—	N5	D3	PD3	—
E6	VDD	—	—	N6	URXD0	PUA1	—
E7	VDD	—	—	N7	CLKOUT	—	—
E8	VDD	—	—	N8	VDDPLL	—	—
E9	VDD	—	—	N9	NC	—	—
E10	VDD	—	—	N10	TEST	—	—
E11	VDD	—	—	N11	VSTBY	—	—

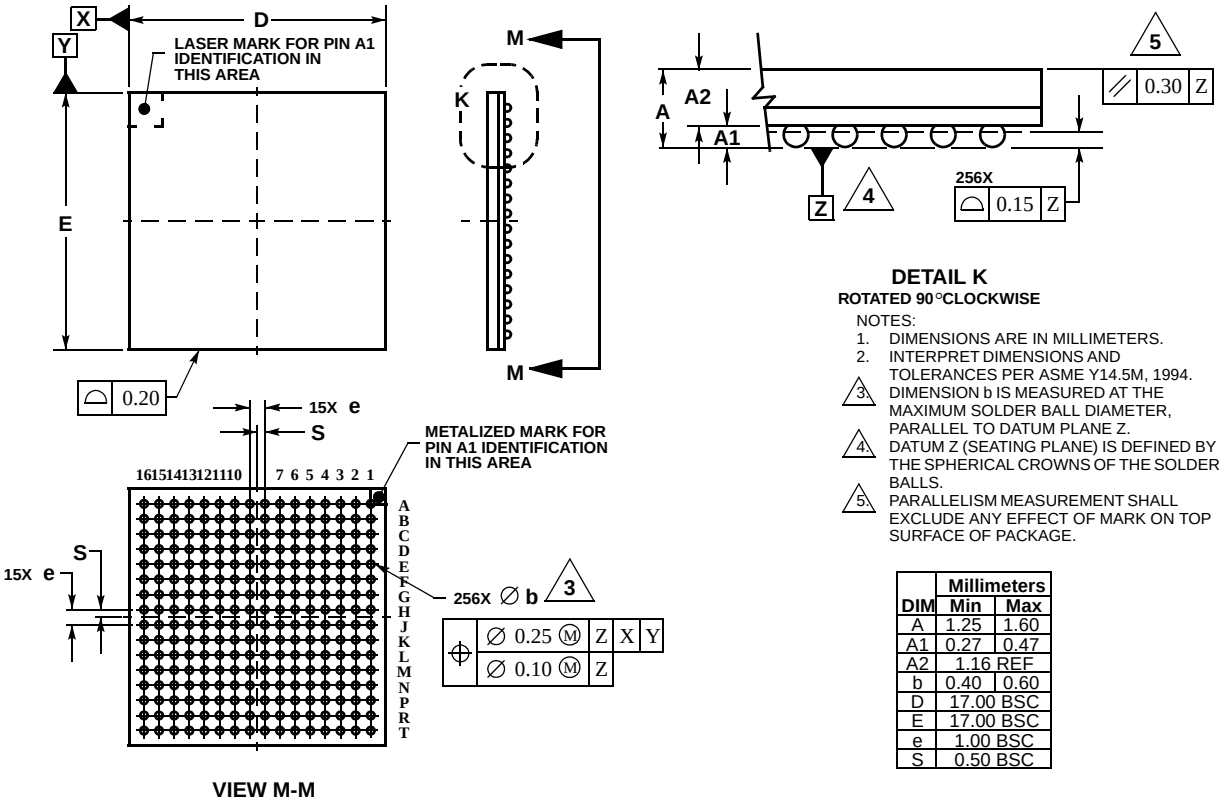
Table 32-1. Signal Description by Pin Number (continued)

MAPBGA Pin	Pin Functions			MAPBGA Pin	Pin Functions		
	Primary	Secondary	Tertiary		Primary	Secondary	Tertiary
E12	VSS	—	—	N12	GPTB0	PTB0	—
E13	CANTX	PAS2	UTXD2	N13	GPTA0	PTA0	—
E14	SDA	PAS1	URXD2	N14	SIZ1	PE3	SYNCA
E15	SCL	PAS0	UTXD2	N15	R/$\overline{W}$	PE4	—
E16	QSPI_DIN	PQS1	—	N16	$\overline{OE}$	PE7	—
F1	A1	PH1	—	P1	D4	PD4	—
F2	A0	PH0	—	P2	VDDH	—	—
F3	D31	PA7	—	P3	AN55	PQA3	ETRIG1
F4	NC	—	—	P4	VRH	—	—
F5	VDD	—	—	P5	VSSA	—	—
F6	VSS	—	—	P6	D0	PD0	—
F7	VDD	—	—	P7	UTXD1	PUA2	—
F8	VDD	—	—	P8	VSSPLL	—	—
F9	VDD	—	—	P9	DSCLK	$\overline{TRST}$	—
F10	VDD	—	—	P10	$\overline{BKPT}$	TMS	—
F11	VSS	—	—	P11	$\overline{RSTO}$	—	—
F12	VDD	—	—	P12	GPTB1	PTB1	—
F13	QSPI_DOUT	PQS0	—	P13	GPTA1	PTA1	—
F14	QSPI_CLK	PQS2	—	P14	$\overline{BS3}$	PJ7	—
F15	QSPI_CS0	PQS3	—	P15	$\overline{TEA}$	PE5	—
F16	QSPI_CS1	PQS4	—	P16	$\overline{TA}$	PE6	—
G1	D30	PA6	—	R1	AN3	PQB3	ANZ
G2	D29	PA5	—	R2	AN1	PQB1	ANX
G3	D28	PA4	—	R3	AN56	PQA4	ETRIG2
G4	D27	PA3	—	R4	AN52	PQA0	MA0
G5	VDD	—	—	R5	VDDA	—	—
G6	VDD	—	—	R6	D1	PD1	—
G7	VSS	—	—	R7	URXD1	PUA3	—
G8	VSS	—	—	R8	XTAL	—	—
G9	VSS	—	—	R9	JTAG_EN	—	—
G10	VSS	—	—	R10	DSI	TDI	—
G11	VDD	—	—	R11	$\overline{RSTI}$	—	—

Table 32-1. Signal Description by Pin Number (continued)

MAPBGA Pin	Pin Functions			MAPBGA Pin	Pin Functions		
	Primary	Secondary	Tertiary		Primary	Secondary	Tertiary
G12	VDD	—	—	R12	GPTB2	PTB2	—
G13	QSPI_CS2	PQS5	—	R13	GPTA2	PTA2	—
G14	QSPI_CS3	PQS6	—	R14	CLKMOD0	—	—
G15	$\overline{\text{DRAMW}}$	PSD3	—	R15	$\overline{\text{BS1}}$	PJ5	—
G16	$\overline{\text{SDRAM\_CS0}}$	PSD1	—	R16	$\overline{\text{BS0}}$	PJ4	—
H1	D26	PA2	—	T1	VSSA	—	—
H2	D25	PA1	—	T2	AN2	PQB2	ANY
H3	D24	PA0	—	T3	AN0	PQB0	ANW
H4	D23	PB7	—	T4	AN53	PQA1	MA1
H5	VDD	—	—	T5	VRL	—	—
H6	VDD	—	—	T6	D2	PD2	—
H7	VSS	—	—	T7	UTXD0	PUA0	—
H8	VSS	—	—	T8	EXTAL	—	—
H9	VSS	—	—	T9	TCLK	—	—
H10	VSS	—	—	T10	DSO	TDO	—
H11	VDD	—	—	T11	$\overline{\text{RCON}}$	—	—
H12	VDD	—	—	T12	GPTB3	PTB3	—
H13	$\overline{\text{SDRAM\_CS1}}$	PSD2	—	T13	GPTA3	PTA3	—
H14	SCKE	PSD0	—	T14	CLKMOD1	—	—
H15	$\overline{\text{SRAS}}$	PSD5	—	T15	$\overline{\text{BS2}}$	PJ6	—
H16	$\overline{\text{SCAS}}$	PSD4	—	T16	VSS	—	—

[1] NC = no connect



NOTES:
1. DIMENSIONS ARE IN MILLIMETERS.
2. INTERPRET DIMENSIONS AND TOLERANCES PER ASME Y14.5M, 1994.
3. DIMENSION b IS MEASURED AT THE MAXIMUM SOLDER BALL DIAMETER, PARALLEL TO DATUM PLANE Z.
4. DATUM Z (SEATING PLANE) IS DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.
5. PARALLELISM MEASUREMENT SHALL EXCLUDE ANY EFFECT OF MARK ON TOP SURFACE OF PACKAGE.

DIM	Millimeters Min	Max
A	1.25	1.60
A1	0.27	0.47
A2	1.16 REF	
b	0.40	0.60
D	17.00 BSC	
E	17.00 BSC	
e	1.00 BSC	
S	0.50 BSC	

Figure 32-3. 256 MAPBGA Package Dimensions

32.2 Ordering Information

Table 32-2. Orderable Part Numbers

Freescale Part Number	Description	Package	Speed	Temperature
MCF5214CVF66	MCF5214 RISC Microprocessor	256 MAPBGA	66.67 MHz	-40° to +85° C
MCF5216CVF66	MCF5216 RISC Microprocessor	256 MAPBGA	66.67 MHz	-40° to +85° C
MCF5280CVF66	MCF5280 RISC Microprocessor	256 MAPBGA	66.67 MHz	-40° to +85° C
MCF5280CVF80	MCF5280 RISC Microprocessor	256 MAPBGA	80 MHz	-40° to +85° C
MCF5281CVF66	MCF5281 RISC Microprocessor	256 MAPBGA	66.67 MHz	-40° to +85° C
MCF5281CVF80	MCF5281 RISC Microprocessor	256 MAPBGA	80 MHz	-40° to +85° C
MCF5282CVF66	MCF5282 RISC Microprocessor	256 MAPBGA	66.67 MHz	-40° to +85° C
MCF5282CVF80	MCF5282 RISC Microprocessor	256 MAPBGA	80 MHz	-40° to +85° C

Chapter 33
Electrical Characteristics

This chapter contains electrical specification tables and reference timing diagrams for the MCF528*x* and MCF521*x* microcontroller units. This section contains detailed information on power considerations, DC/AC electrical characteristics, and AC timing specifications.

NOTE

The parameters specified in this MCU document supersede any values found in the module specifications.

33.1 Maximum Ratings

Table 33-1. Absolute Maximum Ratings[1, 2]

Rating	Symbol	Value	Unit
Supply Voltage	V_{DD}	– 0.3 to +4.0	V
Clock Synthesizer Supply Voltage	V_{DDPLL}	– 0.3 to +4.0	V
RAM Memory Standby Supply Voltage	V_{STBY}	– 0.3 to + 4.0	V
Flash Memory Supply Voltage[3]	V_{DDF}	– 0.3 to +4.0	V
Flash Memory Program / Erase Supply Voltage[3]	V_{PP}	– 0.3 to + 6.0	V
Analog Supply Voltage	V_{DDA}	– 0.3 to +6.0	V
Analog Reference Supply Voltage	V_{RH}	– 0.3 to +6.0	V
Analog ESD Protection Voltage	V_{DDH}	– 0.3 to +6.0	V
Digital Input Voltage[4]	V_{IN}	– 0.3 to + 6.0	V
Analog Input Voltage	V_{AIN}	– 0.3 to + 6.0	V
EXTAL pin voltage	V_{EXTAL}	0 to 3.3	V
XTAL pin voltage	V_{XTAL}	0 to 3.3	V
Instantaneous Maximum Current Single pin limit (applies to all pins) [3, 5, 6]	I_D	25	mA
Maximum Power Supply Current[6]	I_{DD}	300	mA
Operating Temperature Range (Packaged)	T_A	– 40 to 85	°C
Storage Temperature Range	T_{stg}	– 65 to 150	°C
Maximum operating junction temperature	T_j	105	°C
ESD Target for Human Body Model[7]	HBM	2000	V

1 Functional operating conditions are given in DC Electrical Specifications. Absolute Maximum Ratings are stress ratings only, and functional operation at the maxima is not guaranteed. Stress beyond those listed may affect device reliability or cause permanent damage to the device.

2 This device contains circuitry protecting against damage due to high static voltage or electrical fields; however, it is advised that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{DD}).

3 Not applicable for MCF5280.

4 Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive and negative clamp voltages, then use the larger of the two values. 6.0V voltage excludes XTAL and EXTAL pads.

5 All functional non-supply pins are internally clamped to V_{SS} and V_{DD}.

6 Power supply must maintain regulation within operating V_{DD} range during instantaneous and operating maximum current conditions. If positive injection current ($V_{in} > V_{DD}$) is greater than I_{DD}, the injection current may flow out of V_{DD} and could result in external power supply going out of regulation. Insure external V_{DD} load will shunt current greater than maximum injection current. This will be the greatest risk when the MCU is not consuming power (ex; no clock).Power supply must maintain regulation within operating V_{DD} range during instantaneous and operating maximum current conditions.

7 All ESD testing methodology is in conformity with CDF-AEC-Q100 Stress Test Qualification for Automotive Grade Integrated Circuits.

NOTE

It is crucial during power-up that V_{DD} never exceeds V_{DDH} by more than ≈0.3 V. There are diode devices between the two voltage domains, and violating this rule can lead to a latch-up condition.

33.2 Thermal Characteristics

Table 33-2 lists thermal resistance values.

Table 33-2. Thermal Characteristics

Characteristic		Symbol	Value	Unit
Junction to ambient, natural convection	Four layer board (2s2p)	θ_{JMA}	26[1,2]	°C/W
Junction to ambient (@200 ft/min)	Four layer board (2s2p)	θ_{JMA}	23[1,2]	°C/W
Junction to board		θ_{JB}	15[3]	°C/W
Junction to case		θ_{JC}	10[4]	°C/W
Junction to top of package	Natural convection	Ψ_{jt}	2[1,5]	°C/W

1 θ_{JMA} and Ψ_{jt} parameters are simulated in accordance with EIA/JESD Standard 51-2 for natural convection. Freescale recommends the use of θ_{JA} and power dissipation specifications in the system design to prevent device junction temperatures from exceeding the rated specification. System designers should be aware that device junction temperatures can be significantly influenced by board layout and surrounding devices. Conformance to the device junction temperature specification can be verified by physical measurement in the customer's system using the Ψ_{jt} parameter, the device power dissipation, and the method described in EIA/JESD Standard 51-2.

2 Per JEDEC JESD51-6 with the board horizontal.

3 Thermal resistance between the die and the printed circuit board per JEDEC JESD51-8. Board temperature is measured on the top surface of the board near the package.

4 Thermal resistance between the die and the case top surface as measured by the cold plate method (MIL SPEC-883 Method 1012.1).

5 Thermal characterization parameter indicating the temperature difference between package top and the junction temperature per JEDEC JESD51-2. When Greek letters are not available, the thermal characterization parameter is written as Psi-JT.

The average chip-junction temperature (T_J) in °C can be obtained from:

$$T_J = T_A + (P_D \times \Theta_{JMA}) \quad (1)$$

Where:

T_A = Ambient Temperature, ×C

Q_{JMA} = Package Thermal Resistance, Junction-to-Ambient, ×C/W

P_D = $P_{INT} + P_{I/O}$

P_{INT} = $I_{DD} \times V_{DD}$, Watts - Chip Internal Power

$P_{I/O}$ = Power Dissipation on Input and Output Pins — User Determined

For most applications $P_{I/O} < P_{INT}$ and can be neglected. An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273°C) \quad (2)$$

Solving equations 1 and 2 for K gives:

$$K = P_D \times (T_A + 273 \times C) + Q_{JMA} \times P_D^2 \quad (3)$$

where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring P_D (at equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

33.3 DC Electrical Specifications

Table 33-3. DC Electrical Specifications[1]

($V_{SS} = V_{SSPLL} = V_{SSF} = V_{SSA} = 0\ V_{DC}$)

Characteristic	Symbol	Min	Max	Unit
Input High Voltage	V_{IH}	0.7 x V_{DD}	5.25	V
Input Low Voltage	V_{IL}	V_{SS} – 0.3	0.35 x V_{DD}	V
Input Hysteresis	V_{HYS}	0.06 x V_{DD}	—	mV
Input Leakage Current V_{in} = V_{DD} or V_{SS}, Input-only pins	I_{in}	-1.0	1.0	µA
High Impedance (Off-State) Leakage Current V_{in} = V_{DD} or V_{SS}, All input/output and output pins	I_{OZ}	-1.0	1.0	µA
Output High Voltage (All input/output and all output pins) I_{OH} = –5.0 mA	V_{OH}	V_{DD} - 0.5	—	V
Output Low Voltage (All input/output and all output pins) I_{OL} = 5.0mA	V_{OL}	—	0.5	V
Weak Internal Pull Up Device Current, tested at V_{IL} Max.	I_{APU}	-10	-130	µA
Input Capacitance [2] All input-only pins All input/output (three-state) pins	C_{in}	 — —	 7 7	pF

Table 33-3. DC Electrical Specifications[1] (continued)

(V_{SS} = V_{SSPLL} = V_{SSF} = V_{SSA} = 0 V_{DC})

Characteristic	Symbol	Min	Max	Unit
Load Capacitance[3] (50% Partial Drive) (100% Full Drive)	C_L		 25 50	pF
Supply Voltage (includes core modules and pads)	V_{DD}	2.7	3.6	V
RAM Memory Standby Supply Voltage Normal Operation: $V_{DD} > V_{STBY}$ - 0.3 V Standby Mode: $V_{DD} < V_{STBY}$ - 0.3 V	V_{STBY}	 0.0 1.8	 3.6 3.6	V
Flash Memory Supply Voltage (not applicable to MCF5280)	V_{DDF}	2.7	3.6	V

1 Refer to Table 33-8 through Table 33-12 for additional PLL, QADC, and Flash specifications.
2 This parameter is characterized before qualification rather than 100% tested.
3 Refer to the chip configuration section for more information. Drivers for the SDRAM pins are at 25pF drive strength.Drivers for the QADC pins are at 50pF drive strength.

33.4 Power Consumption Specifications

Table 33-4. STOP Mode Current Consumption Specifications

Characteristic	Symbol	Typical– Master Mode	Typical– Single Chip Mode[1]	Max[2]	Unit
System clocks disabled (LPCR[STPMD] = 00)	I_{DD}	25	7.9	—	mA
System clocks and CLKOUT disabled (LPCR[STPMD] = 01)	I_{DD}	7.3	5.6	—	mA
System clocks, CLKOUT, and PLL disabled (LPCR[STPMD] = 10)	I_{DD}	4.5	4.7	—	mA
System clocks, CLKOUT, PLL, and OSC disabled (LPCR[STPMD] = 11)	I_{DD}	400	750	1000	µA

1 Single chip mode current measured with all pins in general purpose input mode except for the UART0 and FEC pins that are enabled for their module functionality.
2 Maximum values can vary depending on the system's state and signal loading.

Figure 33-1 shows typical WAIT/DOZE and RUN mode power consumption for both master and single chip mode as measured on an M5282EVB.

For master mode the RUN mode current was measured executing a continuous loop that performs no operation while running from the on-chip SRAM.

For WAIT/DOZE mode measurements the peripherals on the device are in their default power savings mode, so the WAIT and DOZE power consumption are the same. Some modules can be programmed to shutdown in WAIT and/or DOZE modes. Refer to module chapters for more information.

All single chip mode measurements were taken with all pins in general purpose input mode except for the UART0 and FEC pins that are enabled for their module functionality; however, neither module is being accessed at the time of the current measurement. Single chip RUN mode current was measured executing a continuous loop that performs no operation while running from the on-chip Flash.

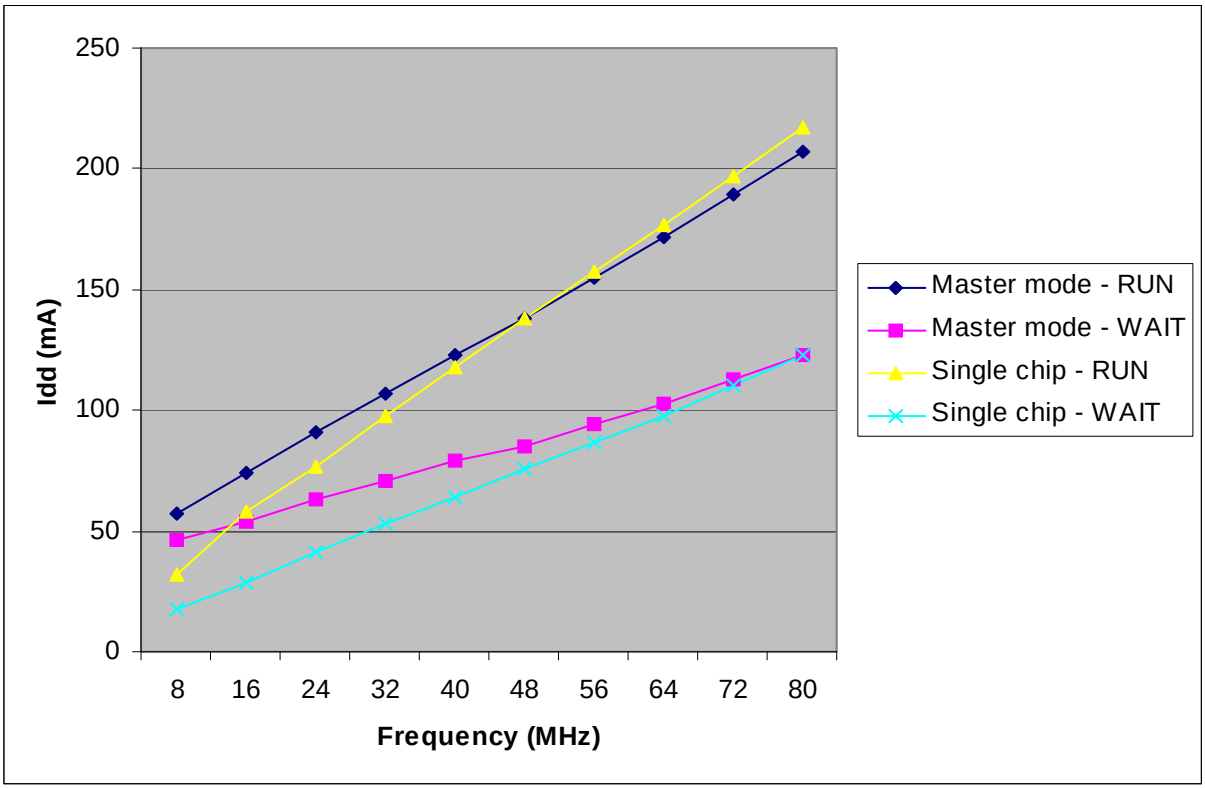


Figure 33-1. Typical WAIT/DOZE Mode Current Consumption

Table 33-5 lists the estimated power consumption for individual modules. The current consumption is for the module itself and does not include power for I/O.

Table 33-5. Estimated Module Power Consumption

Module	Estimated Power	Unit
EIM	20	µA/MHz
SDRAMC	30	µA/MHz
FEC	60	µA/MHz
Watchdog	1.5	µA/MHz
PIT	1	µA/MHz
FlexCAN	15	µA/MHz
QSPI, UART, I^2C, and Timers	75	µA/MHz

Table 33-6. Typical Application Power Consumption

Application	Current
Dhrystone benchmark running from cache and on-chip SRAM (running at 64 MHz f_{sys})	181.6 mA
dBUG ROM monitor running from external flash and SDRAM (running at 64 MHz f_{sys})	155 mA

Table 33-7 lists the maximum power consumption specifications.

Table 33-7. Maximum Power Consumption Specifications

Characteristic	Symbol	Typical	Max	Unit
Operating Supply Current [1]	I_{DD}			
Master Mode				
• 66 MHz		—	200	mA
• 80 MHz (MCF528*x* only)		—	240	mA
Single Chip Mode		—	150	mA
WAIT/DOZE		—		
• 66 MHz		—	125	mA
• 80 MHz (MCF528*x* only)		—	150	mA
Clock Synthesizer Supply Current	I_{DDPLL}			
Normal Operation 8.25 MHz crystal, VCO on, Max f_{sys}		—	4	mA
STOP (OSC and PLL enabled)		—	2	mA
STOP (OSC enable, PLL disabled)		—	1	mA
STOP (OSC and PLL disabled)		—	10	µA
RAM Memory Standby Supply Current	I_{STBY}			
Normal Operation: $V_{DD} > V_{STBY} - 0.3$ V		—	10	µA
Transient Condition: $V_{STBY} - 0.3$ V > $V_{DD} > V_{SS} + 0.5$ V		—	7	mA
Standby Operation: $V_{DD} < V_{SS} + 0.5$ V		—	20	µA
Flash Memory Supply Current	I_{DDF}			
Read		16.5[3]	30	mA
Program or Erase[2]		25[4]	64	mA
Idle		1.6[4]	20	mA
STOP		0.2	10	µA
Analog Supply Current	I_{DDA}			
Normal Operation		—	5.0	mA
Low-Power Stop		—	10.0	µA

1 Current measured at maximum system clock frequency, all modules active, and default drive strength with matching load.

2 Programming and erasing all 8 blocks of the Flash.

3 Measured with f_{sys} of 64 MHz.

4 Measured with f_{sys} of 32 MHz and f_{clk} of 187.5 kHz.

33.5 Phase Lock Loop Electrical Specifications

Table 33-8. PLL Electrical Specifications

(V_{DD} and V_{DDPLL} = 2.7 to 3.6 V, V_{SS} = V_{SSPLL} = 0 V)

Characteristic	Symbol	Min	Max		Unit
			66MHz	80MHz	
PLL Reference Frequency Range Crystal reference External reference 1:1 Mode	 $f_{ref\_crystal}$ f_{ref_ext} $f_{ref\_1:1}$	 2 2 33.33	 8.33 8.33 66.66	 10.0 10.0 80	MHz
System Frequency [1] External Clock Mode On-Chip PLL Frequency	f_{sys}	 0 f_{ref} / 32	$f_{sys(max)}$ 66.66 66.66	$f_{sys(max)}$ 80 80	MHz
Loss of Reference Frequency [2, 4]	f_{LOR}	100	1000		kHz
Self Clocked Mode Frequency [3, 4]	f_{SCM}	1	5		MHz
Crystal Start-up Time [4, 5]	t_{cst}	—	10		ms
EXTAL Input High Voltage Crystal Mode All other modes (1:1, Bypass, External)	V_{IHEXT}	 V_{XTAL} + 0.4 2.0	 V_{DD} V_{DD}		V
EXTAL Input Low Voltage Crystal Mode All other modes (1:1, Bypass, External)	V_{ILEXT}	 V_{SS} V_{SS}	 V_{XTAL} - 0.4 0.8		V
XTAL Output High Voltage I_{OH} = 1.0 mA	V_{OL}	 V_{DD}- 1.0	 —		V
XTAL Output Low Voltage I_{OL} = 1.0 mA	V_{OL}	 —	 0.5		V
XTAL Load Capacitance[6]		—	—		pF
PLL Lock Time[4,7]	t_{lpll}	—	500		µs
Power-up To Lock Time [4, 5,8] With Crystal Reference Without Crystal Reference	t_{lplk}	 — —	 10.5 500		 ms µs
1:1 Clock Skew (between CLKOUT and EXTAL) [9]	t_{skew}	-2	2		ns
Duty Cycle of reference [4]	t_{dc}	40	60		% f_{sys}
Frequency un-LOCK Range	f_{UL}	- 1.5	1.5		% f_{sys}
Frequency LOCK Range	fLCK	- 0.75	0.75		% % f_{sys}
CLKOUT Period Jitter [4, 5, 7, 10,11] , Measured at f_{SYS} Max Peak-to-peak (Clock edge to clock edge) — MCF521*x* Peak-to-peak (Clock edge to clock edge) — MCF528*x* Long Term Jitter (Averaged over 2 ms interval)	C_{jitter}	 — — —	 5 10 .01		% f_{sys}

1 All internal registers retain data at 0 Hz.

2 “Loss of Reference Frequency” is the reference frequency detected internally, which transitions the PLL into self clocked mode.

3 Self clocked mode frequency is the frequency that the PLL operates at when the reference frequency falls below f_{LOR} with default MFD/RFD settings.

4 This parameter is characterized before qualification rather than 100% tested.

5 Proper PC board layout procedures must be followed to achieve specifications.

6 Load Capacitance determined from crystal manufacturer specifications and will include circuit board parasitics.

7 This specification applies to the period required for the PLL to relock after changing the MFD frequency control bits in the synthesizer control register (SYNCR).

8 Assuming a reference is available at power up, lock time is measured from the time V_{DD} and V_{DDPLL} are valid to $\overline{RSTO}$ negating. If the crystal oscillator is being used as the reference for the PLL, then the crystal start up time must be added to the PLL lock time to determine the total start-up time.

9 PLL is operating in 1:1 PLL mode.

10 Jitter is the average deviation from the programmed frequency measured over the specified interval at maximum f_{sys}. Measurements are made with the device powered by filtered supplies and clocked by a stable external clock signal. Noise injected into the PLL circuitry via V_{DDPLL} and V_{SSPLL} and variation in crystal oscillator frequency increase the Cjitter percentage for a given interval

11 Based on slow system clock of 40 MHz measured at f_{sys} max.

33.6 QADC Electrical Characteristics

Table 33-9. QADC Absolute Maximum Ratings

Parameter	Symbol	Min	Max	Unit
Analog Supply, with reference to V_{SSA}	V_{DDA}	–0.3	6.0	V
Internal Digital Supply [1], with reference to V_{SS}	V_{DD}	–0.3	4.0	V
Reference Supply, with reference to V_{RL}	V_{RH}	–0.3	6.0	V
V_{SS} Differential Voltage	$V_{SS} – V_{SSA}$	–0.1	0.1	V
V_{DD} Differential Voltage [2]	$V_{DD} – V_{DDA}$	–6.0	4.0	V
V_{REF} Differential Voltage	$V_{RH} – V_{RL}$	–0.3	6.0	V
V_{RH} to V_{DDA} Differential Voltage [3]	$V_{RH} – V_{DDA}$	–6.0	6.0	V
V_{RL} to V_{SSA} Differential Voltage	$V_{RL} – V_{SSA}$	–0.3	0.3	V
V_{DDH} to V_{DDA} Differential Voltage	$V_{DDH} – V_{DDA}$	–1.0	1.0	V
Maximum Input Current [4, 5, 6]	I_{MA}	–25	25	mA

1 For internal digital supply of V_{DD} = 3.3V typical.

2 Refers to allowed random sequencing of power supplies.

3 Refers to allowed random sequencing of power supplies.

4 Transitions within the limit do not affect device reliability or cause permanent damage. Exceeding limit may cause permanent conversion error on stressed channels and on unstressed channels.

5 Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values using $V_{POSCLAMP} = V_{DDA} + 0.3V$ and $V_{NEGCLAMP} = –0.3$ V, then use the larger of the calculated values.

6 Condition applies to one pin at a time.

Table 33-10. QADC Electrical Specifications (Operating) [1]

(V_{DDH} and V_{DDA} = 5.0 V_{dc} ± 0.5V, V_{DD} = 2.7-3.6V, V_{SS} and V_{SSA} = 0 V_{dc}, FQCLK = 2.0 MHz, TA within operating temperature range)

Parameter	Symbol	Min	Max	Unit
Analog Supply — MCF521*x* Analog Supply — MCF528*x*	V_{DDA}	4.5 3.3	5.5 5.5	V
V_{SS} Differential Voltage	$V_{SS} - V_{SSA}$	-100	100	mV
Reference Voltage Low [2]	V_{RL}	V_{SSA}	V_{SSA} + 0.1	V
Reference Voltage High [2]	V_{RH}	V_{DDA} – 0.1	V_{DDA}	V
V_{REF} Differential Voltage — MCF521*x* V_{REF} Differential Voltage — MCF528*x*	$V_{RH} - V_{RL}$	4.5 3.3	5.5 5.5	V
Input Voltage	V_{INDC}	V_{SSA}–0.3	V_{DDA} + 0.3	V
Input High Voltage, PQA and PQB	V_{IH}	0.7 (V_{DDA})	V_{DDA} + 0.3	V
Input Low Voltage, PQA and PQB	V_{IL}	V_{SSA} – 0.3	0.4(V_{DDA})	V
Input Hysteresis, PQA, PQB [3]	V_{HYS}	0.5	—	V
Output High Voltage, PQA/PQB [3] I_{OH} = TBD	V_{OH}	VDDH-0.8	— —	V
Analog Supply Current Normal Operation [4] Low-Power Stop	I_{DDA}	— —	5.0 10.0	mA µA
Reference Supply Current, DC Reference Supply Current, Transient	I_{REF} I_{REF}	— —	250 2.0	µA mA
Load Capacitance, PQA/PQB	C_L	—	50	pF
Input Current, Channel Off [5]	I_{OFF}	-200	200	nA
Total Input Capacitance [6] PQA Not Sampling PQB Not Sampling Incremental Capacitance added during sampling	C_{IN}	— — —	15 10 5	pF

1 QADC converter specifications are only guaranteed for VDDH and VDDA = 5.0V +/- 0.5V. VDDH and VDDA may be powered down to 2.7V with only GPIO functions supported.

2 To obtain full-scale, full-range results, $V_{SSA} \le V_{RL} \le V_{INDC} \le V_{RH} \le V_{DDA}$

3 Parameter applies to the following pins:
Port A: PQA[4:3]/AN[56:55]/ETRIG[2:1], PQA[1:0]/AN[53:52]/MA[1:0]
Port B: PQB[3:0]/AN[3:0]/AN[Z:W]

4 Current measured at maximum system clock frequency with QADC active.

5 Maximum leakage occurs at maximum operating temperature. Current decreases by approximately one-half for each 8 to 12° C, in the ambient temperature range of 50 to 125 °C.

6 This parameter is characterized before qualification rather than 100% tested.

Table 33-11. QADC Conversion Specifications (Operating)

(V_{DDH} and V_{DDA} = 5.0 V_{dc} ± 0.5V, V_{DD}= 2.7-3.6V, V_{SS} and V_{SSA}= 0 V_{dc}, V_{RH} – V_{RL} = 5 V_{dc} ± 0.5V, T_A within operating temperature range, f_{sys} = 16 MHz)

Num	Parameter	Symbol	Min	Max	Unit
1	QADC Clock (QCLK) Frequency[1]	F_{QCLK}	0.5	2.1	MHz
2	Conversion Cycles	CC	14	28	QCLK cycles
3	Conversion Time F_{QCLK} = 2.0 MHz[1] Min = CCW/IST =%00 Max = CCW/IST =%11	T_{CONV}	7.0	14.0	µs
4	Stop Mode Recovery Time	T_{SR}	—	10	µs
5	Resolution[2]	—	5	—	mV
6	Absolute (total unadjusted) error [3, 4, 5] F_{QCLK} = 2.0 MHz [2], 2 clock input sample time	AE	-2	2	Counts
7	Absolute (total unadjusted) error [3, 4, 5] F_{QCLK} = 2.0 MHz [2], 2 clock input sample time	—	-3	3	Counts

1 Conversion characteristics vary with F_{QCLK} rate. Reduced conversion accuracy occurs at max F_{QCLK} rate. Using the QADC pins as GPIO functions during conversions may result in degraded results. Best QADC conversion accuracy is achieved at a frequency of 2 MHz.

2 At V_{RH} – V_{RL} = 5.12 V, one count = 5 mV

3 Accuracy tested and guaranteed at V_{RH} – V_{RL} = 5.0V ± 0.5V

4 Current Coupling Ratio, K, is defined as the ratio of the output current, I_{out}, measured on the pin under test to the injection current, I_{inj}, when both adjacent pins are overstressed with the specified injection current. K = I_{out}/I_{inj}. The input voltage error on the channel under test is calculated as Verr = $I_{inj} * K * R_S$.

5 Performance expected with production silicon.

33.7 Flash Memory Characteristics

The Flash memory characteristics are shown in Table 33-12 and Table 33-13.

NOTE

The MCF5280 does not contain Flash memory.

Table 33-12. SGFM Flash Program and Erase Characteristics

(V_{DDF} = 2.7 to 3.6 V)

Parameter	Symbol	Min	Typ	Max	Unit
System clock (read only) — MCF521*x* System clock (read only) — MCF528*x*	$f_{sys(R)}$	0 0	—	66.67 80	MHz
System clock (program/erase)[1] — MCF521*x* System clock (program/erase)[1] — MCF528*x*	$f_{sys(P/E)}$	0.15	—	66.67 80	MHz

1 Refer to the Flash section for more information

Table 33-13. SGFM Flash Module Life Characteristics

(V_{DDF} = 2.7 to 3.6 V)

Parameter	Symbol	Value	Unit
Maximum number of guaranteed program/erase cycles[1] before failure	P/E	10,000[2]	Cycles
Data retention at average operating temperature of 85°C	Retention	10	Years

1 A program/erase cycle is defined as switching the bits from 1 → 0 → 1.

2 Reprogramming of a Flash array block prior to erase is not required.

33.8 External Interface Timing Characteristics

Table 33-14 lists processor bus input timings.

NOTE

All processor bus timings are synchronous; that is, input setup/hold and output delay with respect to the rising edge of a reference clock. The reference clock is the CLKOUT output.

All other timing relationships can be derived from these values.

Table 33-14. Processor Bus Input Timing Specifications

Name	Characteristic[1]	Symbol	Min	Max	Unit
B0	CLKOUT — MCF521*x* CLKOUT — MCF528*x*	t_{CYC}	15.15 12.5	—	ns
Control Inputs					
B1a	Control input valid to CLKOUT high[2]	t_{CVCH}	10	—	ns
B1b	$\overline{\text{BKPT}}$ valid to CLKOUT high[3]	t_{BKVCH}	10	—	ns
B2a	CLKOUT high to control inputs invalid[2]	t_{CHCII}	0	—	ns
B2b	CLKOUT high to asynchronous control input $\overline{\text{BKPT}}$ invalid[3]	t_{BKNCH}	0	—	ns
Data Inputs					
B4	Data input (D[31:0]) valid to CLKOUT high	t_{DIVCH}	6	—	ns
B5	CLKOUT high to data input (D[31:0]) invalid	t_{CHDII}	0	—	ns

1 Timing specifications have been indicated taking into account the full drive strength for the pads.

2 $\overline{\text{TEA}}$ and $\overline{\text{TA}}$ pins are being referred to as control inputs.

3 Refer to figure A-19.

Timings listed in Table 33-14 are shown in Figure 33-2.

* The timings are also valid for inputs sampled on the negative clock edge.

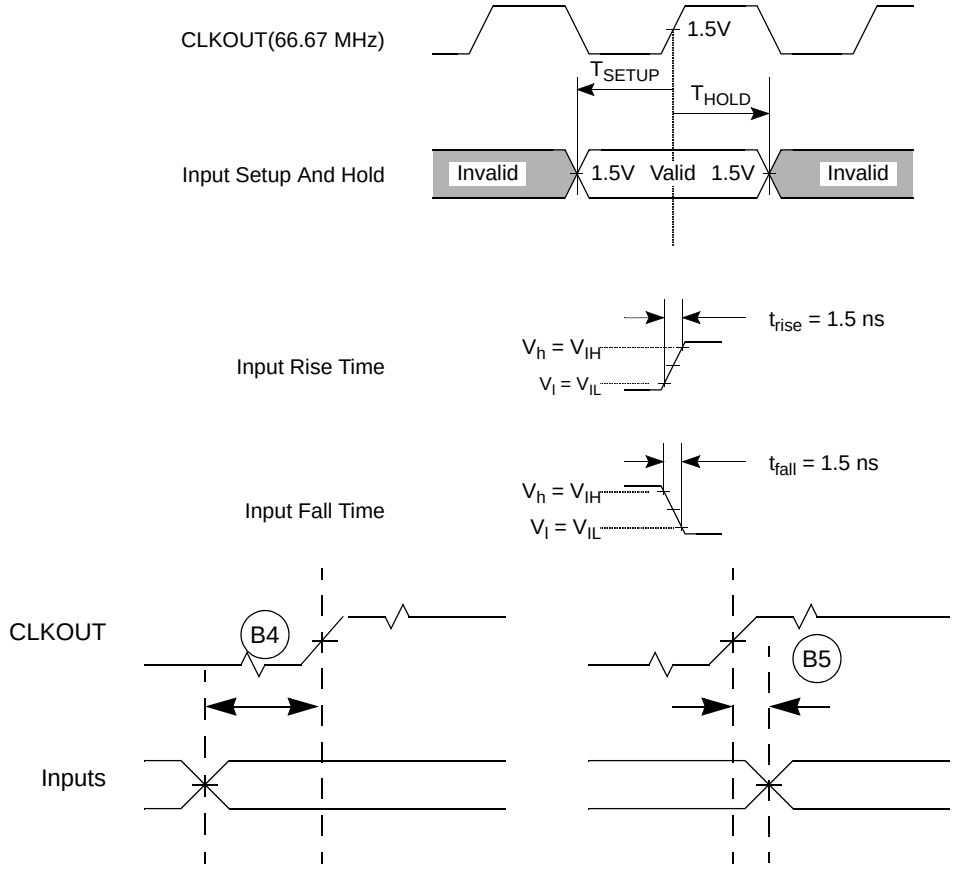


Figure 33-2. General Input Timing Requirements

33.9 Processor Bus Output Timing Specifications

Table 33-14 lists processor bus output timings.

Table 33-15. External Bus Output Timing Specifications

Name	Characteristic	Symbol	Min	Max	Unit
Control Outputs					
B6a	CLKOUT high to chip selects valid [1]	t_{CHCV}	—	$0.5t_{CYC}$ +10	ns
B6b	CLKOUT high to byte enables ($\overline{BS}$[3:0]) valid[2]	t_{CHBV}	—	$0.5t_{CYC}$ +10	ns
B6c	CLKOUT high to output enable ($\overline{OE}$) valid[3]	t_{CHOV}	—	$0.5t_{CYC}$ +10	ns
B7	CLKOUT high to control output ($\overline{BS}$[3:0], $\overline{OE}$) invalid	t_{CHCOI}	$0.5t_{CYC}$ + 2	—	ns
B7a	CLKOUT high to chip selects invalid	t_{CHCI}	$0.5t_{CYC}$ + 2	—	ns
Address and Attribute Outputs					

Table 33-15. External Bus Output Timing Specifications (continued)

Name	Characteristic	Symbol	Min	Max	Unit
B8	CLKOUT high to address (A[23:0]) and control ($\overline{TS}$, SIZ[1:0], $\overline{TIP}$, R/$\overline{W}$) valid	t_{CHAV}	—	10	ns
B9	CLKOUT high to address (A[23:0]) and control ($\overline{TS}$, SIZ[1:0], $\overline{TIP}$, R/$\overline{W}$) invalid	t_{CHAI}	2	—	ns
Data Outputs					
B11	CLKOUT high to data output (D[31:0]) valid	t_{CHDOV}	—	10	ns
B12	CLKOUT high to data output (D[31:0]) invalid	t_{CHDOI}	2	—	ns
B13	CLKOUT high to data output (D[31:0]) high impedance	t_{CHDOZ}	—	6	ns

1 $\overline{CS}n$ transitions after the falling edge of CLKOUT.
2 $\overline{BS}$ transitions after the falling edge of CLKOUT.
3 $\overline{OE}$ transitions after the falling edge of CLKOUT.

Read/write bus timings listed in Table 33-15 are shown in Figure 33-3, Figure 33-4, and Figure 33-5.

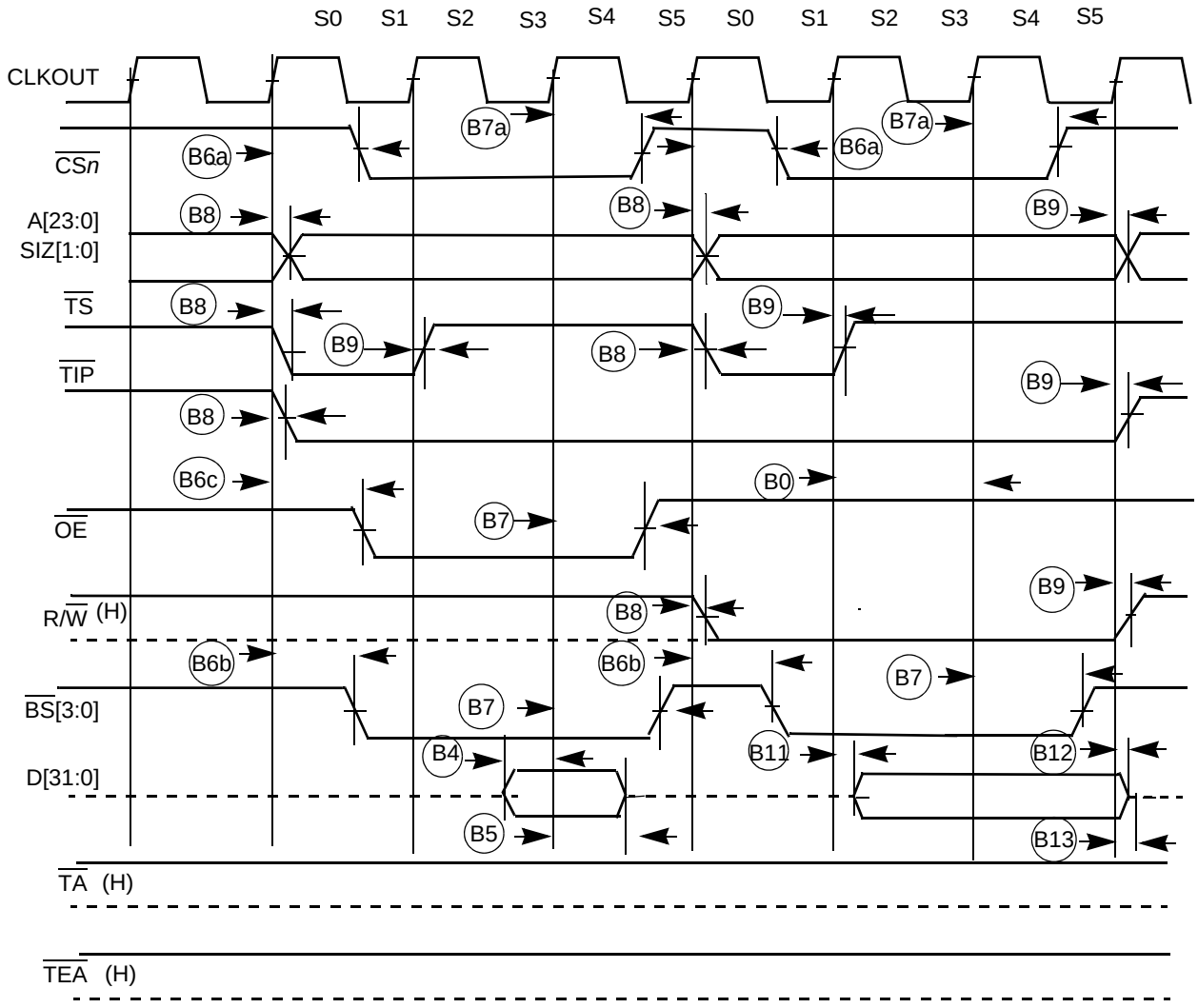


Figure 33-3. Read/Write (Internally Terminated) Timing

Figure 33-4 shows a bus cycle terminated by $\overline{TA}$ showing timings listed in Table 33-15.

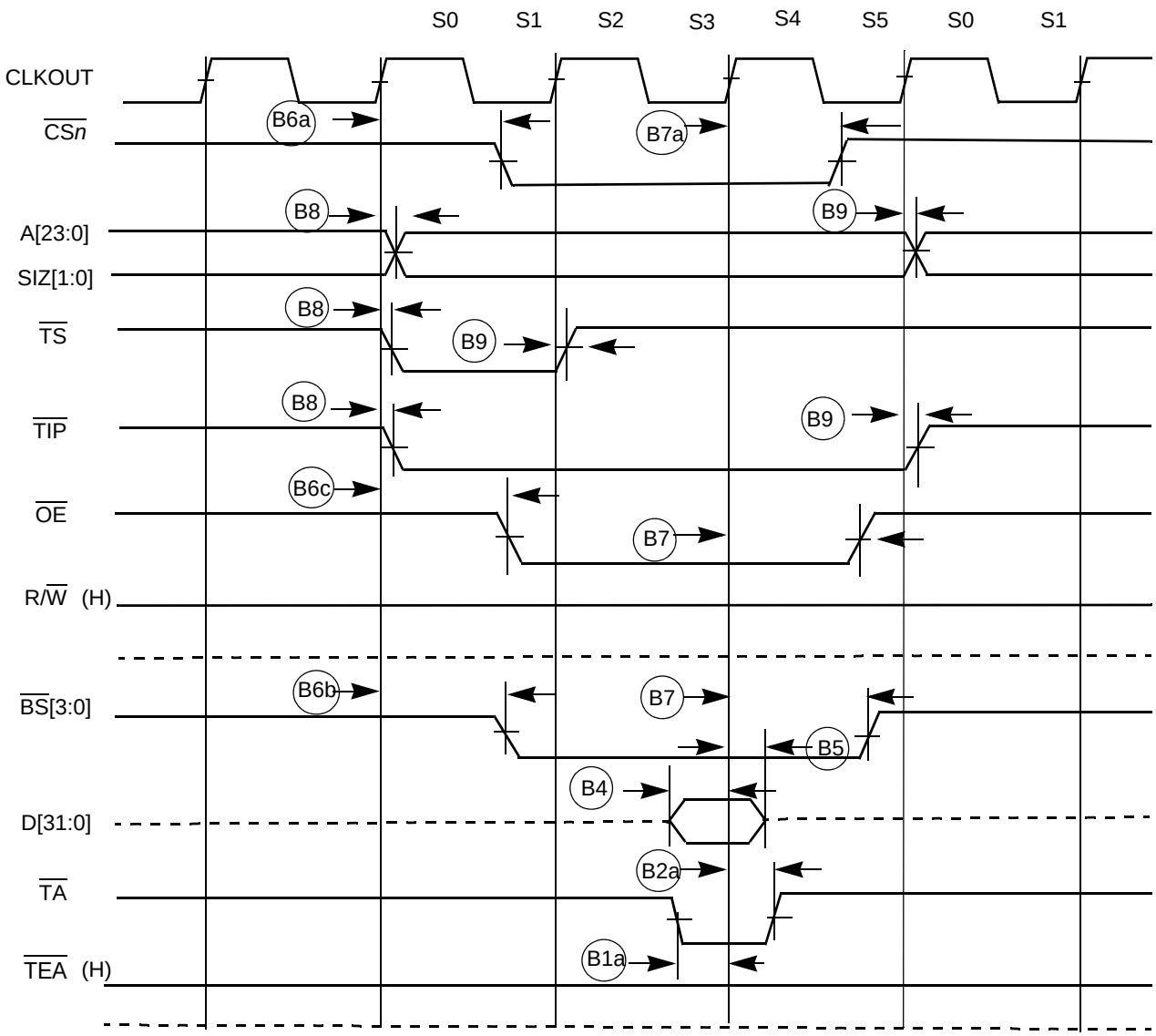


Figure 33-4. Read Bus Cycle Terminated by TA

Figure 33-5 shows a bus cycle terminated by TEA; it displays the timings listed in Table 33-15.

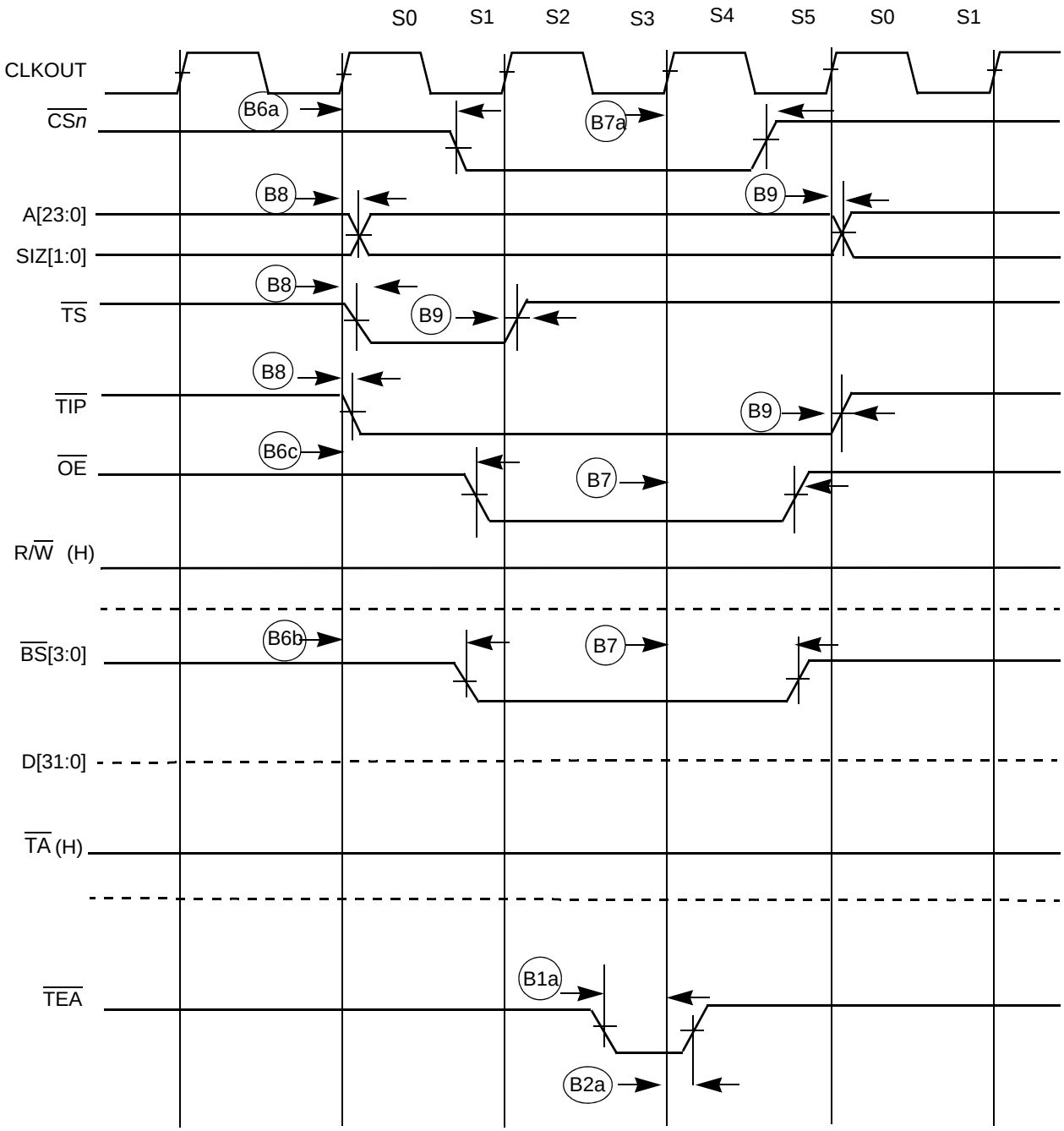
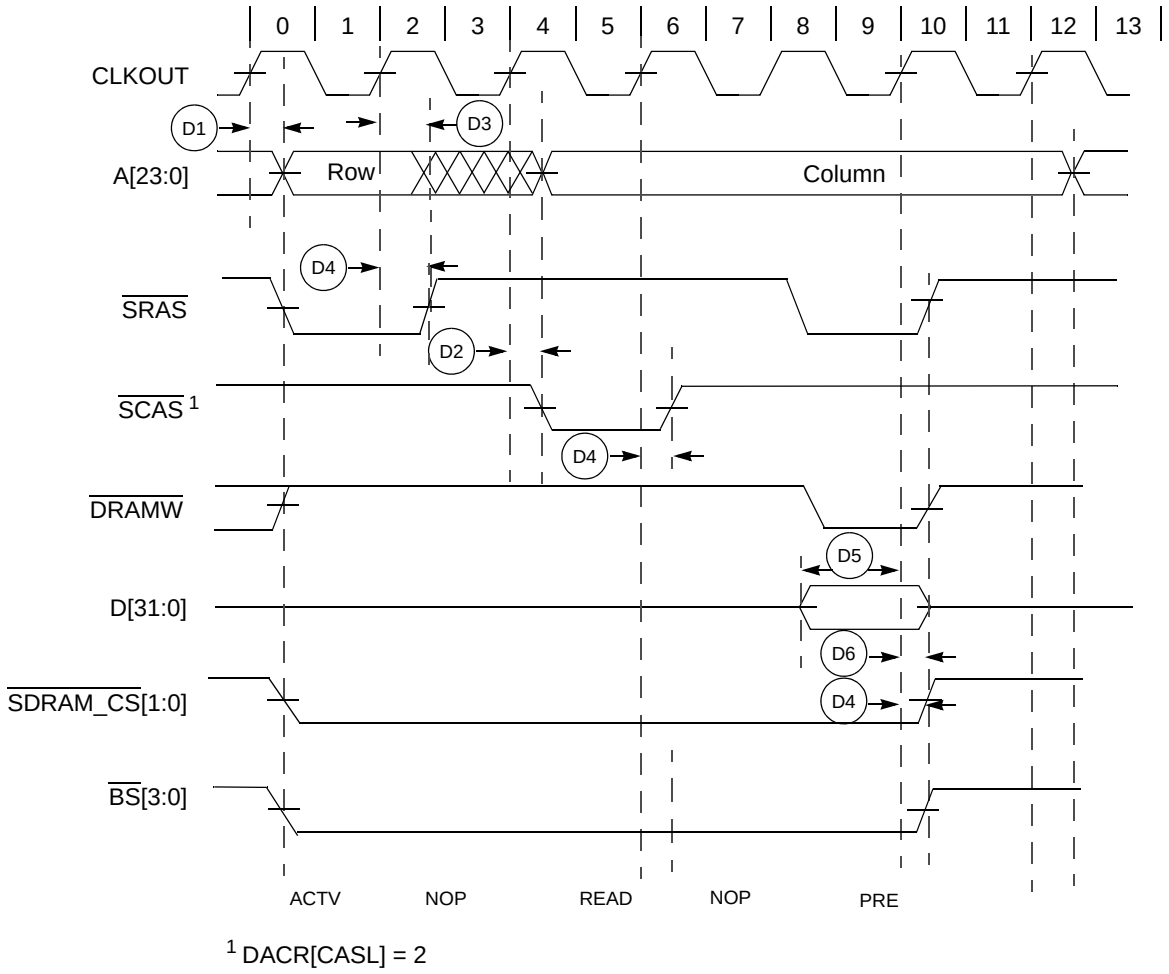


Figure 33-5. Read Bus Cycle Terminated by TEA

Figure 33-6 shows an SDRAM read cycle.



1 DACR[CASL] = 2

Figure 33-6. SDRAM Read Cycle

Table 33-16. SDRAM Timing

NUM	Characteristic[1]	Symbol	Min	Max	Unit
D1	CLKOUT high to SDRAM address valid	t_{CHDAV}	—	10	ns
D2	CLKOUT high to SDRAM control valid	t_{CHDCV}	—	10	ns
D3	CLKOUT high to SDRAM address invalid	t_{CHDAI}	2	—	ns
D4	CLKOUT high to SDRAM control invalid	t_{CHDCI}	2	—	ns
D5	SDRAM data valid to CLKOUT high	t_{DDVCH}	6	—	ns
D6	CLKOUT high to SDRAM data invalid	t_{CHDDI}	1	—	ns
D7[2]	CLKOUT high to SDRAM data valid	t_{CHDDVW}	—	10	ns
D8[2]	CLKOUT high to SDRAM data invalid	t_{CHDDIW}	2	—	ns

1 All timing specifications are based on taking into account, a 25pF load on the SDRAM output pins.

2 D7 and D8 are for write cycles only.

Figure 33-7 shows an SDRAM write cycle.

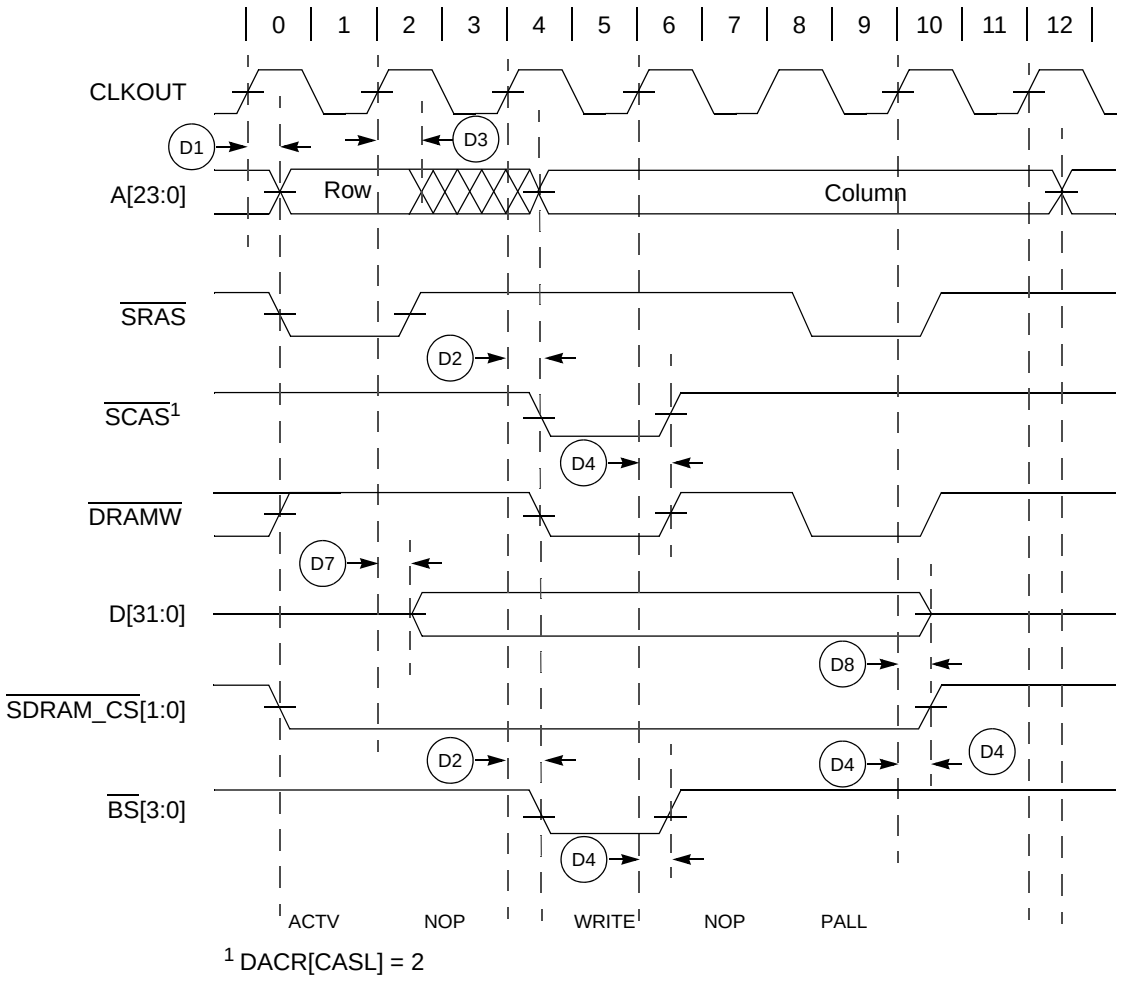


Figure 33-7. SDRAM Write Cycle

33.10 General Purpose I/O Timing

Table 33-17. GPIO Timing[1, 2]

(V_{DD} = 2.7 to 3.6 V, V_{SS} = 0 V, V_{DDH} = 5 V)

NUM	Characteristic	Symbol	Min	Max	Unit
G1 G2	CLKOUT High to GPIO Output Valid	t_{CHPOV}	—	12	ns
	CLKOUT High to GPIO Output Invalid	t_{CHPOI}	2	—	ns
G3 G4	GPIO Input Valid to CLKOUT High	t_{PVCH}	10	—	ns
	CLKOUT High to GPIO Input Invalid	t_{CHPI}	2	—	ns
G1a G2a	CLKOUT High to PQA/PQB Output Valid	t_{CHPAOV}	—	12	ns
	CLKOUT High to PQA/PQB Output Invalid	t_{CHPAOI}	2	—	ns
G3a G4a	PQA/PQB Input Valid to CLKOUT Low	t_{PAVCH}	10	—	ns
	CLKOUT Low to PQA/PQB Input Invalid	t_{CHPAI}	2	—	ns

1 GPIO pins include: Ports A-I, INT, SPI, SCI1/2 (including SCI functions), FlexCAN and Timer.
2 Because of long delays associated with the PQA/PQB pads, signals on the PQA/PQB pins will be updated on the following edge of the clock.

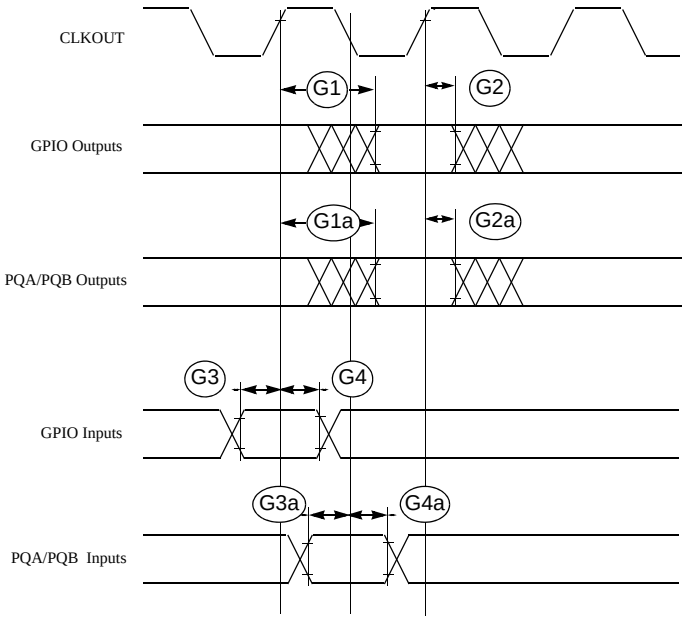


Figure 33-8. GPIO Timing

33.11 Reset and Configuration Override Timing

Table 33-18. Reset and Configuration Override Timing

(V_{DD} = 2.7 to 3.6 V, V_{SS} = 0 V)[1]

NUM	Characteristic	Symbol	Min	Max	Unit
R1	$\overline{\text{RSTI}}$ Input valid to CLKOUT High	t_{RVCH}	10	—	ns
R2	CLKOUT High to $\overline{\text{RSTI}}$ Input invalid	t_{CHRI}	2	—	ns
R3	$\overline{\text{RSTI}}$ Input valid Time [2]	t_{RIVT}	5	—	t_{CYC}
R4	CLKOUT High to $\overline{\text{RSTO}}$ Valid	t_{CHROV}	—	12	ns
R5	$\overline{\text{RSTO}}$ valid to Config. Overrides valid	t_{ROVCV}	0	—	ns
R6	Configuration Override Setup Time to $\overline{\text{RSTO}}$ invalid	t_{COS}	20	—	t_{CYC}
R7	Configuration Override Hold Time after $\overline{\text{RSTO}}$ invalid	t_{COH}	0	—	ns
R8	$\overline{\text{RSTO}}$ invalid to Configuration Override High Impedance	t_{ROICZ}	—	1	t_{CYC}

1 All AC timing is shown with respect to 50% V_{DD} levels unless otherwise noted.
2 During low-power STOP, the synchronizers for the $\overline{\text{RSTI}}$ input are bypassed and $\overline{\text{RSTI}}$ is asserted asynchronously to the system. Thus, $\overline{\text{RSTI}}$ must be held a minimum of 100 ns.

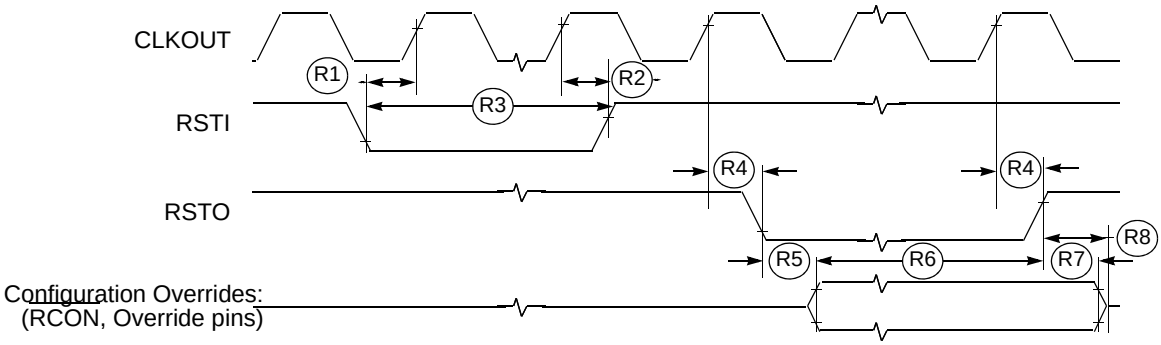


Figure 33-9. $\overline{\text{RSTI}}$ and Configuration Override Timing

33.12 I²C Input/Output Timing Specifications

Table 33-19 lists specifications for the I²C input timing parameters shown in Figure 33-10.

Table 33-19. I²C Input Timing Specifications between SCL and SDA

Num	Characteristic	Min	Max	Units
I1	Start condition hold time	2	—	Bus clocks
I2	Clock low period	8	—	Bus clocks
I3	SCL/SDA rise time (V_{IL} = 0.5 V to V_{IH} = 2.4 V)	—	1	mS
I4	Data hold time	0	—	ns
I5	SCL/SDA fall time (V_{IH} = 2.4 V to V_{IL} = 0.5 V)	—	1	mS
I6	Clock high time	4	—	Bus clocks
I7	Data setup time	0	—	ns
I8	Start condition setup time (for repeated start condition only)	2	—	Bus clocks
I9	Stop condition setup time	2	—	Bus clocks

Table 33-20 lists specifications for the I²C output timing parameters shown in Figure 33-10.

Table 33-20. I²C Output Timing Specifications between SCL and SDA

Num	Characteristic	Min	Max	Units
I1[1]	Start condition hold time	6	—	Bus clocks
I2[1]	Clock low period	10	—	Bus clocks
I3 [2]	SCL/SDA rise time (V_{IL} = 0.5 V to V_{IH} = 2.4 V)	—	—	µS
I4 [1]	Data hold time	7	—	Bus clocks
I5 [3]	SCL/SDA fall time (V_{IH} = 2.4 V to V_{IL} = 0.5 V)	—	3	ns
I6 [1]	Clock high time	10	—	Bus clocks
I7 [1]	Data setup time	2	—	Bus clocks

Table 33-20. I²C Output Timing Specifications between SCL and SDA (continued)

Num	Characteristic	Min	Max	Units
I8 [1]	Start condition setup time (for repeated start condition only)	20	—	Bus clocks
I9 [1]	Stop condition setup time	10	—	Bus clocks

1 Note: Output numbers depend on the value programmed into the IFDR; an IFDR programmed with the maximum frequency (IFDR = 0x20) results in minimum output timings as shown in Table 33-20. The I²C interface is designed to scale the actual data transition time to move it to the middle of the SCL low period. The actual position is affected by the prescale and division values programmed into the IFDR; however, the numbers given in Table 33-20 are minimum values.

2 Because SCL and SDA are open-collector-type outputs, which the processor can only actively drive low, the time SCL or SDA take to reach a high level depends on external signal capacitance and pull-up resistor values.

3 Specified at a nominal 50-pF load.

Figure 33-10 shows timing for the values in Table 33-19 and Table 33-20.

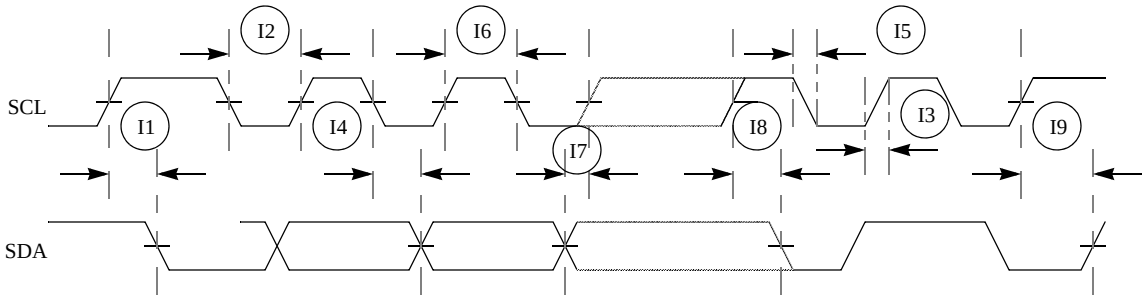


Figure 33-10. I²C Input/Output Timings

33.13 Fast Ethernet AC Timing Specifications

MII signals use TTL signal levels compatible with devices operating at either 5.0 V or 3.3 V.

NOTE

The MCF5214 and MCF5216 do not contain an FEC module.

33.13.1 MII Receive Signal Timing (ERXD[3:0], ERXDV, ERXER, and ERXCLK)

The receiver functions correctly up to a ERXCLK maximum frequency of 25 MHz +1%. There is no minimum frequency requirement. In addition, the processor clock frequency must exceed twice the ERXCLK frequency.

Table 33-21 lists MII receive channel timings.

Table 33-21. MII Receive Signal Timing

Num	Characteristic [1]	Min	Max	Unit
M1	ERXD[3:0], ERXDV, ERXER to ERXCLK setup	5	—	ns
M2	ERXCLK to ERXD[3:0], ERXDV, ERXER hold	5	—	ns
M3	ERXCLK pulse width high	35%	65%	ERXCLK period
M4	ERXCLK pulse width low	35%	65%	ERXCLK period

1 ERXDV, ERXCLK, and ERXD[0] have same timing in 10 Mbps 7-wire interface mode.

Figure 33-11 shows MII receive signal timings listed in Table 33-21.

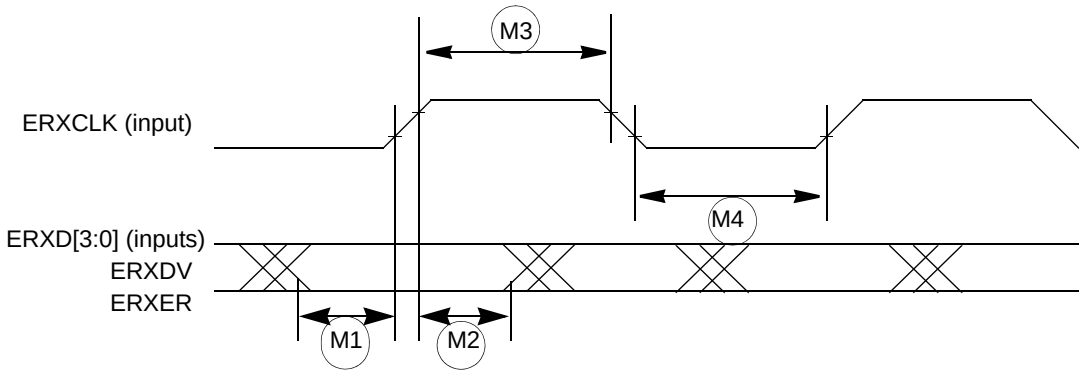


Figure 33-11. MII Receive Signal Timing Diagram

33.13.2 MII Transmit Signal Timing (ETXD[3:0], ETXEN, ETXER, ETXCLK)

The transmitter functions correctly up to a ETXCLK maximum frequency of 25 MHz +1%. In addition, the processor clock frequency must exceed twice the ETXCLK frequency.

Table 33-22 lists MII transmit channel timings.

Table 33-22. MII Transmit Signal Timing

Num	Characteristic[1]	Min	Max	Unit
M5	ETXCLK to ETXD[3:0], ETXEN, ETXER invalid	5	—	ns
M6	ETXCLK to ETXD[3:0], ETXEN, ETXER valid	—	25	ns
M7	ETXCLK pulse width high	35%	65%	ETXCLK period
M8	ETXCLK pulse width low	35%	65%	ETXCLK period

1 ETXCLK, ETXD0, and ETXEN have the same timing in 10 Mbit 7-wire interface mode.

Figure 33-12 shows MII transmit signal timings listed in Table 33-22.

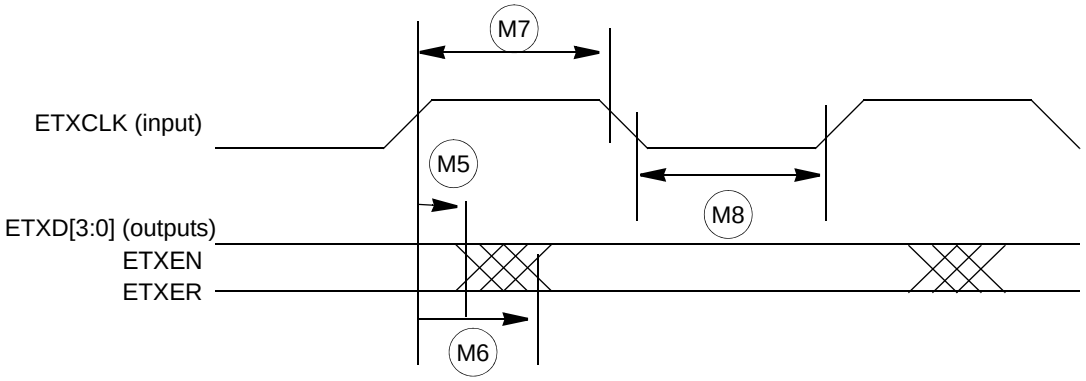


Figure 33-12. MII Transmit Signal Timing Diagram

33.13.3 MII Async Inputs Signal Timing (ECRS and ECOL)

Table 33-23 lists MII asynchronous inputs signal timing.

Table 33-23. MII Async Inputs Signal Timing

Num	Characteristic	Min	Max	Unit
M9[1]	ECRS, ECOL minimum pulse width	1.5	—	ETXCLK period

1 ECOL has the same timing in 10 Mbit 7-wire interface mode.

Figure 33-13 shows MII asynchronous input timings listed in Table 33-23.

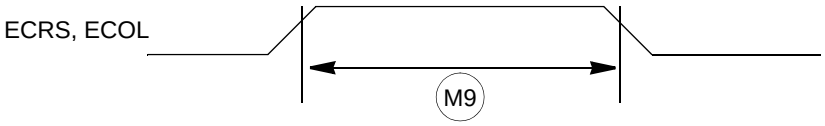


Figure 33-13. MII Async Inputs Timing Diagram

33.13.4 MII Serial Management Channel Timing (EMDIO and EMDC)

The FEC functions correctly with a maximum MDC frequency of 2.5 MHz. Table 33-24 lists MII serial management channel timings.

Table 33-24. MII Serial Management Channel Timing

Num	Characteristic	Min	Max	Unit
M10	EMDC falling edge to EMDIO output invalid (minimum propagation delay)	0	—	ns
M11	EMDC falling edge to EMDIO output valid (max prop delay)	—	25	ns
M12	EMDIO (input) to EMDC rising edge setup	10	—	ns
M13	EMDIO (input) to EMDC rising edge hold	0	—	ns
M14	EMDC pulse width high	40%	60%	MDC period
M15	EMDC pulse width low	40%	60%	MDC period

Figure 33-14 shows MII serial management channel timings listed in Table 33-24.

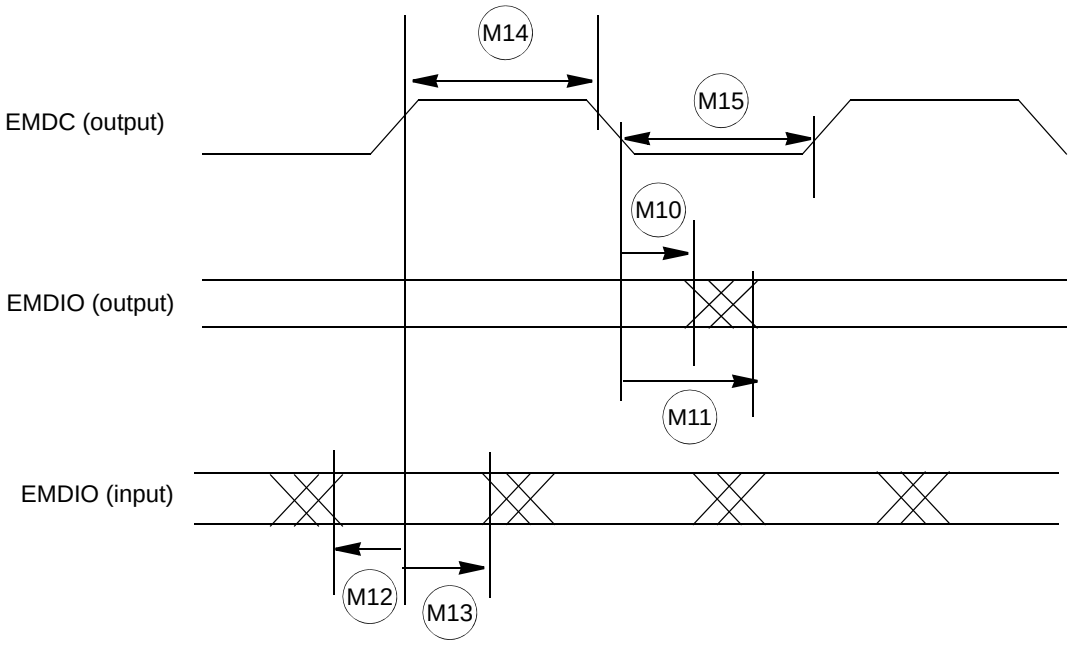


Figure 33-14. MII Serial Management Channel Timing Diagram

33.14 DMA Timer Module AC Timing Specifications

Table 33-25 lists timer module AC timings.

Table 33-25. Timer Module AC Timing Specifications

Name	Characteristic [1]	Min	Max	Unit
T1	DTIN0 / DTIN1 / DTIN2 / DTIN3 cycle time	3	—	t_{CYC}
T2	DTIN0 / DTIN1 / DTIN2 / DTIN3 pulse width	1	—	t_{CYC}

[1] All timing references to CLKOUT are given to its rising edge when bit 3 of the SDRAM control register is 0.

33.15 QSPI Electrical Specifications

Table 33-26 lists QSPI timings.

Table 33-26. QSPI Modules AC Timing Specifications

Name	Characteristic	Min	Max	Unit
QS1	QSPI_CS[3:0] to QSPI_CLK	1 × tcyc	510 × tcyc	ns
QS2	QSPI_CLK high to QSPI_DOUT valid.	—	12	ns
QS3	QSPI_CLK high to QSPI_DOUT invalid. (Output hold)	2	—	ns
QS4	QSPI_DIN to QSPI_CLK (Input setup)	10	—	ns
QS5	QSPI_DIN to QSPI_CLK (Input hold)	10	—	ns

The values in Table 33-26 correspond to Figure 33-15.

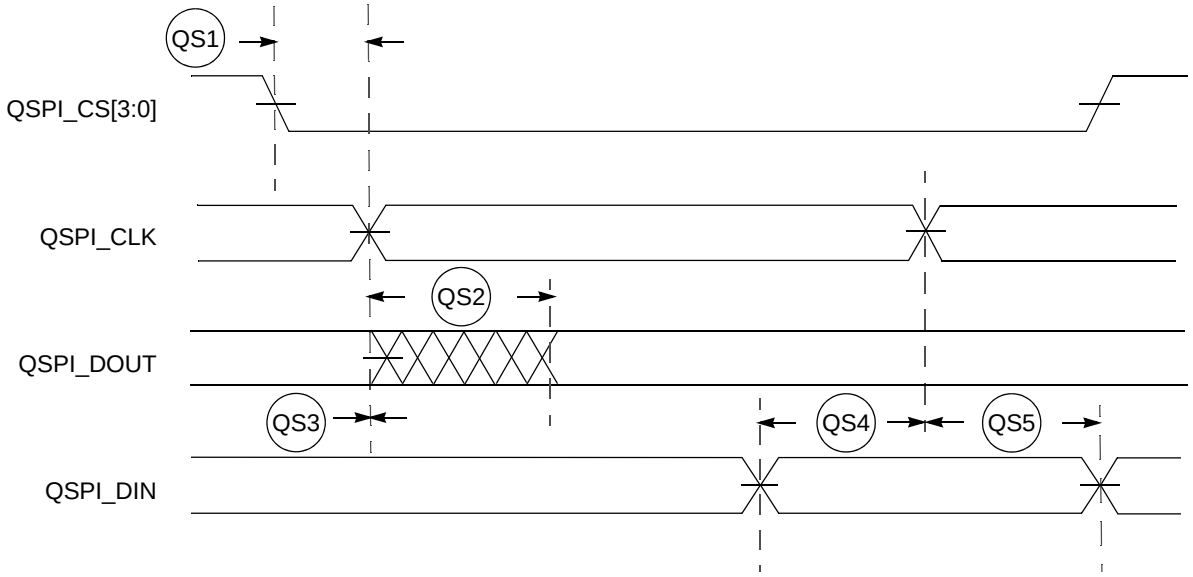


Figure 33-15. QSPI Timing

33.16 JTAG and Boundary Scan Timing

Table 33-27. JTAG and Boundary Scan Timing

Num	Characteristics[1]	Symbol	Min	Max	Unit
1	TCLK Frequency of Operation	f_{JCYC}	DC	1/4	f_{sys}
2	TCLK Cycle Period	t_{JCYC}	4	—	t_{CYC}
3	TCLK Clock Pulse Width	t_{JCW}	25.0	—	ns
4	TCLK Rise and Fall Times	t_{JCRF}	0.0	3.0	ns
5	Boundary Scan Input Data Setup Time to TCLK Rise	t_{BSDST}	5.0	—	ns
6	Boundary Scan Input Data Hold Time after TCLK Rise	t_{BSDHT}	25.0	—	ns
7	TCLK Low to Boundary Scan Output Data Valid	t_{BSDV}	0.0	30.0	ns
8	TCLK Low to Boundary Scan Output High Z	t_{BSDZ}	0.0	30.0	ns
9	TMS, TDI Input Data Setup Time to TCLK Rise	t_{TAPBST}	5.0	—	ns
10	TMS, TDI Input Data Hold Time after TCLK Rise	t_{TAPBHT}	10.0	—	ns
11	TCLK Low to TDO Data Valid	t_{TDODV}	0.0	25.0	ns
12	TCLK Low to TDO High Z	t_{TDODZ}	0.0	8.0	ns
13	$\overline{\text{TRST}}$ Assert Time	t_{TRSTAT}	100.0	—	ns
14	$\overline{\text{TRST}}$ Setup Time (Negation) to TCLK High	t_{TRSTST}	10.0	—	ns

1 JTAG_EN is expected to be a static signal. Hence, it is not associated with any timing

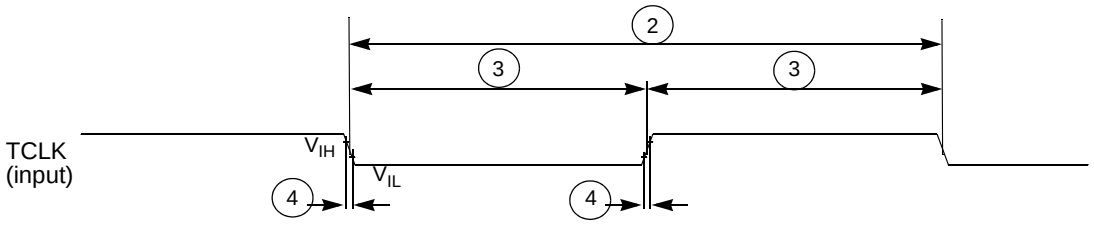


Figure 33-16. Test Clock Input Timing

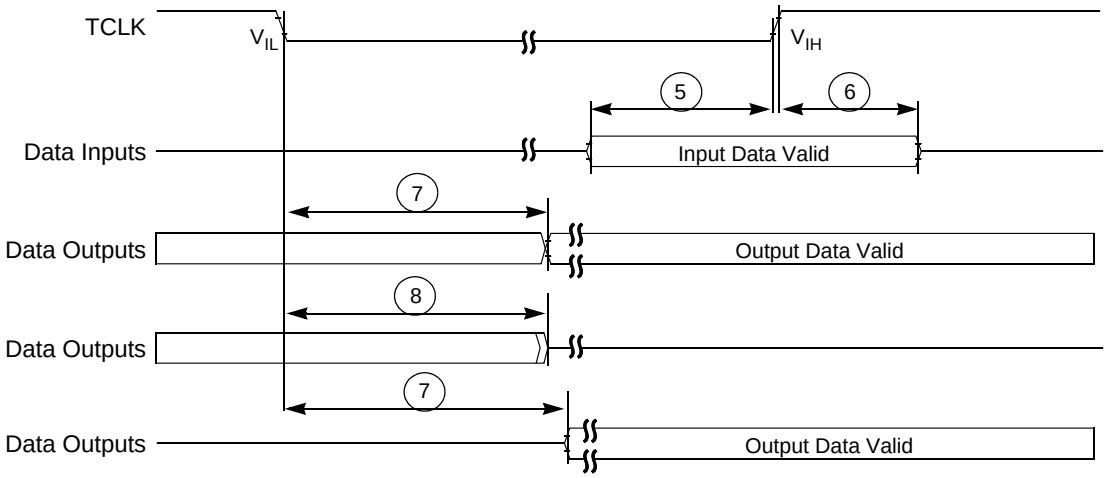


Figure 33-17. Boundary Scan (JTAG) Timing

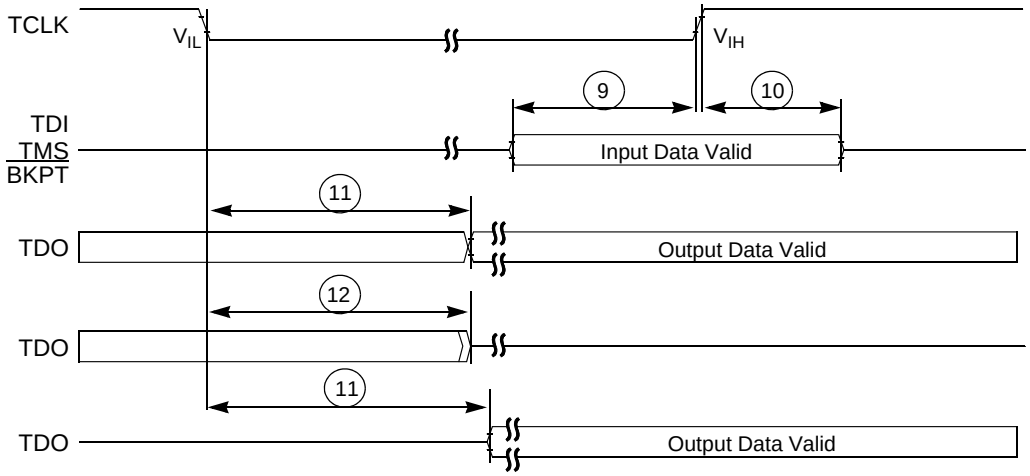


Figure 33-18. Test Access Port Timing

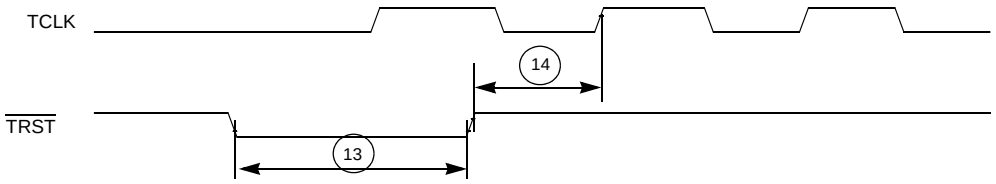


Figure 33-19. TRST Timing

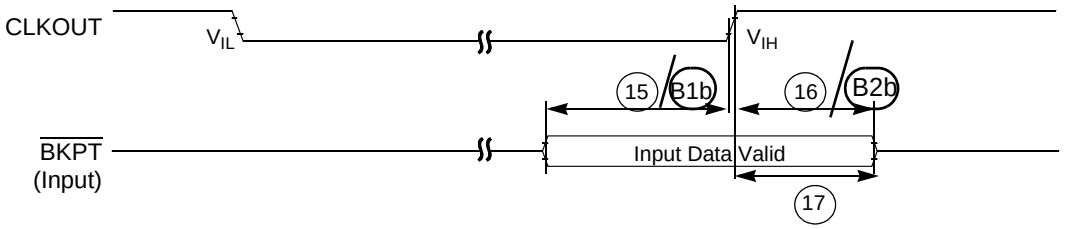


Figure 33-20. BKPT Timing

33.17 Debug AC Timing Specifications

Table 33-28 lists specifications for the debug AC timing parameters shown in Figure 33-22.

Table 33-28. Debug AC Timing Specification

Num	Characteristic	Min	Max	Units
D1	PST, DDATA to TCLK setup	5		ns
D2	TCLK to PST, DDATA hold	2		ns
D3	DSI-to-DSCLK setup	1		CLKOUT cycles
D4 [1]	DSCLK-to-DSO hold	4		CLKOUT cycles
D5	DSCLK cycle time	5		CLKOUT cycles
D6	BKPT input data setup time to CLKOUT Rise	5.0		ns
D7	BKPT input data hold time to CLKOUT Rise	2.0		ns
D8	CLKOUT high to BKPT high Z	0.0	10.0	ns

1 DSCLK and DSI are synchronized internally. D4 is measured from the synchronized DSCLK input relative to the rising edge of CLKOUT.

Figure 33-21 shows real-time trace timing for the values in Table 33-28.

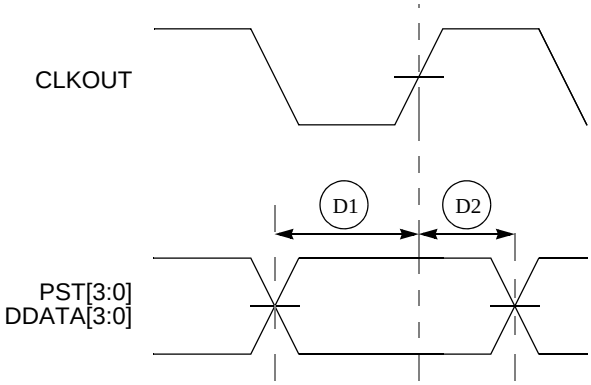


Figure 33-21. Real-Time Trace AC Timing

Figure 33-22 shows BDM serial port AC timing for the values in Table 33-28.

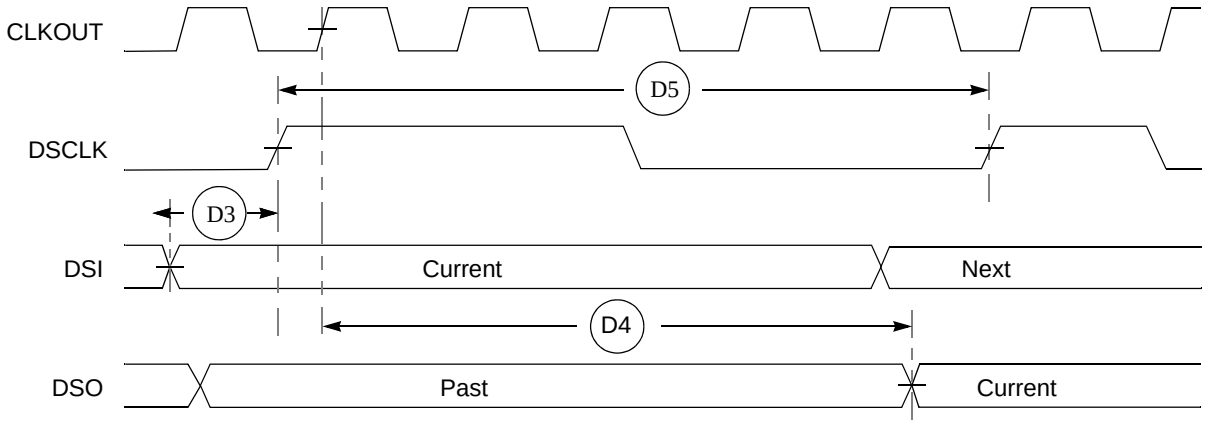


Figure 33-22. BDM Serial Port AC Timing

Appendix A
Register Memory Map

Table A-1 summarizes the address, name, and byte assignment for registers within the 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 @ 0x807	MAC Accumulator 0, 1 extension bytes	ACCext01	16
CPU @ 0x808	MAC Accumulator 2, 3 extension bytes	ACCext23	16
CPU @ 0x809	MAC Accumulator 1	ACC1	16
CPU @ 0x80A	MAC Accumulator 2	ACC2	16
CPU @ 0x80B	MAC Accumulator 3	ACC3	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
IPSBAR + 0x00_0000	System Control Module	64 bytes
IPSBAR + 0x00_0040	SDRAM Controller	64 bytes
IPSBAR + 0x00_0080	Chip Selects	128 bytes
IPSBAR + 0x00_0100	DMA (Channel 0)	64 bytes
IPSBAR + 0x00_0140	DMA (Channel 1)	64 bytes
IPSBAR + 0x00_0180	DMA (Channel 2)	64 bytes
IPSBAR + 0x00_01C0	DMA (Channel 3)	64 bytes
IPSBAR + 0x00_0200	UART0	64 bytes
IPSBAR + 0x00_0240	UART1	64 bytes
IPSBAR + 0x00_0280	UART2	64 bytes
IPSBAR + 0x00_0300	I^2C	64 bytes
IPSBAR + 0x00_0340	QSPI	64 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_0C00	Interrupt Controller 0	256 bytes
IPSBAR + 0x00_0D00	Interrupt Controller 1	256 bytes
IPSBAR + 0x00_0F00	Global Interrupt Acknowledge Cycles	256 bytes
IPSBAR + 0x00_1000	Fast Ethernet Controller (Registers and MIB RAM) (Reserved for MCF5214 and MCF5216)	1 K
IPSBAR + 0x00_1400	Fast Ethernet Controller (FIFO Memory) (Reserved for MCF5214 and MCF5216)	1K
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	Watchdog Timer	64K
IPSBAR + 0x15_0000	Programmable Interval Timer 0	64K
IPSBAR + 0x16_0000	Programmable Interval Timer 1	64K
IPSBAR + 0x17_0000	Programmable Interval Timer 2	64K
IPSBAR + 0x18_0000	Programmable Interval Timer 3	64K
IPSBAR + 0x19_0000	QADC	64K
IPSBAR + 0x1A_0000	General Purpose Timer A	64K
IPSBAR + 0x1B_0000	General Purpose Timer B	64K

Table A-2. Module Memory Map Overview (continued)

Address	Module	Size
IPSBAR + 0x1C_0000	FlexCAN	64K
IPSBAR + 0x1D_0000	CFM (Flash) Control Registers	64K
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	Bus Master Park Register	MPARK	32
IPSBAR + 0x020	Master Privilege Register	MPR	32
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
SDRAMC Registers			
IPSBAR + 0x040	DRAM Control Register	DCR	16
IPSBAR + 0x048	DRAM Address and Control Register 0	DACR0	32
IPSBAR + 0x04C	DRAM Mask Register Block 0	DMR0	32
IPSBAR + 0x050	DRAM Address and Control Register 1	DACR1	32

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x054	DRAM Mask Register Block 1	DMR1	32
Chip Select Registers			
IPSBAR + 0x080	Chip Select Address Register 0	CSAR0	16
IPSBAR + 0x084	Chip Select Mask Register 0	CSMR0	32
IPSBAR + 0x08A	Chip Select Control Register 0	CSCR0	16
IPSBAR + 0x08C	Chip Select Address Register 1	CSAR1	16
IPSBAR + 0x090	Chip Select Mask Register 1	CSMR1	32
IPSBAR + 0x096	Chip Select Control Register 1	CSCR1	16
IPSBAR + 0x098	Chip Select Address Register 2	CSAR2	16
IPSBAR + 0x09C	Chip Select Mask Register 2	CSMR2	32
IPSBAR + 0x0A2	Chip Select Control Register 2	CSCR2	16
IPSBAR + 0x0A4	Chip Select Address Register 2	CSAR3	16
IPSBAR + 0x0A8	Chip Select Mask Register 3	CSMR3	32
IPSBAR + 0x0AE	Chip Select Control Register 3	CSCR3	16
IPSBAR + 0x0B0	Chip Select Address Register 4	CSAR4	16
IPSBAR + 0x0B4	Chip Select Mask Register 4	CSMR4	32
IPSBAR + 0x0BA	Chip Select Control Register 4	CSCR4	16
IPSBAR + 0x0BC	Chip Select Address Register 5	CSAR5	16
IPSBAR + 0x0C0	Chip Select Mask Register 5	CSMR5	32
IPSBAR + 0x0C6	Chip Select Control Register 5	CSCR5	16
IPSBAR + 0x0C8	Chip Select Address Register 6	CSAR6	16
IPSBAR + 0x0CC	Chip Select Mask Register 6	CSMR6	32
IPSBAR + 0x0D2	Chip Select Control Register 6	CSCR6	16
DMA Registers			
IPSBAR + 0x100	Source Address Register 0	SAR0	32
IPSBAR + 0x104	Destination Address Register 0	DAR0	32
IPSBAR + 0x108	DMA Control Register 0	DCR0	32
IPSBAR + 0x10C	Byte Count Register 0	BCR0[1]	32
IPSBAR + 0x110	DMA Status Register 0	DSR0	8
IPSBAR + 0x140	Source Address Register 1	SAR1	32
IPSBAR + 0x144	Destination Address Register 1	DAR1	32
IPSBAR + 0x148	DMA Control Register 1	DCR1	32

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x14C	Byte Count Register 1	BCR1[1]	32
IPSBAR + 0x150	DMA Status Register 1	DSR1	8
IPSBAR + 0x180	Source Address Register 2	SAR2	32
IPSBAR + 0x184	Destination Address Register 2	DAR2	32
IPSBAR + 0x188	DMA Control Register 2	DCR2	32
IPSBAR + 0x18C	Byte Count Register 2	BCR2[1]	32
IPSBAR + 0x190	DMA Status Register 2	DSR2	8
IPSBAR + 0x1C0	Source Address Register 3	SAR3	32
IPSBAR + 0x1C4	Destination Address Register 3	DAR3	32
IPSBAR + 0x1C8	DMA Control Register 3	DCR3	32
IPSBAR + 0x1CC	Byte Count Register 3	BCR3[1]	32
IPSBAR + 0x1D0	DMA Status Register 3	DSR3	8
UART Registers			
IPSBAR + 0x200	UART Mode Register 0[2]	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
IPSBAR + 0x234	(Read) UART Input Port Register 0	UIP0	8
	(Write) Reserved		8

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
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[2]	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[2]	UMR12, UMR22	8
IPSBAR + 0x284	(Read) UART Status Register 2	USR2	8
	(Write) UART Clock Select Register 2[1]	UCSR2	8

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

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
DMA Timer Registers			
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

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0xC18	Interrupt Level Request Register 0	ILRR0	8
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

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0xC61	Interrupt Control Register 0-33	ICR033	8
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

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0xCEC	Level 3 Interrupt Acknowledge Register 0	L3IACKR0	8
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
Interrupt Controller 1			
IPSBAR + 0xD00	Interrupt Pending Register High 1	IPRH1	32
IPSBAR + 0xD04	Interrupt Pending Register Low 1	IPRL1	32
IPSBAR + 0xD08	Interrupt Mask Register High 1	IMRH1	32
IPSBAR + 0xD0C	Interrupt Mask Register Low 1	IMRL1	32
IPSBAR + 0xD10	Interrupt Force Register High 1	INTFRCH1	32
IPSBAR + 0xD14	Interrupt Force Register Low 1	INTFRCL1	32
IPSBAR + 0xD18	Interrupt Level Request Register 1	ILRR1	8
IPSBAR + 0xD19	Interrupt Acknowledge Level and Priority Register 1	IACKLPR1	8
IPSBAR + 0xD48	Interrupt Control Register 1-08	ICR108	8
IPSBAR + 0xD49	Interrupt Control Register 1-09	ICR109	8
IPSBAR + 0xD4A	Interrupt Control Register 1-10	ICR110	8
IPSBAR + 0xD4B	Interrupt Control Register 1-11	ICR111	8
IPSBAR + 0xD4C	Interrupt Control Register 1-12	ICR112	8
IPSBAR + 0xD4D	Interrupt Control Register 1-13	ICR113	8
IPSBAR + 0xD4E	Interrupt Control Register 1-14	ICR114	8
IPSBAR + 0xD4F	Interrupt Control Register 1-15	ICR115	8
IPSBAR + 0xD50	Interrupt Control Register 1-16	ICR116	8
IPSBAR + 0xD51	Interrupt Control Register 1-17	ICR117	8
IPSBAR + 0xD52	Interrupt Control Register 1-18	ICR118	8
IPSBAR + 0xD53	Interrupt Control Register 1-19	ICR119	8
IPSBAR + 0xD54	Interrupt Control Register 1-20	ICR120	8
IPSBAR + 0xD55	Interrupt Control Register 1-21	ICR121	8
IPSBAR + 0xD56	Interrupt Control Register 1-22	ICR122	8
IPSBAR + 0xD57	Interrupt Control Register 1-23	ICR123	8
IPSBAR + 0xD58	Interrupt Control Register 1-24	ICR124	8
IPSBAR + 0xD59	Interrupt Control Register 1-25	ICR125	8

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0xD5A	Interrupt Control Register 1-26	ICR126	8
IPSBAR + 0xDE0	Software Interrupt Acknowledge Register 1	SWACKR1	8
IPSBAR + 0xDE4	Level 1 Interrupt Acknowledge Register 1	L1IACKR1	8
IPSBAR + 0xDE8	Level 2 Interrupt Acknowledge Register 1	L2IACKR1	8
IPSBAR + 0xDEC	Level 3 Interrupt Acknowledge Register 1	L3IACKR1	8
IPSBAR + 0xDF0	Level 4 Interrupt Acknowledge Register 1	L4IACKR1	8
IPSBAR + 0xDF4	Level 5 Interrupt Acknowledge Register 1	L5IACKR1	8
IPSBAR + 0xDF8	Level 6 Interrupt Acknowledge Register 1	L6IACKR1	8
IPSBAR + 0xDFC	Level 7 Interrupt Acknowledge Register 1	L7IACKR1	8
Global Interrupt Acknowledge Cycle Registers			
IPSBAR + 0xFE0	Global Software Interrupt Acknowledge Register	GSWACKR	8
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
FEC Registers (Reserved for MCF5214 and MCF5216)			
IPSBAR + 0x1004	Interrupt Event Register	EIR	32
IPSBAR + 0x1008	Interrupt Mask Register	EIMR	32
IPSBAR + 0x1010	Receive Descriptor Active Register	RDAR	32
IPSBAR + 0x1014	Transmit Descriptor Active Register	XDAR	32
IPSBAR + 0x1024	Ethernet Control Register	ECR	32
IPSBAR + 0x1040	MII Data Register	MDATA	32
IPSBAR + 0x1044	MII Speed Control Register	MSCR	32
IPSBAR + 0x1064	MIB Control/Status Register	MIBC	32
IPSBAR + 0x1084	Receive Control Register	RCR	32
IPSBAR + 0x10C4	Transmit Control Register	TCR	32
IPSBAR + 0x10E4	Physical Address Low	PALR	32
IPSBAR + 0x10E8	Physical Address High+ Type Field	PAUR	32
IPSBAR + 0x10EC	Opcode + Pause Duration	OPD	32

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x1118	Upper 32 bits of individual hash table	IAUR	32
IPSBAR + 0x111C	Lower 32 bits of individual hash table	IALR	32
IPSBAR + 0x1120	Upper 32-bits of group hash table	GAUR	32
IPSBAR + 0x1124	Lower 32-bits of group hash table	GALR	32
IPSBAR + 0x1144	Transmit FIFO Watermark	TFWR	32
IPSBAR + 0x114C	FIFO Receive Bound Register	FRBR	32
IPSBAR + 0x1150	FIFO Receive Start Address	FRSR	32
IPSBAR + 0x1180	Pointer to Receive Descriptor Ring	ERDSR	32
IPSBAR + 0x1184	Pointer to Transmit Descriptor Ring	ETDSR	32
IPSBAR + 0x1188	Maximum Receive Buffer Size	EMRBR	32
IPSBAR + 0x1200-0x13FF	RAM used to store management counters	MIB_RAM	32
GPIO Registers			
IPSBAR + 0x10_0000	Port A Output Data Register	PORTA	8
IPSBAR + 0x10_0001	Port B Output Data Register	PORTB	8
IPSBAR + 0x10_0002	Port C Output Data Register	PORTC	8
IPSBAR + 0x10_0003	Port D Output Data Register	PORTD	8
IPSBAR + 0x10_0004	Port E Output Data Register	PORTE	8
IPSBAR + 0x10_0005	Port F Output Data Register	PORTF	8
IPSBAR + 0x10_0006	Port G Output Data Register	PORTG	8
IPSBAR + 0x10_0007	Port H Output Data Register	PORTH	8
IPSBAR + 0x10_0008	Port J Output Data Register	PORTJ	8
IPSBAR + 0x10_0009	Port DD Output Data Register	PORTDD	8
IPSBAR + 0x10_000A	Port EH Output Data Register (Reserved for MCF5214 and MCF5216)	PORTEH	8
IPSBAR + 0x10_000B	Port EL Output Data Register	PORTEL	8

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x10_000C	Port AS Output Data Register	PORTAS	8
IPSBAR + 0x10_000D	Port QS Output Data Register	PORTQS	8
IPSBAR + 0x10_000E	Port SD Output Data Register	PORTSD	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_0014	Port A Data Direction Register	DDRA	8
IPSBAR + 0x10_0015	Port B Data Direction Register	DDRB	8
IPSBAR + 0x10_0016	Port C Data Direction Register	DDRC	8
IPSBAR + 0x10_0017	Port D Data Direction Register	DDRD	8
IPSBAR + 0x10_0018	Port E Data Direction Register	DDRE	8
IPSBAR + 0x10_0019	Port F Data Direction Register	DDRF	8
IPSBAR + 0x10_001A	Port G Data Direction Register	DDRG	8
IPSBAR + 0x10_001B	Port H Data Direction Register	DDRH	8
IPSBAR + 0x10_001C	Port J Data Direction Register	DDRJ	8
IPSBAR + 0x10_001D	Port DD Data Direction Register	DDRDD	8
IPSBAR + 0x10_001E	Port EH Data Direction Register (Reserved for MCF5214 and MCF5216)	DDREH	8
IPSBAR + 0x10_001F	Port EL Data Direction Register	DDREL	8
IPSBAR + 0x10_0020	Port AS Data Direction Register	DDRAS	8
IPSBAR + 0x10_0021	Port QS Data Direction Register	DDRQS	8

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x10_0022	Port SD Data Direction Register	DDRSD	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_0028	Port A Pin Data/Set Data Register	PORTAP/ SETA	8
IPSBAR + 0x10_0029	Port B Pin Data/Set Data Register	PORTBP/ SETB	8
IPSBAR + 0x10_002A	Port C Pin Data/Set Data Register	PORTCP/ SETC	8
IPSBAR + 0x10_002B	Port D Pin Data/Set Data Register	PORTDP/ SETD	8
IPSBAR + 0x10_002C	Port E Pin Data/Set Data Register	PORTEP/ SETE	8
IPSBAR + 0x10_002D	Port F Pin Data/Set Data Register	PORTFP/ SETF	8
IPSBAR + 0x10_002E	Port G Pin Data/Set Data Register	PORTGP/ SETG	8
IPSBAR + 0x10_002F	Port H Pin Data/Set Data Register	PORTHP/ SETH	8
IPSBAR + 0x10_0030	Port J Pin Data/Set Data Register	PORTJP/ SETJ	8
IPSBAR + 0x10_0031	Port DD Pin Data/Set Data Register	PORTDDP/ SETDD	8
IPSBAR + 0x10_0032	Port EH Pin Data/Set Data RegisterPort EH Data Direction Register (Reserved for MCF5214 and MCF5216)	PORTEHP/ SETEH	8
IPSBAR + 0x10_0033	Port EL Pin Data/Set Data Register	PORTELP/ SETEL	8
IPSBAR + 0x10_0034	Port AS Pin Data/Set Data Register	PORTASP/ SETAS	8
IPSBAR + 0x10_0035	Port QS Pin Data/Set Data Register	PORTQSP/ SETQS	8
IPSBAR + 0x10_0036	Port SD Pin Data/Set Data Register	PORTSDP/ SETSD	8
IPSBAR + 0x10_0037	Port TC Pin Data/Set Data Register	PORTTCP/ SETTC	8

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
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_003C	Port A Clear Output Data Register	CLRA	8
IPSBAR + 0x10_003D	Port B Clear Output Data Register	CLRB	8
IPSBAR + 0x10_003E	Port C Clear Output Data Register	CLRC	8
IPSBAR + 0x10_003F	Port D Clear Output Data Register	CLRD	8
IPSBAR + 0x10_0040	Port E Clear Output Data Register	CLRE	8
IPSBAR + 0x10_0041	Port F Clear Output Data Register	CLRF	8
IPSBAR + 0x10_0042	Port G Clear Output Data Register	CLRG	8
IPSBAR + 0x10_0043	Port H Clear Output Data Register	CLRH	8
IPSBAR + 0x10_0044	Port J Clear Output Data Register	CLRJ	8
IPSBAR + 0x10_0045	Port DD Clear Output Data Register	CLRDD	8
IPSBAR + 0x10_0046	Port EH Clear Output Data RegisterPort EH Data Direction Register (Reserved for MCF5214 and MCF5216)	CLREH	8
IPSBAR + 0x10_0047	Port EL Clear Output Data Register	CLREL	8
IPSBAR + 0x10_0048	Port AS Clear Output Data Register	CLRAS	8
IPSBAR + 0x10_0049	Port QS Clear Output Data Register	CLRQS	8
IPSBAR + 0x10_004A	Port SD Clear Output Data Register	CLRSD	8
IPSBAR + 0x10_004B	Port TC Clear Output Data Register	CLRTC	8
IPSBAR + 0x10_004C	Port TD Clear Output Data Register	CLRTD	8
IPSBAR + 0x10_004D	Port UA Clear Output Data Register	CLRUA	8

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x10_0050	Port B, C, and D Pin Assignment Register	PBCDPAR	8
IPSBAR + 0x10_0051	Port F Pin Assignment Register	PFPAR	8
IPSBAR + 0x10_0052	Port E Pin Assignment Register	PEPAR	16
IPSBAR + 0x10_0054	Port J Pin Assignment Register	PJPAR	8
IPSBAR + 0x10_0055	Port SD Pin Assignment Register	PSDPAR	8
IPSBAR + 0x10_0056	Port AS Pin Assignment Register	PASPAR	16
IPSBAR + 0x10_0058	Port EH and EL Pin Assignment Register (Port EH is not present on MCF5214 and MCF5216)	PEHLPAR	8
IPSBAR + 0x10_0059	Port QS Pin Assignment Register	PQSPAR	8
IPSBAR + 0x10_005A	Port TC Pin Assignment Register	PTCPAR	8
IPSBAR + 0x10_005B	Port TD Pin Assignment Register	PTDPAR	8
IPSBAR + 0x10_005C	Port UA Pin Assignment Register	PUAPAR	8
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

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
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
Watchdog Timer Registers			
IPSBAR + 0x14_0000	Watchdog Control Register	WCR	16
IPSBAR + 0x14_0002	Watchdog Modulus Register	WMR	16
IPSBAR + 0x14_0004	Watchdog Count Register	WCNTR	16
IPSBAR + 0x14_0006	Watchdog Service Register	WSR	16
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
Programmable Interrupt Timer 2 Registers			
IPSBAR + 0x17_0000	PIT Control and Status Register 2	PCSR 2	16

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x17_0002	PIT Modulus Register 2	PMR 2	16
IPSBAR + 0x17_0004	PIT Count Register 2	PCNTR 2	16
Programmable Interrupt Timer 3 Registers			
IPSBAR + 0x18_0000	PIT Control and Status Register 3	PCSR 3	16
IPSBAR + 0x18_0002	PIT Modulus Register 3	PMR 3	16
IPSBAR + 0x18_0004	PIT Count Register 3	PCNTR 3	16
QADC Registers			
0x19_0000	QADC Module Configuration Register	QADCMCR	16
0x19_0006	Port QA Data Register	PORTQA	8
IPSBAR + 0x19_0007	Port QB Data Register	PORTQB	8
0x19_0008	Port QA Data Direction Register	DDRQA	8
IPSBAR + 0x19_0009	Port QB Data Direction Register	DDRQB	8
0x19_000A	QADC Control Register 0	QACR0	16
0x19_000C	QADC Control Register 1	QACR1	16
0x19_000E	QADC Control Register 2	QACR2	16
0x19_0010	QADC Status Register 0	QASR0	16
0x19_0012	QADC Status Register 1	QASR1	16
0x19_0200–0x19_027E	Conversion Command Word Table	CCW0–CCW63	64x16
0x19_0280–0x19_02FE	Right Justified, Unsigned Result Register	RJURR0–RJURR63	64x16
0x19_0300–0x19_037E	Left Justified, Signed Result Register	LJSRR0–LJSRR63	64x16
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

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
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
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

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
General Purpose Timer B Registers			
IPSBAR + 0x1B_0000	GPTB IC/OC Select Register	GPTBIOS	8
IPSBAR + 0x1B_0001	GPTB Compare Force Register	GPTBCFORC	8
IPSBAR + 0x1B_0002	GPTB Output Compare 3 Mask Register	GPTBOC3M	8
IPSBAR + 0x1B_0003	GPTB Output Compare 3 Data Register	GPTBOC3D	8
IPSBAR + 0x1B_0004	GPTB Counter Register	GPTBCNT	16
IPSBAR + 0x1B_0006	GPTB System Control Register 1	GPTBSCR1	8
IPSBAR + 0x1B_0008	GPTB Toggle-on-Overflow Register	GPTBTOV	8
IPSBAR + 0x1B_0009	GPTB Control Register 1	GPTBCTL1	8
IPSBAR + 0x1B_000B	GPTB Control Register 2	GPTBCTL2	8
IPSBAR + 0x1B_000C	GPTB Interrupt Enable Register	GPTBIE	8
IPSBAR + 0x1B_000E	GPTB System Control Register 2	GPTBSCR2	8
IPSBAR + 0x1B_000E	GPTB Flag Register 1	GPTBFLG1	8
IPSBAR + 0x1B_000F	GPTB Flag Register 2	GPTBFLG2	8
IPSBAR + 0x1B_0010	GPTB Channel 0 Register	GPTBC0	16
IPSBAR + 0x1B_0012	GPTB Channel 1 Register	GPTBC1	16
IPSBAR + 0x1B_0014	GPTB Channel 2 Register	GPTBC2	16
IPSBAR + 0x1B_0016	GPTB Channel 3 Register	GPTBC3	16
IPSBAR + 0x1B_0018	Pulse Accumulator Control Register	GPTBPACTL	8
IPSBAR + 0x1B_0019	Pulse Accumulator Flag Register	GPTBPAFLG	8

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
IPSBAR + 0x1B_001A	Pulse Accumulator Counter Register	GPTBPACNT	16
IPSBAR + 0x1B_001D	GPTB Port Data Register	GPTBPORT	8
IPSBAR + 0x1B_001E	GPTB Port Data Direction Register	GPTBDDR	8
FlexCAN Registers			
IPSBAR + 0x1C_0000	Module Configuration Register	CANMCR	16
IPSBAR + 0x1C_0006	Control Register 0	CANCTRL0	8
IPSBAR + 0x1C_0007	Control Register 1	CANCTRL1	8
IPSBAR + 0x1C_0008	Prescaler Divider	PRESDIV	8
IPSBAR + 0x1C_0009	Control Register 2	CANCTRL2	8
IPSBAR + 0x1C_000A	Free Running TImer	TIMER	16
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_0020	Error and Status	ESR	16
IPSBAR + 0x1C_0022	Interrupt Masks	IMASK	16
IPSBAR + 0x1C_0024	Interrupt Flags	IFLAG	16
IPSBAR + 0x1C_0026	Rx Error Counters	RXERRCNT	8
IPSBAR + 0x1C_0027	Tx Error Counter	TXERRCNT	8
IPSBAR + 0x1C_0080	Message Buffer 0 - Message Buffer 15	MBUFF0–MBUFF15	16x16bytes
Flash Registers			
IPSBAR + 0x1D_0000	CFM Configuration Register	CFMMCR	16

Table A-3. Register Memory Map (continued)

Address	Name	Mnemonic	Size
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
IPSBAR + 0x1D_0020	CFM User Status Register	CFMUSTAT	8
IPSBAR + 0x1D_0024	CFM Command Register	CFMCMD	8

1 The DMA module originally supported a left-justified 16-bit byte count register (BCR). This function was later reimplemented as a right-justified 24-bit BCR. The operation of the DMA and the interpretation of the BCR is controlled by the MPARK[BCR24BIT]. See Chapter 8, “System Control Module (SCM)” for more details.

2 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.

Appendix B
Revision History

This appendix lists major changes between versions of the MCF5282UM document.

B.1 Changes Between Rev. 0 and Rev. 0.1

Table B-1. Rev. 0 to Rev. 0.1 Changes

Location	Description
Title page	Changed title from “MCF5282 ColdFire® Integrated Microprocessor User’s Manual” to “MCF5282 ColdFire® Microcontroller User’s Manual.”
33.1/33-1	Added “This product incorporates SuperFlash® technology licensed from SST.”
Table 9-4 on page 9-6	Changed equation in footnote to $f_{sys} = f_{ref} \times 2(MFD + 2)/2$ exp RFD; $f_{ref} \times 2(MFD + 2) \leq 80$ MHz, $f_{sys} \leq 66$ MHz.
Table 9-4 on page 9-6	Multiplied all PLL frequencies in table by 2.
Table 10-13 on page 10-13	Changed DTMRx to DTIMx.
Figure 10-1 on page 10-6	Changed bit numbers from 63–32 to 31–0.
Figure 10-3 on page 10-8	Changed bit numbers from 63–32 to 31–0.
Figure 10-5 on page 10-10	Changed bit numbers from 63–32 to 31–0.
14.2.4/14-22	Added Section 14.2.4, “Chip Configuration Signals.”
Table 14-3 on page 14-11	Added Table 14-3.
15.2/15-3	Added “Unlike the MCF5272, the MCF5282 does not have an independent SDRAM clock signal. For the MCF5282, the timing of the SDRAM controller is controlled by the CLKOUT signal.”
15.2.3.2/15-13	Added Section 15.2.3.2, “SDRAM Byte Strobe Connections.”
15.2.3.1/15-9	Added “Note: Because the MCF5282 has 24 external address lines, the maximum SDRAM address size is 128 Mbits.”
Figure 27-4 on page 27-6	Changed reset value to 0010_0000_0000_0000.
Chapter 30, “Debug Support”	Changed “PSTCLK” references to “CLKOUT.”
Figure 30-41 on page 30-45	Changed “$\overline{\text{TEA}}$” to “$\overline{\text{TA}}$.”

Table B-1. Rev. 0 to Rev. 0.1 Changes (continued)

Location	Description
Figure 32-1 on page 32-2	Changed “RAS0” and “RAS1” to “SDRAM_CS0” and “SDRAM_CS1.”
Table 32-1 on page 32-4	Added Table 32-1.
Table 33-3 on page 33-3	Changed max input high voltage to 5.25 V.
Appendix A, “Register Memory Map	Changed “System Integration Module” to “System Control Module.”

B.2 Changes Between Rev. 0.1 and Rev. 1

Table B-2. Rev. 0.1 to Rev. 1 Changes

Location	Description
Figure 6-1 on page 6-3	Replaced Figure 6-1 with a more accurate block diagram.
6.2/6-2	Enhanced discussion of Flash blocks.
6.3.4.3/6-10	Added “Note: Enabling Flash security will disable BDM communications.”
6.3.4.3/6-10	Added “Note: When Flash security is enabled, the chip will boot in single chip mode regardless of the external reset configuration.”
6.4.3.1/6-17	Changed text in Step 1 to read “If $f_{SYS} \div 2$ is greater than 12.8 MHz, PRDIV8 = 1; otherwise PRDIV8 = 0.”
6.4.3.1/6-17	Changed equation in Step 2 to the following: $$DIV[5:0] = \frac{f_{SYS}}{2 \times 200kHz \times (1 + (PRDIV8 \times 7))}$$
6.4.3.1/6-17	Changed equation in Step 3 to the following: $$f_{CLK} = \frac{f_{SYS}}{2 \times (DIV[5:0] + 1) \times (1 + (PRDIV8 \times 7))}$$
6.4.3.1/6-17	Changed equations in example to reflect revisions above.
6.4.3.1/6-17	Changed text to read “So, for f_{SYS} = 66 MHz, writing 0x54 to CFMCLKD will set FCLK to 196.43 kHz which is a valid frequency for the timing of program and erase functions.”
6.4.3.1/6-17	Changed text to read “Consider the follwoing example for f_{SYS} = 66 MHz.”
Table 6-12 on page 6-16	Added “Page erase verify” category.
Table 6-13 on page 6-19	Added “Page erase verify” category and description.
Table 6-14 on page 6-23	Added “Access error” row.

Table B-2. Rev. 0.1 to Rev. 1 Changes (continued)

Location	Description
Chapter 8, "System Control Module (SCM) and 16.2/16-2	Moved information in Section 8.4.6, "DMA Request Control Register," to Section 16.2, "DMA Request Control (DMAREQC)."
Figure 8-2 on page 8-4	Changed offset for the copy of RAMBAR to "0x008."
Table 8-5 on page 8-6	Changed CWTIC to CWTIF.
8.5.2.1/8-9	Changed text to read "Setting MPARK[PRK_LAST] causes the arbitration pointer to be parked on the highest priority master."
Figure 9-2 on page 9-4	Changed "÷ MFD (2–9)" to "÷ MFD (4–18)."
Table 9-7 on page 9-10	Changed equation in "Normal PLL Clock Mode" row to the following: $f_{sys} = f_{ref} \times 2(MFD + 2)/2^{RFD}$
Chapter 12, "Chip Select Module	Eliminated Section 12.4.1.4, "Code Example."
Figure 12-4 on page 12-8	In "Reset: CSCR0" row, changed "D7, D6, D5" to "—, D19, D18."
Table 14-1 on page 14-3	Replaced "$\overline{SCKE}$" with "SCKE."
17.4.21/17-23	Changed text to read "The transmit FIFO uses addresses from the start of the FIFO to the location four bytes before the address programmed into the FRSR."
Table 17-29 on page 17-28	Added the following footnote: "The receive buffer pointer, which contains the address of the associated data buffer, must always be evenly divisible by 16. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller."
Table 17-30 on page 17-29	Added the following footnote: "The transmit buffer pointer, which contains the address of the associated data buffer, must always be evenly divisible by 4. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller."
Figure 18-1 on page 18-2	Changed value in "Divide by" block to 8192.
Table 19-3 on page 19-4	Multiplied all system clock divisor values in PRE field description by 2.
19.3.3/19-7	Changed equation in text to the following: Timeout period = PRE[3:0] × (PM[15:0] + 1) × system clock ÷ 2
Figure 23-12 on page 23-14	In "UISR Field" row, changed bit 6 to a reserved bit.
Table 23-10 on page 23-14	Changed bit 6 to a reserved bit.

Table B-2. Rev. 0.1 to Rev. 1 Changes (continued)

Location	Description
Table 25-12 on page 25-22	Changed equation in PRES_DIV field description to the following: $\text{S-clock} = \frac{f_{SYS}}{2(\text{PRESDIV} + 1)}$
27.6.2/27-8	Added “Note: When Flash security is enabled, the chip will boot in single chip mode regardless of the external reset configuration.”
Table 28-4 on page 28-10	Changed equation in QPR field description to the following: $f_{QCLK} = \frac{f_{SYS}}{2(\text{QPR}[6:0] + 1)}$
Table 28-5 on page 28-11	Multiplied all f_{SYS} divisor values in this table by 2.
30.1/30-1	Added “Note: Enabling Flash security will disable BDM communications.”
Figure 32-1 on page 32-2	Replaced “$\overline{\text{SCKE}}$” with “SCKE.”
Table 32-1 on page 32-4	Replaced “PEL2” with “PEL6, ” “PNQ6” with “PNQ7,” “PNQ5” with “PNQ6,” “PEL5” with “PEL1,” “PNQ4” with “PNQ5,” “PNQ3” with “PNQ4,” “PNQ2” with “PNQ3,” “PNQ1” with “PNQ2,” “PNQ0” with “PNQ1,” “PQS0” with “PQS1,” “PQS1” with “PQS0,” “PJ6” with “PJ7,” “$\overline{\text{RAS0}}$” with “$\overline{\text{SDRAM\_CS0}}$,” “$\overline{\text{RAS1}}$” with “$\overline{\text{SDRAM\_CS1}}$,” and “$\overline{\text{SCKE}}$” with “SCKE.”
Table 33-1 on page 33-1	Changed value for “ESD Target for Human Body Model” to “2000” and “ESD Target for Machine Model” to “200.”
Table 33-13 on page 33-11	Changed value in “Maximum number of guaranteed program/erase cycles before failure” row to “10,000.”
Table 33-15 on page 33-12	Changed the max value in specs B6a–B6c to “$0.5t_{CYC}$ + 10.”
Table 33-15 on page 33-12 Figure 33-3 on page 33-14 Figure 33-4 on page 33-15 Figure 33-5 on page 33-16	Changed the min value in spec B7a to “$0.5t_{CYC}$ + 2” and reflected the change in Figure 33-3, Figure 33-4, and Figure 33-5.
Table 33-16 on page 33-17	Changed the min value in spec D8 to “2” and the max value to”'—”.
Table 33-17 on page 33-18	Changed the max value in spec G1a to “12.”
Table 33-17 on page 33-18	Added the following footnote: “Because of long delays associated with the PQA/PQB pads, signals on the PQA/PQB pins will be updated on the following edge of the clock.”

Table B-2. Rev. 0.1 to Rev. 1 Changes (continued)

Location	Description
33.13/33-21	Added timing diagrams and tables to Section 33.13, “Fast Ethernet AC Timing Specifications.”
Table 33-27 on page 33-25	Changed the max value in spec 1 to “1/4.”
Table 33-27 on page 33-25	Changed the min value in spec 2 to “4.”
Table 33-27 on page 33-25	Changed the min value in spec 3 to “25.”
Table 33-27 on page 33-25	Changed the min value in spec 6 to “25.”
Table 33-27 on page 33-25	Changed the max value in spec 7 to “30.”
Table 33-27 on page 33-25	Changed the max value in spec 8 to “30.”
Table 33-27 on page 33-25	Changed the max value in spec 11 to “25.”
Table 33-28 on page 33-27	Changed the min value in spec D1 to “5.”
Table 33-28 on page 33-27	Changed the min value in spec D2 to “2.”
Table A-3 on page A-3	Changed offset for the copy of RAMBAR to “0x008.”

B.3 Changes Between Rev. 1 and Rev. 2

Table B-3. Rev. 1 to Rev. 2 Changes

Location	Description
Throughout Manual	Added MCF5281 device to manual. The MCF5281 implements half the Flash of the MCF5282.
1.1/1-1	Changed the description of real time debug support. It has only **one** user-visible hardware breakpoint register.
Table 2-2/2-7	Change the I field description to read: “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.”
Table 5-1/5-2	Replaced the description of PRI1 and PRI2.
Table 5-1/5-3	Added note to the SPV bit description, “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).’”
Figure 6-2/6-4	Replaced Figure 6-2, “CFM 512K Array Memory Map” and renamed it “CFM Array Memory Map”
Table 6-12/6-16	Change value for page erase verify command to 0x06.
Table 6-13/6-20	Change value for page erase verify command to 0x06.

Table B-3. Rev. 1 to Rev. 2 Changes (continued)

Location	Description
Table 8-3/8-5	Add the following note to the BDE bit description: “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 5.3.1, ‘SRAM Base Address Register (RAMBAR).’”
Figure 9-1/9-3	Remove ÷ 2 from CLKGEN block.
10.3.6/10-11	Add this text to the end of the first paragraph: “If a specific interrupt request is completely unused, the ICR*nx* value can remain in its reset (and disabled) state.”
10.5/10-17	Added the following 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.”
Figure 12-4/12-8	Changed CSCR*n* to reflect that AA is set at reset.
13.5/13-15	Removed final paragraph. The paragraph incorrectly states that the MCF5282 does not have a bus monitor.
Table 14-3/14-11	Changed pull-up indications in the ‘Internal Pull-Up’ column.
Table 17-13/17-26	Change encodings for bits 31–9 to: 0 The corresponding interrupt source is masked. 1The corresponding interrupt source is not masked.
Chapter 19	Change PIT1–PIT4 to PIT0–PIT3 throughout chapter. When a timer is referenced individually, PIT1 should be PIT0, PIT2 should be PIT1, PIT3 should be PIT2, and PIT4 should be PIT3. Other chapters in the user’s manual use the correct nomenclature: PIT0–PIT3.
19.6.3/19-7	Change timeout period equation to the equation below. $\text{Timeout period} = \frac{\text{PRE[3:0]} \times (\text{PM[15:0]} + 1) \times 2}{\text{system clock}}$
Figure 23-11	Change UISR bits 5–3 to reserved bits
24.6.1/24-11	Change ‘I2CR = 0xA’ to ‘I2CR = 0xA0.’
27.2.1/27-2	Changed ‘When interfacing to 16-bit ports, the port C and D pins and PJ[**5:4**] ($\overline{\text{BS}}$[**1:0**]) can be configured as general-purpose input/output (I/O)’
32.2/32-7	Added additional device number order information to Table 32-2 for MCF5280 and MCF5281 at 66- and 80-MHz, and MCF5282 at 80 MHz.
Chapter 33	Delete references to ‘$T_A = T_L$ to T_H’.
Table 33-1/33-1	Replace V_{in} row with the row below, in which the maximum value has been changed to 6.0 V.
Table 33-6/33-8	Replace I_{DDA} row with the row below, in which the maximum value in normal operation has been changed to 5.0 mA.
Figure 33-5/33-16	Replaced figure, “SDRAM Read Cycle”

B.4 Changes Between Rev. 2 and Rev. 2.1

Table B-4. Rev. 2 to Rev. 2.1 Changes

Location	Description
Title Page	Added MCF5280 to “Devices Supported” list on the title page.
Table 33-8/33-9	Deleted reference to “TA=TL to TH”

B.5 Changes Between Rev. 2.1 and Rev. 2.2

Table B-5. Rev. 2.1 to Rev. 2.2 Changes

Location	Description
Chapter 33	Added Power Spec info to Electricals chapter

B.6 Changes Between Rev. 2.2 and Rev. 2.3

Table B-6. Rev. 2.2 to Rev. 2.3 Changes

Location	Description
Figure 4-2/4-6	Changed bit 23 from DIDI to DISI
Table 4-6/4-9	Under ‘Configuration’ for ‘Instruction Cache’ the ‘Operation’ entry changed to “Invalidate 2 KByte data cache”
Table 4-6/4-9	Under ‘Configuration’ for ‘Data Cache’ the ‘Operation’ entry changed to “Invalidate 2 KByte instruction cache”
Figure 6-3/6-6	Changed bit 8 to write-only instead of read/write
Table 6-10/6-15	Removed “selected by BKSL[1:0]” as these are internal signal names not necessary for end-user.
10.3.2/10-8	Added note after register descriptions: ‘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.
Table 17-2/17-5	In PALR/PAUR entry, deleted “(only needed for full duplex flow control)”
Figure 17-23/17-39	Changed FRSR to read/write instead of read-only
25.4.10/25-16	Changed CANICR to ICR*n*
Table 25-17/25-29	Added the following information to BITERR and ACKERR descriptions: “To clear this bit, first read it as a one, then write it as a one. Writing zero has no effect.”
Table 25-17/25-30	Changed bit ordering: ERRINT should be bit 2 and BOFFINT should be bit 1.

Table B-6. Rev. 2.2 to Rev. 2.3 Changes (continued)

Location	Description
Table 25-19/25-32	Changed BUF*n*I field description from “To clear an interrupt flag, first read the flag as a one, then write it as a **zero**” to “To clear an interrupt flag, first read the flag as a one, then write it as a **one**.”
Chapter 33	Updated power consumption tables.

B.7 Changes Between Rev. 2.3 and Rev. 3

Table B-7. Rev. 2.3 to Rev. 3 Changes

Location	Description
Throughout	Added MCF5214 and MCF5216 to list of the devices supported in this document. These two devices are the same as the MCF5282 except they do not have an FEC and have a rated frequency of 66 MHz. Changed title of document.
Preface	Moved revision history to this appendix
Table 2-1/Page 2-4	Remove last sentence in C bit field description.
Table 2-3/Page 2-7	Change PC's Written with MOVEC entry to “No”.
Section 2.5/Page 2-8	Change last bullet to “Use of separate system stack pointers for user and supervisor modes”
Section 2.5/Page 2-9	Change last sentence in fourth paragraph (step 2) to “The IACK cycle is mapped to special locations within the interrupt controller's address space with the interrupt level encoded in the address."
Figure 3-6/Page 3-8	Add minus sign to the exponent so that it is “–(i + 1 – N)”.
Table 4-3/Page 4-5	Change reset value of ACR0, ACR1 to “See Section” since some of the bits are undefined after reset.
Figure 4-2/Page 4-6	Change CACR fields to R/W, since they may be read via the debug module.
Table 4-5/Page 4-8	For split instruction/data cache entry, swap text in parantheses in the description field. Instruction cache uses the upper half of the arrays, while data cache uses the lower half.
Figure 4-3/Page 4-9	Change reset value of ACR: Bits 31-16, 14-13, 6-5, and 2 are undefined, and other bits are cleared. Change ACR fields to R/W, since they may be read via the debug module.
Section 4.4.2.2/Page 4-9	Change note to: **NOTE** Peripheral (IPSBAR) space should not be cached. The combination of the CACR defaults and the two ACR*n* registers must define the non-cacheable attribute for this address space.
Figure 5-1/Page 5-2	Change RAMBAR fields to R/W, since they may be read via the debug module.
Table 5-1/Page 5-2	The PRI1/PRI2 text description does not match the table below it. It should be: “If a bit is set, CPU has priority. If a bit is cleared, DMA has priority.”
Figure 6-3/Page 6-6	Changed FLASHBAR[WP] to read-only.
Table 6-2/Page 6-7	Changed bit description of FLASHBAR[WP] to read-only and that this bit is always set.

Table B-7. Rev. 2.3 to Rev. 3 Changes (continued)

Location	Description
Chapter 8	Remove any references to the core watchdog timer being able to reset the device. It is only able to interrupt the processor. Use the peripheral watchdog timer described in Chapter 18 if needing a watchdog timer to reset the device.
Table 8-5/Page 8-6	CWCR[CWRI] bit description, change "...is programmed in the interrupt control register 7 (ICR7)..." to "...is programmed in the interrupt control register 8 (ICR8)..."
Table 9-4/Page 9-7	In the table for MFD bit definition, footnote (1) equation should read: $f_{sys} = \frac{f_{ref} \times 2(MFD + 2)}{2^{RFD}}$; $f_{ref} \times 2(MFD + 2) \leq f_{sys(max)}$; $f_{sys} \leq f_{sys(max)}$ Where $f_{sys(max)}$ is the maximum system frequency for the particular MCF5282 device (66MHz or 80MHz)
Section 10.3.6/Page 10-11	Include the following text in the section description and as a note in Figure 10-9. "It is the responsibility of the software to program the ICR*n*x registers with unique and non-overlapping level and priority definitions. Failure to program the ICR*n*x registers in this manner can result in undefined behavior. If a specific interrupt request is completely unused, the ICR*n*x value can remain in its reset (and disabled) state."
Figure 10-6/Page 10-9	Interrupt Force Register Low (INTFRCL*n*) is illustrated as read-only in the figure. However, this register should be read/write.
Table 10-14/Page 10-15	Change flag clearing mechanism for sources 24-26. They should read as follows: Write ERR_INT = 1 after reading ERR_INT = 1 Write BOFF_INT = 1 after reading BOFF_INT = 1 Write WAKE_INT = 1 after reading WAKE_INT = 1
Table 12-7/Page 12-7	BAM bit field description, the first example should read "So, if CSAR0 = 0x0000 and CSMR0[BAM] = 0x0001" instead of "So, if CSAR0 = 0x0000 and CSMR0[BAM] = 0x0008".
Table 10-2/Page 10-4	In footnote, remove mention of the SWIACK register, as it is not supported in the global IACK space.
Section 10.3.7/Page 10-16	Change last paragraph to: "In addition to the IACK registers within each interrupt controller, there are global L*n*IACK registers. A read from one of the global L*n*IACK registers returns the vector for the highest priority unmasked interrupt within a level for all interrupt controllers. There is no global SWIACK register. However, reading the SWIACK register from each interrupt controller returns the vector number of the highest priority unmasked request within that controller."
Figure 15-1/Page 15-1	Change SDRAM address lines from A[31:0] to A[23:0].
Table 15-1/Page 15-3	NOP command entry. Replace "$\overline{\text{SRAS}}$ asserted" with "$\overline{\text{SDRAM\_CS}}$[1:0] asserted"
Table 15-5/Page 15-7	Add the following note to the DACR*n*[CBM] field description: **Note:** It is important to set CBM according to the location of the command bit.
Section 16.5/Page 16-11	Remove last sentence in this section starting with "BCR*n* decrements..." since SAA bit is not supported.

Table B-7. Rev. 2.3 to Rev. 3 Changes (continued)

Location	Description
Section 17.4.6/Page 17-7	Add the following subsection entitled “Duplicate Frame Transmission”: The FEC fetches transmit buffer descriptors (TxBDs) and the corresponding transmit data continuously until the transmit FIFO is full. It does not determine whether the TxBD to be fetched is already being processed internally (as a result of a wrap). As the FEC nears the end of the transmission of one frame, it begins to DMA the data for the next frame. In order to remain one BD ahead of the DMA, it also fetches the TxBD for the next frame. It is possible that the FEC will fetch from memory a BD that has already been processed but not yet written back (that is, it is read a second time with the R bit still set). In this case, the data is fetched and transmitted again. Using at least three TxBDs fixes this problem for large frames, but not for small frames. To ensure correct operation for either large or small frames, one of the following must be true: • The FEC software driver ensures that there is always at least one TxBD with the ready bit cleared. • Every frame uses more than one TxBD and every TxBD but the last is written back immediately after the data is fetched. • The FEC software driver ensures a minimum frame size, n. The minimum number of TxBDs is then (Tx FIFO Size ÷ (n + 4)) rounded up to the nearest integer (though the result cannot be less than three). The default Tx FIFO size is 192 bytes; this size is programmable.
Table 17-9/Page 17-17	Correct MIB block counters end addresses to IPSBAR + 0x12FF.
Table 17-11/Page 17-19	Add RMON_R_DROP with an IPSBAR Offset of 0x1280 and a description of ‘Count of frames not counted correctly’.
Figure 17-26/Page 17-41	Change EMRBR register address from “IPSBAR + 0x11B8” to “IPSBAR + 0x1188”.
Section 20.5.13/Page 20-12	Deleted reference to nonexistent CF bits in the figure and bit descriptions for the GPTFLG2 register.
Figure 23-18/Page 23-18	Remove the two 16-bit divider blocks from timer input, as the divider is not available using external clock sources.
Section 23.5.1.2.2/Page 23-19	Remove 16-bit divider from equation, as the divider is not available using external clock sources.
Section 25.5.8/Page 25-25	Change end of last sentence from “...and can be written by the host to ‘0’.” to “...and can be written by the host to ‘1’.”
Table 25-17/Page 25-29	Remove the following information from the BITERR and ACKERR descriptions as these fields are read only: “To clear this bit, first read it as a one, then write it as a one. Writing zero has no effect.” (This is a rescindment of a previous documentation errata.) Change last sentence in ERRINT description from: “To clear this bit, first read it as a one, then write as a zero. Writing a one has no effect.” to “To clear this bit, first read it as a one, then write a one. Writing a zero has no effect.” Add the following information to the BOFFINT and WAKEINT descriptions: “To clear this bit, first read it as a one, then write it as a one. Writing zero has no effect.”
Table 25-17/Page 25-27	Definition of bits ERRINT and BOFFINT are incorrect for register ESTAT: ERRINT should be bit 1, BOFFINT should be bit 2. They should be cleared by writing a one instead of a zero.

Table B-7. Rev. 2.3 to Rev. 3 Changes (continued)

<table>
<tr><th>Location</th><th>Description</th></tr>
<tr><td>Table 26-1/Page 26-5</td><td>Change description field for DTOUT1 from “DMA timer 1 output / Port TD[3]...” to “DMA timer 1 output / Port TD[2]...”
Change description field for DTIN0 from “DMA timer 0 input / Port TD[3]...” to “DMA timer 1 output / Port TD[1]...”
Change description field for DTOUT0 from “DMA timer 0 output / Port TD[3]...” to “DMA timer 1 output / Port TD[0]...”</td></tr>
<tr><td>Chapter 27</td><td>The CLKMOD pins determine the clock mode regardless of $\overline{\text{RCON}}$ assertion during reset. Because of this, the following were done for clarity:
• Removed the RCON[RPLLREF, RPLLSEL] bit fields and descriptions.
• Added the following note in the Reset Configuration section: “The CLKMOD pins always determine the clock mode, regardless of the $\overline{\text{RCON}}$ pin value.”
• In the Configuration During Reset table, Clock Mode section, changed the Default Configuration from “RCON[7:6]” to “N/A”
• In Clock Mode Selection section, changed “The clock mode is selected during reset and reflected...” to “The clock mode is selected during reset by the CLKMOD pins and reflected...”
• In Clock Mode Selection table, added CLKMOD[1:0] column showing the corresponding CLKMOD settings for each mode.</td></tr>
<tr><td>Table 30-12/Page 30-14</td><td>Add the following note to the PBR[Address] field description:
Note: PBR[0] should always be loaded with a 0.</td></tr>
<tr><td>Table 30-20/Page 30-35</td><td>Change CSR’s initial state to 0x0000_0000.</td></tr>
<tr><td>Chapter 33</td><td>Add the following note:
“It is crucial during power-up that VDD never exceeds VDDH by more that ~0.3V. There are diode devices between the two voltage domains, and violating this rule can lead to a latch-up condition.”</td></tr>
<tr><td>Table 33-3/Page 33-3</td><td>In the V_{OH} and V_{OL} entries, change the respective I_{OH} and I_{OL} specs from “I_{OH} = -2.0mA” to “I_{OH} = -5.0mA” and “I_{OL} = +2.0mA” to “I_{OL} = +5.0mA”</td></tr>
<tr><td>Table 33-8/Page 33-7</td><td>In the PLL Electrical Specifications table, only specs for the 80MHz MCF5282 device were listed. Insert specs for the 66MHz device in the first 2 rows and also declare symbol $f_{sys(max)}$, as shown below:
<table>
<tr><th rowspan="2">Characteristic</th><th rowspan="2">Symbol</th><th rowspan="2">Min</th><th colspan="2">Max</th><th rowspan="2">Unit</th></tr>
<tr><th>66MHz</th><th>80MHz</th></tr>
<tr><td>PLL Reference Frequency Range
Crystal reference
External reference
1:1 Mode</td><td>
$f_{ref\_crystal}$
f_{ref_ext}
$f_{ref\_1:1}$</td><td>
2
2
33.33</td><td>
8.33
8.33
66.66</td><td>
10.0
10.0
80</td><td>MHz</td></tr>
<tr><td>System Frequency [1]
External Clock Mode
On-Chip PLL Frequency</td><td>f_{sys}</td><td>
0
f_{ref} / 32</td><td>$f_{sys(max)}$
66.66
66.66</td><td>$f_{sys(max)}$
80
80</td><td>MHz</td></tr>
</table></td></tr>
<tr><td>Table 33-8/Page 33-7</td><td>Change EXTAL Input High Voltage (V_{IHEXT}) Crystal Mode minimum spec from “V_{DD} - 1.0” to “V_{XTAL} + 0.4”.
Change EXTAL Input Low Voltage (V_{ILEXT}) Crystal Mode maximum spec from “1.0” to “V_{XTAL} - 0.4”.</td></tr>
<tr><td>Section 33.13.1/Page 33-21</td><td>Remove second sentence:
“There is no minimum frequency requirement.”</td></tr>
</table>

Table B-7. Rev. 2.3 to Rev. 3 Changes (continued)

Location	Description
Section 33.13.2/Page 33-22	Remove second sentence: "There is no minimum frequency requirement." Remove second paragraph as this feature is not supported on this device: "The transmit outputs (ETXD[3:0], ETXEN, ETXER) can be programmed to transition from either the rising or falling edge of ETXCLK, and the timing is the same in either case. This options allows the use of non-compliant MII PHYs. Refer to the Ethernet chapter for details of this option and how to enable it."
Table A-3/Page A-4	The CSMR1 and CSCR1 register addresses are incorrect. They should be IPSBAR + 0x090 and IPSBAR + 0x096 respectively

Appendix C
Index

D

E

F

G

Q

R

S

T

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for 32-bit Microcontrollers - MCU ***category:***

Click to view products by NXP ***manufacturer:***

Other Similar products are found below :

MB91F575BHSPMC-GSE1 MB91F594BSPMC-GSE1 PIC32MX120F032B-50I/ML MB91F464AAPMC-GSE2 MB91F577BHSPMC-GSE1 MB91F528USCPMC-GSE2 MB91F248PFV-GE1 MB91F594BPMC-GSE1 MB91243PFV-GS-136E1 MB91F577BHSPMC1-GSE1 PIC32MM0032GPL020-E/ML PIC32MM0016GPL028-E/SS PIC32MM0016GPL028-E/ML PIC32MM0032GPL028-E/ML PIC32MM0032GPL028-E/M6 SPC5604BF2VLL6 MB91F526KSEPMC-GSE1 TLE9872QTW40XUMA1 FT902L-T R5F564MLCDFB#31 R5F564MLCDFC#31 R5F523E5ADFL#30 R5F524TAADFF#31 MCF51AC256ACPUE PIC32MX150F128D-I/ML PIC32MX230F064D-I/PT PIC32MM0064GPL028-I/ML PIC32MM0064GPL028-I/SP PIC32MM0064GPL028-I/SO PIC32MX120F032D-I/TL PIC32MX130F064D-I/ML PIC32MZ2064DAB169-I/HF PIC32MZ2064DAB288-I/4J ATUC256L4U-AUT R5F56318CDBG#U0 PIC32MX150F128C-I/TL PIC32MX130F064C-ITL PIC32MX230F064D-IML PIC32MX154F128D-I/PT PIC32MX154F128B-V/SO AT32UC3L0128-AUT PIC32MX254F128B-I/SO PIC32MX230F128H-I/MR PIC32MX150F128D-50I/TL PIC32MZ1064DAB288-I/4J PIC32MZ1064DAB169-I/HF ATUC64D4-Z1UT AT32UC3A3128S-CTUT ATUC128L3U-Z3UT ATUC64L3U-Z3UT