MCF5307 ColdFire® Integrated Microprocessor User's Manual

MCF5307UM/D
Rev. 2.0, 08/2000

ColdFire is a registered trademark and DigitalDNA is a trademark of Motorola, Inc.
I^2C is a registered trademark of Philips Semiconductors

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1–303–675–2140 or 1–800–441–2447

JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center, 3–20–1, Minami–Azabu. Minato–ku, Tokyo 106–8573 Japan. 81–3–3440–3569

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852–26668334

Technical Information Center: 1–800–521–6274

HOME PAGE: http://www.motorola.com/semiconductors

Document Comments: FAX (512) 895-2638, Attn: RISC Applications Engineering

World Wide Web Addresses: http://www.motorola.com/PowerPC
http://www.motorola.com/NetComm
http://www.motorola.com/ColdFire

Freescale Semiconductor, Inc.

CONTENTS

CONTENTS

CONTENTS

CONTENTS

Chapter 5
Debug Support

CONTENTS

Part II
System Integration Module (SIM)

Chapter 6
SIM Overview

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

Chapter 15
Parallel Port (General-Purpose I/O)

CONTENTS

CONTENTS

CONTENTS

CONTENTS

Chapter 20
Electrical Specifications

ILLUSTRATIONS

ILLUSTRATIONS

ILLUSTRATIONS

ILLUSTRATIONS

ILLUSTRATIONS

ILLUSTRATIONS

TABLES

TABLES

TABLES

TABLES

TABLES

TABLES

About This Book

The primary objective of this user's manual is to define the functionality of the MCF5307 processors for use by software and hardware developers.

The information in this book is subject to change without notice, as described in the disclaimers on the title page of this book. As with any technical documentation, it is the readers' responsibility to be sure they are 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.motorola.com/coldfire.

Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products for the MCF5307. It is assumed that the reader understands operating systems, microprocessor system design, basic principles of software and hardware, and basic details of the ColdFire architecture.

Organization

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

- Chapter 1, "Overview," includes general descriptions of the modules and features incorporated in the MCF5307, focussing in particular on new features.
- Part I is intended for system designers who need to understand the operation of the MCF5307 ColdFire core.
  - — Chapter 2, "ColdFire Core," provides an overview of the microprocessor core of the MCF5307. The chapter begins with a description of enhancements from the V2 ColdFire core, and then fully describes the V3 programming model as it is implemented on the MCF5307. It also includes a full description of exception handling, data formats, an instruction set summary, and a table of instruction timings.
  - — Chapter 3, "Hardware Multiply/Accumulate (MAC) Unit," describes the MCF5307 multiply/accumulate unit, which executes integer multiply, multiply-accumulate, and miscellaneous register instructions. The MAC is integrated into the operand execution pipeline (OEP).

— Chapter 4, “Local Memory.” This chapter describes the MCF5307 implementation of the ColdFire V3 local memory specification. It consists of the two following major sections.

– Section 4.2, “SRAM Overview,” describes the MCF5307 on-chip static RAM (SRAM) implementation. It covers general operations, configuration, and initialization. It also provides information and examples showing how to minimize power consumption when using the SRAM.

– Section 4.7, “Cache Overview,” describes the MCF5307 cache implementation, including organization, configuration, and coherency. It describes cache operations and how the cache interacts with other memory structures.

— Chapter 5, “Debug Support,” describes the Revision C enhanced hardware debug support in the MCF5307. This revision of the ColdFire debug architecture encompasses earlier revisions.

- Part II, “System Integration Module (SIM),” describes the system integration module, which provides overall control of the bus and serves as the interface between the ColdFire core processor complex and internal peripheral devices. It includes a general description of the SIM and individual chapters that describe components of the SIM, such as the phase-lock loop (PLL) timing source, interrupt controller for peripherals, configuration and operation of chip selects, and the SDRAM controller.

— Chapter 6, “SIM Overview,” describes the SIM programming model, bus arbitration, and system-protection functions for the MCF5307.

— Chapter 7, “Phase-Locked Loop (PLL),” describes configuration and operation of the PLL module. It describes in detail the registers and signals that support the PLL implementation.

— Chapter 8, “I^2C Module,” describes the MCF5307 I^2C module, including I^2C protocol, clock synchronization, and the registers in the I^2C programing model. It also provides extensive programming examples.

— Chapter 9, “Interrupt Controller,” describes operation of the interrupt controller portion of the SIM. Includes descriptions of the registers in the interrupt controller memory map and the interrupt priority scheme.

— Chapter 10, “Chip-Select Module,” describes the MCF5307 chip-select implementation, including the operation and programming model, which includes the chip-select address, mask, and control registers.

— Chapter 11, “Synchronous/Asynchronous DRAM Controller Module,” describes configuration and operation of the synchronous/asynchronous DRAM controller component of the SIM. It begins with a general description and brief glossary, and includes a description of signals involved in DRAM operations. The remainder of the chapter is divided between descriptions of asynchronous and synchronous operations.

- Part III, “Peripheral Module,” describes the operation and configuration of the MCF5307 DMA, timer, UART, and parallel port modules, and describes how they interface with the system integration unit, described in Part II.
  - Chapter 12, “DMA Controller Module,” provides an overview of the DMA controller 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, showing timing diagrams for various operations.
  - Chapter 13, “Timer Module,” describes configuration and operation of the two general-purpose timer modules, timer 0 and timer 1. It includes programming examples.
  - Chapter 14, “UART Modules,” describes the use of the universal asynchronous/synchronous receiver/transmitters (UARTs) implemented on the MCF5307 and includes programming examples.
  - Chapter 15, “Parallel Port (General-Purpose I/O),” describes the operation and programming model of the parallel port pin assignment, direction-control, and data registers. It includes a code example for setting up the parallel port.
- Part IV, “Hardware Interface,” provides a pinout and both electrical and functional descriptions of the MCF5307 signals. It also describes how these signals interact to support the variety of bus operations shown in timing diagrams.
  - Chapter 16, “Mechanical Data,” provides a functional pin listing and package diagram for the MCF5307.
  - Chapter 17, “Signal Descriptions,” provides an alphabetical listing of MCF5307 signals. This chapter describes the MCF5307 signals. In particular, it shows which are inputs or outputs, how they are multiplexed, which signals require pull-up resistors, and the state of each signal at reset.
  - Chapter 18, “Bus Operation,” describes data transfers, error conditions, bus arbitration, and reset operations. It describes transfers initiated by the MCF5307 and by an external bus master, and includes detailed timing diagrams showing the interaction of signals in supported bus operations. Note that Chapter 11, “Synchronous/Asynchronous DRAM Controller Module,” describes DRAM cycles.
  - Chapter 19, “IEEE 1149.1 Test Access Port (JTAG),” describes configuration and operation of the MCF5307 JTAG test implementation. It describes the use of JTAG instructions and provides information on how to disable JTAG functionality.
  - Chapter 20, “Electrical Specifications,” describes AC and DC electrical specifications and thermal characteristics for the MCF5307. Because additional speeds may have become available since the publication of this book, consult Motorola’s ColdFire web page, http://www.motorola.com/coldfire, to confirm that this is the latest information.

This manual includes the following appendix:

- Appendix A, “List of Memory Maps,” lists the entire address-map for MCF5307 memory-mapped registers.

This manual also includes a glossary and an index.

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 Documentation

The ColdFire documentation is available from the sources listed on the back cover of this manual. Document order numbers are included in parentheses for ease in ordering.

- *ColdFire Programmers Reference Manual, R1.0* (MCF5200PRM/AD)
- 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*. These include the following:
  - *ColdFire MCF5102 User’s Manual* (MCF5102UM/AD)
  - *ColdFire MCF5202 User’s Manual* (MCF5202UM/AD)
  - *ColdFire MCF5204 User’s Manual* (MCF5204UM/AD)
  - *ColdFire MCF5206 User’s Manual* (MCF5206EUM/AD)
  - *ColdFire MCF5206E User’s Manual* (MCF5206EUM/AD)
- *ColdFire Programmers Reference Manual, R1.0* (MCF5200PRM/AD)
- *Using Microprocessors and Microcomputers: The Motorola Family*, William C. Wray, Ross Bannatyne, Joseph D. Greenfield

Additional literature on ColdFire implementations is being released as new processors become available. For a current list of ColdFire documentation, refer to the World Wide Web at http://www.motorola.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
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

Acronyms and Abbreviations

Table i lists acronyms and abbreviations used in this document.

Table i. Acronyms and Abbreviated Terms

Term	Meaning
ADC	Analog-to-digital conversion
ALU	Arithmetic logic unit
AVEC	Autovector
BDM	Background debug mode
BIST	Built-in self test
BSDL	Boundary-scan description language
CODEC	Code/decode
DAC	Digital-to-analog conversion
DMA	Direct memory access
DSP	Digital signal processing
EA	Effective address
EDO	Extended data output (DRAM)
FIFO	First-in, first-out

Table i. Acronyms and Abbreviated Terms (Continued)

Term	Meaning
GPIO	General-purpose I/O
I^2C	Inter-integrated circuit
IEEE	Institute for Electrical and Electronics Engineers
IFP	Instruction fetch pipeline
IPL	Interrupt priority level
JEDEC	Joint Electron Device Engineering Council
JTAG	Joint Test Action Group
LIFO	Last-in, first-out
LRU	Least recently used
LSB	Least-significant byte
lsb	Least-significant bit
MAC	Multiple accumulate unit
MBAR	Memory base address register
MSB	Most-significant byte
msb	Most-significant bit
Mux	Multiplex
NOP	No operation
OEP	Operand execution pipeline
PC	Program counter
PCLK	Processor clock
PLL	Phase-locked loop
PLRU	Pseudo least recently used
POR	Power-on reset
PQFP	Plastic quad flat pack
RISC	Reduced instruction set computing
Rx	Receive
SIM	System integration module
SOF	Start of frame
TAP	Test access port
TTL	Transistor-to-transistor logic
Tx	Transmit
UART	Universal asynchronous/synchronous receiver transmitter

Terminology and Notational Conventions

Table ii shows notational conventions used throughout this document.

Table ii Notational Conventions

Instruction	Operand Syntax
Opcode Wildcard	
cc	Logical condition (example: NE for not equal)
Register Specifications	
An	Any address register n (example: A3 is address register 3)
Ay,Ax	Source and destination address registers, respectively
Dn	Any data register n (example: D5 is data register 5)
Dy,Dx	Source and destination data registers, respectively
Rc	Any control register (example VBR is the vector base register)
Rm	MAC registers (ACC, MAC, MASK)
Rn	Any address or data register
Rw	Destination register w (used for MAC instructions only)
Ry,Rx	Any source and destination registers, respectively
Xi	index register i (can be an address or data register: Ai, Di)
Register Names	
ACC	MAC accumulator register
CCR	Condition code register (lower byte of SR)
MACSR	MAC status register
MASK	MAC mask register
PC	Program counter
SR	Status register
Port Name	
PSTDDATA	Processor status/debug data port
Miscellaneous Operands	
#<data>	Immediate data following the 16-bit operation word of the instruction
Í	Effective address
<ea>y,<ea>x	Source and destination effective addresses, respectively
<label>	Assembly language program label
<list>	List of registers for MOVEM instruction (example: D3–D0)
<shift>	Shift operation: shift left (<<), shift right (>>)
<size>	Operand data size: byte (B), word (W), longword (L)
bc	Both instruction and data caches
dc	Data cache

Table ii Notational Conventions (Continued)

Instruction	Operand Syntax
ic	Instruction cache
# <vector>	Identifies the 4-bit vector number for trap instructions
	identifies an indirect data address referencing memory
<xxx>	identifies an absolute address referencing memory
dn	Signal displacement value, n bits wide (example: d16 is a 16-bit displacement)
SF	Scale factor (x1, x2, x4 for indexed addressing mode, <<1n>> for MAC operations)
Operations	
+	Arithmetic addition or postincrement indicator
–	Arithmetic subtraction or predecrement indicator
x	Arithmetic multiplication
/	Arithmetic division
~	Invert; operand is logically complemented
&	Logical AND
|	Logical OR
^	Logical exclusive OR
<<	Shift left (example: D0 << 3 is shift D0 left 3 bits)
>>	Shift right (example: D0 >> 3 is shift D0 right 3 bits)
→	Source operand is moved to destination operand
←→	Two operands are exchanged
sign-extended	All bits of the upper portion are made equal to the high-order bit of the lower portion
If <condition> then <operations> else <operations>	Test the condition. If true, the operations after ‘then’ are performed. If the condition is false and the optional ‘else’ clause is present, the operations after ‘else’ are performed. If the condition is false and else is omitted, the instruction performs no operation. Refer to the Bcc instruction description as an example.
Subfields and Qualifiers	
{}	Optional operation
()	Identifies an indirect address
d_n	Displacement value, n-bits wide (example: d_{16} is a 16-bit displacement)
Address	Calculated effective address (pointer)
Bit	Bit selection (example: Bit 3 of D0)
lsb	Least significant bit (example: lsb of D0)
LSB	Least significant byte
LSW	Least significant word
msb	Most significant bit
MSB	Most significant byte
MSW	Most significant word

Table ii Notational Conventions (Continued)

Instruction	Operand Syntax
Condition Code Register Bit Names	
C	Carry
N	Negative
V	Overflow
X	Extend
Z	Zero

Chapter 1
Overview

This chapter is an overview of the MCF5307 ColdFire processor. It includes general descriptions of the modules and features incorporated in the MCF5307.

1.1 Features

The MCF5307 integrated microprocessor combines a V3 ColdFire processor core with the following components, as shown in Figure 1-1:

- 8-Kbyte unified cache
- 4-Kbyte on-chip SRAM
- Integer/fractional multiply-accumulate (MAC) unit
- Divide unit
- System debug interface
- DRAM controller for synchronous and asynchronous DRAM
- Four-channel DMA controller
- Two general-purpose timers
- Two UARTs
- I^2C™ interface
- Parallel I/O interface
- System integration module (SIM)

Designed for embedded control applications, the MCF5307 delivers 75 Dhrystone 2.1 MIPS at 90 MHz while minimizing system costs.

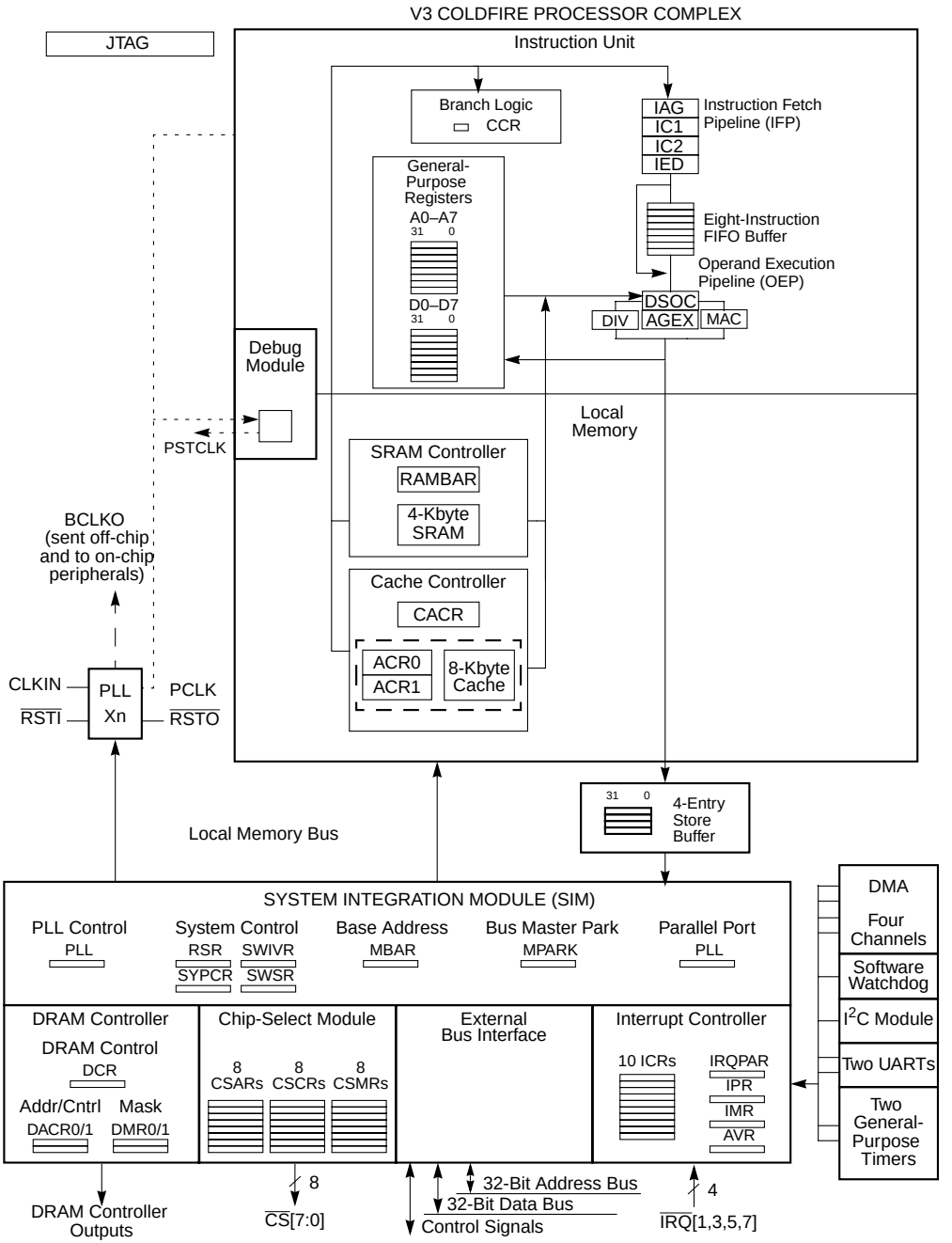


Figure 1-1. MCF5307 Block Diagram

Features common to many embedded applications, such as DMAs, various DRAM controller interfaces, and on-chip memories, are integrated using advanced process technologies.

The MCF5307 extends the legacy of Motorola's 68K family by providing a compatible path for 68K and ColdFire customers in which development tools and customer code can be leveraged. In fact, customers moving from 68K to ColdFire can use code translation and emulation tools that facilitate modifying 68K assembly code to the ColdFire architecture.

Based on the concept of variable-length RISC technology, the ColdFire family combines the architectural simplicity of conventional 32-bit RISC with a memory-saving, variable-length instruction set. In defining the ColdFire architecture for embedded processing applications, a 68K-code compatible core combines performance advantages of a RISC architecture with the optimum code density of a streamlined, variable-length M68000 instruction set.

By using a variable-length instruction set architecture, embedded system designers using ColdFire RISC processors enjoy significant advantages over conventional fixed-length RISC architectures. The denser binary code for ColdFire processors consumes less memory than many fixed-length instruction set RISC processors available. This improved code density means more efficient system memory use for a given application and allows use of slower, less costly memory to help achieve a target performance level.

The MCF5307 is the first standard product to implement the Version 3 ColdFire microprocessor core. To reach higher levels of frequency and performance, numerous enhancements were made to the V2 architecture. Most notable are a deeper instruction pipeline, branch acceleration, and a unified cache, which together provide 75 (Dhrystone 2.1) MIPS at 90 MHz. Increasing the internal speed of the core also allows higher performance while providing the system designer with an easy-to-use lower speed system interface. The processor complex frequency is an integer multiple, 2 to 4 times, of the external bus frequency. The core clock can be stopped to support a low-power mode.

Serial communication channels are provided by an I^2C interface module and two programmable full-duplex UARTs. Four channels of DMA allow for fast data transfer using a programmable burst mode independent of processor execution. The two 16-bit general-purpose multimode timers provide separate input and output signals. For system protection, the processor includes a programmable 16-bit software watchdog timer. In addition, common system functions such as chip selects, interrupt control, bus arbitration, and an IEEE 1149.1 JTAG module are included. A sophisticated debug interface supports background-debug mode plus real-time trace and debug with expanded flexibility of on-chip breakpoint registers. This interface is present in all ColdFire standard products and allows common emulator support across the entire family of microprocessors.

1.2 MCF5307 Features

The following list summarizes MCF5307 features:

- ColdFire processor core
  - Variable-length RISC, clock-multiplied Version 3 microprocessor core
  - Fully code compatible with Version 2 processors
  - Two independent decoupled pipelines: four-stage instruction fetch pipeline (IFP) and two-stage operand execution pipeline (OEP)
  - Eight-instruction FIFO buffer provides decoupling between the pipelines
  - Branch prediction mechanisms for accelerating program execution
  - 32-bit internal address bus supporting 4 Gbytes of linear address space
  - 32-bit data bus
  - 16 user-accessible, 32-bit-wide, general-purpose registers
  - Supervisor/user modes for system protection
  - Vector base register to relocate exception-vector table
  - Optimized for high-level language constructs
- Multiply and accumulate unit (MAC)
  - High-speed, complex arithmetic processing for DSP applications
  - Tightly coupled to the OEP
  - Three-stage execute pipeline with one clock issue rate for 16 x 16 operations
  - 16 x 16 and 32 x 32 multiplies support, all with 32-bit accumulate
  - Signed or unsigned integer support, plus signed fractional operands
- Hardware integer divide unit
  - Unsigned and signed integer divide support
  - Tightly coupled to the OEP
  - 32/16 and 32/32 operation support producing quotient and/or remainder results
- 8-Kbyte unified cache
  - Four-way set-associative organization
  - Operates at higher processor core frequency
  - Provides pipelined, single-cycle access to critical code and data
  - Supports write-through and copyback modes
  - Four-entry, 32-bit store buffer to improve performance of operand writes
- 4-Kbyte SRAM
  - Programmable location anywhere within 4-Gbyte linear address space
  - Higher core-frequency operation
  - Pipelined, single-cycle access to critical code or data

- DMA controller
  - — Four fully programmable channels: two support external requests
  - — Dual-address and single-address transfer support with 8-, 16-, and 32-bit data capability
  - — Source/destination address pointers that can increment or remain constant
  - — 24-bit transfer counter per channel
  - — Operand packing and unpacking supported
  - — Auto-alignment transfers supported for efficient block movement
  - — Bursting and cycle steal support
  - — Two-bus-clock internal access
  - — Automatic DMA transfers from on-chip UARTs using internal interrupts
- DRAM controller
  - — Synchronous DRAM (SDRAM), extended-data-out (EDO) DRAM, and fast page mode support
  - — Up to 512 Mbytes of DRAM
  - — Programmable timer provides CAS-before-RAS refresh for asynchronous DRAMs
  - — Support for two separate memory blocks
- Two UARTs
  - — Full-duplex operation
  - — Programmable clock
  - — Modem control signals available ($\overline{\text{CTS}}$, $\overline{\text{RTS}}$)
  - — Processor-interrupt capability
- Dual 16-bit general-purpose multiple-mode timers
  - — 8-bit prescaler
  - — Timer input and output pins
  - — Processor-interrupt capability
  - — Up to 22-nS resolution at 45 MHz
- 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
- System interface module (SIM)
  - — Chip selects provide direct interface to 8-, 16-, and 32-bit SRAM, ROM,

FLASH, and memory-mapped I/O devices
  - Eight fully programmable chip selects, each with a base address register
  - Programmable wait states and port sizes per chip select
  - User-programmable processor clock/input clock frequency ratio
  - Programmable interrupt controller
  - Low interrupt latency
  - Four external interrupt request inputs
  - Programmable autovector generator
  - Software watchdog timer
- 16-bit general-purpose I/O interface
- IEEE 1149.1 test (JTAG) module
- System debug support
  - Real-time trace for determining dynamic execution path while in emulator mode
  - Background debug mode (BDM) for debug features while halted
  - Real-time debug support, including 6 user-visible hardware breakpoint registers supporting a variety of breakpoint configurations
  - Supports comprehensive emulator functions through trace and breakpoint logic
- On-chip PLL
  - Supports processor clock/bus clock ratios of 66/33, 66/22, 66/16.5, 90/45, 90/30, and 90/22.5
  - Supports low-power mode
- Product offerings
  - 75 Dhrystone 2.1 MIPS at 90 MHz
  - Implemented in 0.35 µ, triple-layer-metal process technology with 3.3-V operation (5.0-V compliant I/O pads)
  - 208-pin plastic QFP package
  - 0°–70° C operating temperature

1.2.1 Process

The MCF5307 is manufactured in a 0.35-µ CMOS process with triple-layer-metal routing technology. This process combines the high performance and low power needed for embedded system applications. Inputs are 3.3-V tolerant; outputs are CMOS or open-drain CMOS with outputs operating from VDD + 0.5 V to GND - 0.5 V, with guaranteed TTL-level specifications.

1.3 ColdFire Module Description

The following sections provide overviews of the various modules incorporated in the MCF5307.

1.3.1 ColdFire Core

The Version 4 ColdFire core consists of two independent and decoupled pipelines to maximize performance—the instruction fetch pipeline (IFP) and the operand execution pipeline (OEP).

1.3.1.1 Instruction Fetch Pipeline (IFP)

The four-stage instruction fetch pipeline (IFP) is designed to prefetch instructions for the operand execution pipeline (OEP). Because the fetch and execution pipelines are decoupled by a eight-instruction FIFO buffer, the fetch mechanism can prefetch instructions in advance of their use by the OEP, thereby minimizing the time stalled waiting for instructions. To maximize the performance of branch instructions, the Version 3 IFP implements a branch prediction mechanism. Backward branches are predicted to be taken. The prediction for forward branches is controlled by a bit in the Condition Code Register (CCR). These predictions allow the IFP to redirect the fetch stream down the path predicted to be taken well in advance of the actual instruction execution. The result is significantly improved performance.

1.3.1.2 Operand Execution Pipeline (OEP)

The prefetched instruction stream is gated from the FIFO buffer into the two-stage OEP. The OEP consists of a traditional two-stage RISC compute engine with a register file access feeding an arithmetic/logic unit (ALU). The OEP decodes the instruction, fetches the required operands and then executes the required function.

1.3.1.3 MAC Module

The MAC unit provides signal processing capabilities for the MCF5307 in a variety of applications including digital audio and servo control. Integrated as an execution unit in the processor's OEP, the MAC unit implements a three-stage arithmetic pipeline optimized for 16 x 16 multiplies. Both 16- and 32-bit input operands are supported by this design in addition to a full set of extensions for signed and unsigned integers, plus signed, fixed-point fractional input operands.

1.3.1.4 Integer Divide Module

Integrated into the OEP, the divide module performs operations using signed and unsigned integers. The module supports word and longword divides producing quotients and/or remainders.

1.3.1.5 8-Kbyte Unified Cache

The MCF5307 architecture includes an 8-Kbyte unified cache. This four-way, set-associative cache provides pipelined, single-cycle access on cached instructions and operands.

As with all ColdFire caches, the cache controller implements a non-lockup, streaming design. The use of processor-local memories decouples performance from external memory speeds and increases available bandwidth for external devices or the on-chip 4-channel DMA.

The cache implements line-fill buffers to optimize 16-byte line burst accesses. Additionally, the cache supports copyback, write-through, or cache-inhibited modes. A 4-entry, 32-bit buffer is used for cache line push operations and can be configured for deferred write buffering in write-through or cache-inhibited modes.

1.3.1.6 Internal 4-Kbyte SRAM

The 4-Kbyte on-chip SRAM module provides pipelined, single-cycle access to memory regions mapped to these devices. The memory can be mapped to any 0-modulo-32K location in the 4-Gbyte address space. The SRAM module is useful for storing time-critical functions, the system stack, or heavily-referenced data operands.

1.3.2 DRAM Controller

The MCF5307 DRAM controller provides a direct interface for up to two blocks of DRAM. The controller supports 8-, 16-, or 32-bit memory widths and can easily interface to PC-100 DIMMs. A unique addressing scheme allows for increases in system memory size without rerouting address lines and rewiring boards. The controller operates in normal mode or in page mode and supports SDRAMs and EDO DRAMs.

1.3.3 DMA Controller

The MCF5307 provides four fully programmable DMA channels for quick data transfer. Dual- and single-address modes support bursting and cycle steal. Data transfers are 32 bits long with packing and unpacking supported along with an auto-alignment option for efficient block transfers. Automatic block transfers from on-chip serial UARTs are also supported through the DMA channels.

1.3.4 UART Modules

The MCF5307 contains two UARTs, which function independently. Either UART can be clocked by the system bus clock, eliminating the need for an external crystal. Each UART module interfaces directly to the CPU, as shown in Figure 1-2.

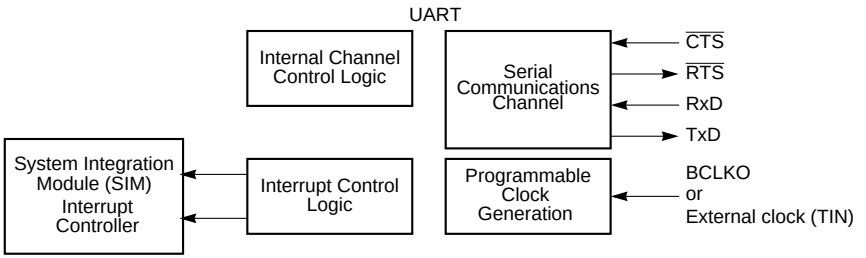


Figure 1-2. UART Module Block Diagram

Each UART module consists of the following major functional areas:

- Serial communication channel
- 16-bit divider for clock generation
- Internal channel control logic
- Interrupt control logic

Each UART contains an programmable clock-rate generator. Data formats can be 5, 6, 7, or 8 bits with even, odd, or no parity, and up to 2 stop bits in 1/16 increments. The UARTs include 4-byte and 2-byte FIFO buffers. The UART modules also provide several error-detection and maskable-interrupt capabilities. Modem support includes request-to-send ($\overline{RTS}$) and clear-to-send ($\overline{CTS}$) lines.

BCLKO provides the time base through a programmable prescaler. The UART time scale can also be sourced from a timer input. Full-duplex, auto-echo loopback, local loopback, and remote loopback modes allow testing of UART connections. The programmable UARTs can interrupt the CPU on various normal or error-condition events.

1.3.5 Timer Module

The timer module includes two general-purpose timers, each of which contains a free-running 16-bit timer for use in any of three modes. One mode captures the timer value with an external event. Another mode triggers an external signal or interrupts the CPU when the timer reaches a set value, while a third mode counts external events.

The timer unit has an 8-bit prescaler that allows programming of the clock input frequency, which is derived from the system bus cycle or an external clock input pin (TIN). The programmable timer-output pin generates either an active-low pulse or toggles the output.

1.3.6 I^2C Module

The I^2C interface is a two-wire, bidirectional serial bus used for quick data exchanges between devices. The I^2C minimizes the interconnection between devices in the end system and is best suited for applications that need occasional bursts of rapid communication over

short distances among several devices. The I^2C can operate in master, slave, or multiple-master modes.

1.3.7 System Interface

The MCF5307 processor provides a direct interface to 8-, 16-, and 32-bit FLASH, SRAM, ROM, and peripheral devices through the use of fully programmable chip selects and write enables. Support for burst ROMs is also included. Through the on-chip PLL, users can input a slower clock (16.6 to 45 MHz) that is internally multiplied to create the faster processor clock (33.3 to 90 MHz).

1.3.7.1 External Bus Interface

The bus interface controller transfers information between the ColdFire core or DMA and memory, peripherals, or other devices on the external bus. The external bus interface provides up to 32 bits of address bus space, a 32-bit data bus, and all associated control signals. This interface implements an extended synchronous protocol that supports bursting operations.

Simple two-wire request/acknowledge bus arbitration between the MCF5307 processor and another bus master, such as an external DMA device, is glueless with arbitration logic internal to the MCF5307 processor. Multiple-master arbitration is also available with some simple external arbitration logic.

1.3.7.2 Chip Selects

Eight fully programmable chip select outputs support the use of external memory and peripheral circuits with user-defined wait-state insertion. These signals interface to 8-, 16-, or 32-bit ports. The base address, access permissions, and internal bus transfer terminations are programmable with configuration registers for each chip select. $\overline{CS0}$ also provides global chip select functionality of boot ROM upon reset for initializing the MCF5307.

1.3.7.3 16-Bit Parallel Port Interface

A 16-bit general-purpose programmable parallel port serves as either an input or an output on a pin-by-pin basis.

1.3.7.4 Interrupt Controller

The interrupt controller provides user-programmable control of ten internal peripheral interrupts and implements four external fixed interrupt-request pins. Each internal interrupt can be programmed to any one of seven interrupt levels and four priority levels within each of these levels. Additionally, the external interrupt request pins can be mapped to levels 1, 3, 5, and 7 or levels 2, 4, 6, and 7. Autovector capability is available for both internal and external interrupts.

1.3.7.5 JTAG

To help with system diagnostics and manufacturing testing, the MCF5307 processor includes dedicated user-accessible test logic that complies with the IEEE 1149.1a standard for boundary-scan testability, often referred to as the Joint Test Action Group, or JTAG. For more information, refer to the IEEE 1149.1a standard.

1.3.8 System Debug Interface

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 unit in the MCF5307 is a compatible upgrade to the MCF52xx debug module with added flexibility in the breakpoint registers and a new command to view the program counter (PC).

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.

The MCF5307's new interrupt servicing options during emulator mode allow real-time critical interrupt service routines to be serviced while processing a debug interrupt event, thereby ensuring that the system continues to operate even during debugging.

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

1.3.9 PLL Module

The MCF5307 PLL module is shown in Figure 1-3.

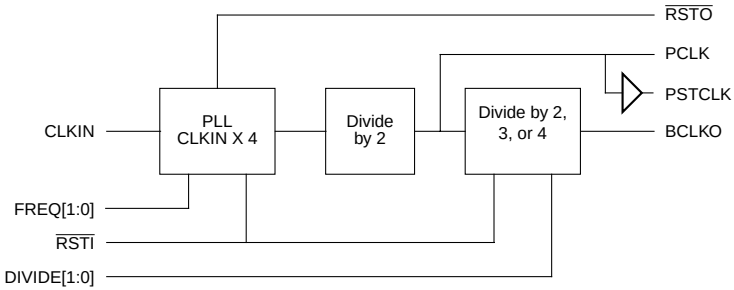


Figure 1-3. PLL Module

The PLL module's three modes of operation are described as follows.

- Reset mode—When $\overline{\text{RSTI}}$ is asserted, the PLL enters reset mode. At reset, the PLL asserts $\overline{\text{RSTO}}$ from the MCF5307. The core:bus frequency ratio and other MCF5307 configuration information are sampled during reset.
- Normal mode—In normal mode, the input frequency programmed at reset is clock-multiplied to provide the processor clock (PCLK).
- Reduced-power mode—In reduced-power mode, the PCLK is disabled by executing a sequence that includes programming a control bit in the system configuration register (SCR) and then executing the STOP instruction. Register contents are retained in reduced-power mode, so the system can be reenabled quickly when an unmasked interrupt or reset is detected.

1.4 Programming Model, Addressing Modes, and Instruction Set

The ColdFire programming model has two privilege levels—supervisor and user. The S bit in the status register (SR) indicates the privilege level. The processor identifies a logical address that differentiates between supervisor and user modes by accessing either the supervisor or user address space.

- User mode—When the processor is in user mode (SR[S] = 0), only a subset of registers can be accessed, and privileged instructions cannot be executed. Typically, most application processing occurs in user mode. User mode is usually entered by executing a return from exception instruction (RTE, assuming the value of SR[S] saved on the stack is 0) or a MOVE, SR instruction (assuming SR[S] is 0).
- Supervisor mode—This mode protects system resources from uncontrolled access by users. In supervisor mode, complete access is provided to all registers and the entire ColdFire instruction set. Typically, system programmers use the supervisor programming model to implement operating system functions and provide I/O

control. The supervisor programming model provides access to the same registers as the user model, plus additional registers for configuring on-chip system resources, as described in Section 1.4.3, “Supervisor Registers.”

Exceptions (including interrupts) are handled in supervisor mode.

1.4.1 Programming Model

Figure 1-4 shows the MCF5307 programming model.

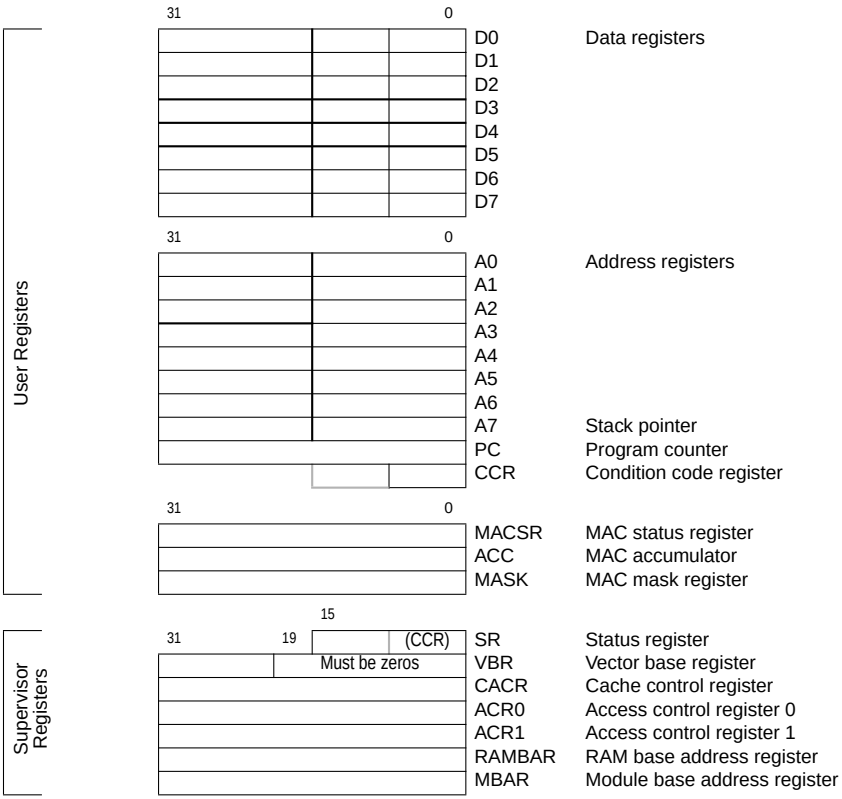


Figure 1-4. ColdFire MCF5307 Programming Model

1.4.2 User Registers

The user programming model is shown in Figure 1-4 and summarized in Table 1-1.

Table 1-1. User-Level Registers

Register	Description
Data registers (D0–D7)	These 32-bit registers are for bit, byte, word, and longword operands. They can also be used as index registers.
Address registers (A0–A7)	These 32-bit registers serve as software stack pointers, index registers, or base address registers. The base address registers can be used for word and longword operations. A7 functions as a hardware stack pointer during stacking for subroutine calls and exception handling.
Program counter (PC)	Contains the address of the instruction currently being executed by the MCF5307 processor
Condition code register (CCR)	The CCR is the lower byte of the SR. It contains indicator flags that reflect the result of a previous operation and are used for conditional instruction execution.
MAC status register (MACSR)	Defines the operating configuration of the MAC unit and contains indicator flags from the results of MAC instructions.
Accumulator (ACC)	General-purpose register used to accumulate the results of MAC operations
Mask register (MASK)	General-purpose register provides an optional address mask for MAC instructions that fetch operands from memory. It is useful in the implementation of circular queues in operand memory.

1.4.3 Supervisor Registers

Table 1-2 summarizes the MCF5307 supervisor-level registers.

Table 1-2. Supervisor-Level Registers

Register	Description
Status register (SR)	The upper byte of the SR provides interrupt information in addition to a variety of mode indicators signaling the operating state of the ColdFire processor. The lower byte of the SR is the CCR, as shown in Figure 1-4.
Vector base register (VBR)	Defines the upper 12 bits of the base address of the exception vector table used during exception processing. The low-order 20 bits are forced to zero, locating the vector table on 0-modulo-1 Mbyte address.
Cache configuration register (CACR)	Defines the operating modes of the Version 4 cache memories. Control fields configuring the instruction, data, and branch cache are provided by this register, along with the default attributes for the 4-Gbyte address space.
Access control registers (ACR0/1)	Define address ranges and attributes associated with various memory regions within the 4-Gbyte address space. Each ACR defines the location of a given memory region and assigns attributes such as write-protection and cache mode (copyback, write-through, cacheability). Additionally, CACR fields assign default attributes to the instruction and data memory spaces.
RAM base address register (RAMBAR)	Provide the logical base address for the 4-Kbyte SRAM module and define attributes and access types allowed for the SRAM.
Module base address register (MBAR)	Defines the logical base address for the memory-mapped space containing the control registers for the on-chip peripherals.

1.4.4 Instruction Set

The ColdFire instruction set supports high-level languages and is optimized for those instructions most commonly generated by compilers in embedded applications. Table 2-8 provides an alphabetized listing of the ColdFire instruction set opcodes, supported operation sizes, and assembler syntax. For two-operand instructions, the first operand is generally the source operand and the second is the destination.

Because the ColdFire architecture provides an upgrade path for 68K customers, its instruction set supports most of the common 68K opcodes. A majority of the instructions are binary compatible or optimized 68K opcodes. This feature, when coupled with the code conversion tools from third-party developers, generally minimizes software porting issues for customers with 68K applications.

Part I
MCF5307 Processor Core

Intended Audience

Part I is intended for system designers who need a general understanding of the functionality supported by the MCF5307. It also describes the operation of the MCF5307

Contents

- Chapter 2, “ColdFire Core,” provides an overview of the microprocessor core of the MCF5307. The chapter begins with a description of enhancements from the V2 ColdFire core, and then fully describes the V3 programming model as it is implemented on the MCF5307. It also includes a full description of exception handling, data formats, an instruction set summary, and a table of instruction timings.
- Chapter 3, “Hardware Multiply/Accumulate (MAC) Unit,” describes the MCF5307 multiply/accumulate unit, which executes integer multiply, multiply-accumulate, and miscellaneous register instructions. The MAC is integrated into the operand execution pipeline (OEP).
- Chapter 4, “Local Memory.” This chapter describes the MCF5307 implementation of the ColdFire V3 local memory specification. It consists of the two following major sections.
  - — Section 4.2, “SRAM Overview,” describes the MCF5307 on-chip static RAM (SRAM) implementation. It covers general operations, configuration, and initialization. It also provides information and examples showing how to minimize power consumption when using the SRAM.
  - — Section 4.7, “Cache Overview,” describes the MCF5307 cache implementation, including organization, configuration, and coherency. It describes cache operations and how the cache interacts with other memory structures.
- Chapter 5, “Debug Support,” describes the Revision C enhanced hardware debug support in the MCF5307. This revision of the ColdFire debug architecture encompasses earlier revisions.

Suggested Reading

The following literature may be helpful with respect to the topics in Part I:

- *ColdFire Programmers Reference Manual, R1.0* (MCF5200PRM/AD)
- *Using Microprocessors and Microcomputers: The Motorola Family*, William C. Wray, Ross Bannatyne, Joseph D. Greenfield

Acronyms and Abbreviations

Table I-i contains acronyms and abbreviations are used in Part I.

Table I-i. Acronyms and Abbreviated Terms

Term	Meaning
ADC	Analog-to-digital conversion
ALU	Arithmetic logic unit
BDM	Background debug mode
BIST	Built-in self test
BSDL	Boundary-scan description language
CODEC	Code/decode
DAC	Digital-to-analog conversion
DMA	Direct memory access
DSP	Digital signal processing
EA	Effective address
EDO	Extended data output (DRAM)
FIFO	First-in, first-out
GPIO	
I^2C	Inter-integrated circuit
IEEE	Institute for Electrical and Electronics Engineers
IFP	Instruction fetch pipeline
IPL	Interrupt priority level
JEDEC	Joint Electron Device Engineering Council
JTAG	Joint Test Action Group
LIFO	Last-in, first-out
LRU	Least recently used
LSB	Least-significant byte
lsb	Least-significant bit

Table I-i. Acronyms and Abbreviated Terms (Continued)

Term	Meaning
MAC	Multiple accumulate unit
MBAR	Memory base address register
MSB	Most-significant byte
msb	Most-significant bit
Mux	Multiplex
NOP	No operation
OEP	Operand execution pipeline
PC	Program counter
PCLK	Processor clock
PLL	Phase-locked loop
PLRU	Pseudo least recently used
POR	Power-on reset
PQFP	Plastic quad flat pack
RISC	Reduced instruction set computing
Rx	Receive
SIM	System integration module
SOF	Start of frame
TAP	Test access port
TTL	Transistor-to-transistor logic
Tx	Transmit
UART	Universal asynchronous/synchronous receiver transmitter

Chapter 2
ColdFire Core

This chapter provides an overview of the microprocessor core of the MCF5307. The chapter begins with a description of enhancements from the Version 2 (V2) ColdFire core, and then fully describes the V3 programming model as it is implemented on the MCF5307. It also includes a full description of exception handling, data formats, an instruction set summary, and a table of instruction timings.

2.1 Features and Enhancements

The MCF5307 is the first standard product to contain a Version 3 ColdFire microprocessor core. To reach higher levels of frequency and performance, numerous enhancements were made to the V2 architecture. Most notable are a deeper instruction pipeline, branch acceleration, and a unified cache, which together provide 75 (Dhrystone 2.1) MIPS at 90 MHz.

The MCF5307 core design emphasizes performance, and backward compatibility represents the next step on the ColdFire performance roadmap.

The following list summarizes MCF5307 features:

- Variable-length RISC, clock-multiplied Version 3 microprocessor core
- Two independent, decoupled pipelines—four-stage instruction fetch pipeline (IFP) and two-stage operand execution pipeline (OEP)
- Eight-instruction FIFO buffer provides decoupling between the pipelines
- Branch prediction mechanisms for accelerating program execution
- 32-bit internal address bus supporting 4 Gbytes of linear address space
- 32-bit data bus
- 16 user-accessible, 32-bit-wide, general-purpose registers
- Supervisor/user modes for system protection
- Vector base register to relocate exception-vector table
- Optimized for high-level language constructs

2.1.1 Clock-Multiplied Microprocessor Core

The MCF5307 incorporates a clock-multiplying phase-locked loop (PLL). Increasing the internal speed of the core also allows higher performance while providing the system designer with an easy-to-use lower speed system interface.

The frequency of the processor complex can be 2x, 3x, or 4x the external bus speed.

The processor, cache, integrated SRAM, and misalignment module operate at the higher speed clock (PCLK); other system integrated modules operate at the speed of the bus clock (BCLKO). When combined with the enhanced pipeline structure of the Version 3 ColdFire core, the processor and its local memories provide a high level of performance for today's demanding embedded applications.

PCLK can be disabled to minimize dissipation when a low-power mode is entered. This is described in Section 7.2.3, "Reduced-Power Mode."

2.1.2 Enhanced Pipelines

The IFP prefetches instructions. The OEP decodes instructions, fetches required operands, then executes the specified function. The two independent, decoupled pipeline structures maximize performance while minimizing core size. Pipeline stages are shown in Figure 2-1 and are summarized as follows:

- Four-stage IFP (plus optional instruction buffer stage)
  - — Instruction address generation (IAG) calculates the next prefetch address.
  - — Instruction fetch cycle 1 (IC1) initiates prefetch on the processor's local instruction bus.
  - — Instruction fetch cycle 2 (IC2) completes prefetch on the processor's instruction local bus.
  - — Instruction early decode (IED) generates time-critical decode signals needed for the OEP.
  - — Instruction buffer (IB) optional stage uses FIFO queue to minimize effects of fetch latency.
- Two-stage OEP
  - — Decode, select/operand fetch (DSOC) decodes the instruction and selects the required components for the effective address calculation, or the operand fetch cycle.
  - — Address generation/execute (AGEX) Calculates the oeprand address, or performs the execution of the instruction.

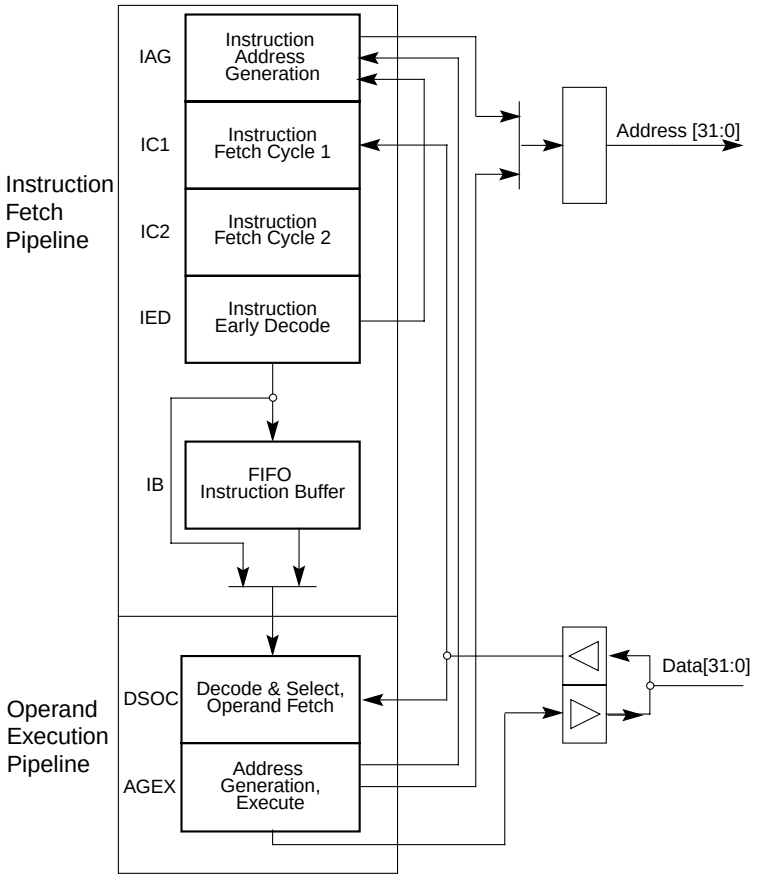


Figure 2-1. ColdFire Enhanced Pipeline

2.1.2.1 Instruction Fetch Pipeline (IFP)

Because the fetch and execution pipelines are decoupled by an eight-instruction FIFO buffer, the IFP can prefetch instructions before the OEP needs them, minimizing stalls.

2.1.2.1.1 Branch Acceleration

Because the IFP and the OEP are decoupled by the instruction buffer, the increased depth of the IFP is generally hidden from the OEP's instruction execution. The one exception is change-of-flow instructions such as unconditional branches or jumps, subroutine calls, and taken conditional branches. To minimize the effects of the increased depth of the IFP, the prefetched instruction stream is monitored for change-of-flow opcodes. When certain types of change-of-flow instructions are detected, the target instruction address is calculated, and fetching immediately begins in the target stream.

For example, if an unconditional BRA instruction is detected, the IED calculates the target of the BRA instruction, and the IAG immediately begins fetching at the target address. Because of the decoupled nature of the two pipelines, the target instruction is available to the OEP immediately after the BRA instruction, giving it a single-cycle execution time.

The acceleration logic uses a static prediction algorithm when processing conditional branch (Bcc) instructions. The default scheme is forward Bcc instructions are predicted as not-taken, while backward Bcc instructions are predicted as taken. A user-mode control bit, CCR[7], allows users to dynamically alter the prediction algorithm for forward Bcc instructions. See Section 2.2.1.5, “Condition Code Register (CCR).

2.1.2.2 Operand Execution Pipeline (OEP)

The OEP is a two-stage pipeline featuring a traditional RISC datapath with a register file feeding an arithmetic/logic unit. For simple register-to-register instructions, the first stage of the OEP performs the instruction decode and fetching of the required register operands (OC), while the actual instruction execution is performed in the second stage (EX).

For memory-to-register instructions, the instruction is effectively staged through the OEP twice in the following way:

- The instruction is decoded and the components of the operand address are selected (DS).
- The operand address is generated using the “execute engine” (AG).
- The memory operand is fetched while any register operand is simultaneously fetched (OC).
- The instruction is executed (EX).

For register-to-memory operations, the stage functions (DS/OC, AG/EX) are effectively performed simultaneously allowing single-cycle execution. For read-modify-write instructions, the pipeline effectively combines a memory-to-register operation with a store operation.

2.1.2.2.1 Illegal Opcode Handling

To aid in conversion from M68000 code, every 16-bit operation word is decoded to ensure that each instruction is valid. If the processor attempts execution of an illegal or unsupported instruction, an illegal instruction exception (vector 4) is taken.

2.1.2.2.2 Hardware Multiply/Accumulate (MAC) Unit

The MAC is an optional unit in Version 3 that provides hardware support for a limited set of digital signal processing (DSP) operations used in embedded code, while supporting the integer multiply instructions in the ColdFire microprocessor family. The MAC features a three-stage execution pipeline, optimized for 16 x 16 multiplies. It is tightly coupled to the OEP, which can issue a 16 x 16 multiply with a 32-bit accumulation plus fetch a 32-bit operand in a single cycle. A 32 x 32 multiply with a 32-bit accumulation requires three cycles before the next instruction can be issued.

Figure 2-2 shows basic functionality of the MAC. A full set of instructions are provided for signed and unsigned integers plus signed, fixed-point fractional input operands.

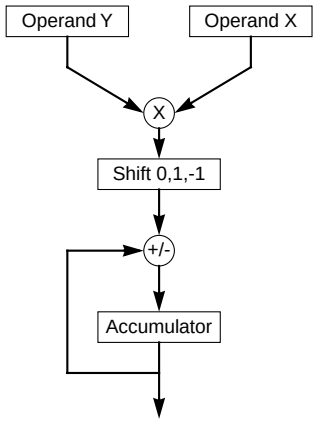


Figure 2-2. ColdFire Multiply-Accumulate Functionality Diagram

The MAC provides functionality in the following three related areas, which are described in detail in Chapter 3, “Hardware Multiply/Accumulate (MAC) Unit.”

- Signed and unsigned integer multiplies
- Multiply-accumulate operations with signed and unsigned fractional operands
- Miscellaneous register operations

2.1.2.2.3 Hardware Divide Unit

The hardware divide unit performs the following integer division operations:

- 32-bit operand/16-bit operand producing a 16-bit quotient and a 16-bit remainder
- 32-bit operand/32-bit operand producing a 32-bit quotient
- 32-bit operand/32-bit operand producing a 32-bit remainder

2.1.3 Debug Module Enhancements

The ColdFire processor core debug interface supports system integration in conjunction with low-cost development tools. Real-time trace and debug information can be accessed through a standard interface, which allows the processor and system to be debugged at full speed without costly in-circuit emulators. The MCF5307 debug unit is a compatible upgrade to the MCF52xx debug module with enhancements that include:

- A new command to obtain the value of the program counter (PC)
- Allowing ORing of terms in creating breakpoints
- Increased flexibility of the breakpoint registers

On-chip breakpoint resources include the following:

- Configuration/status register (CSR)
- Background debug mode (BDM) address attributes register (BAAR)
- Bus attributes and mask register (AATR)
- Breakpoint registers. These can be used to define triggers combining address, data, and PC conditions in single- or dual-level definitions. They include the following:
  - PC breakpoint register (PBR)
  - PC breakpoint mask register (PBMR)
  - Data operand address breakpoint registers (ABHR/ABLR)
  - Data breakpoint register (DBR)
- Data breakpoint mask register (DBMR)
- Trigger definition register (TDR) can be programmed to generate a processor halt or initiate a debug interrupt exception.

These registers can be accessed through the dedicated debug serial communication channel, or from the processor's supervisor programming model, using the WDEBUG instruction.

The enhancements of the Revision B debug specification are fully backward-compatible with the A revision. For more information, see Chapter 5, "Debug Support."

2.2 Programming Model

The MCF5307 programming model consists of three instruction and register groups—user, MAC (also user-mode), and supervisor, shown in Figure 2-2. User mode programs are restricted to user and MAC instructions and programming models. Supervisor-mode system software can reference all user-mode and MAC instructions and registers and additional supervisor instructions and control registers. The user or supervisor programming model is selected based on SR[S]. The following sections describe the registers in the user, MAC, and supervisor programming models.

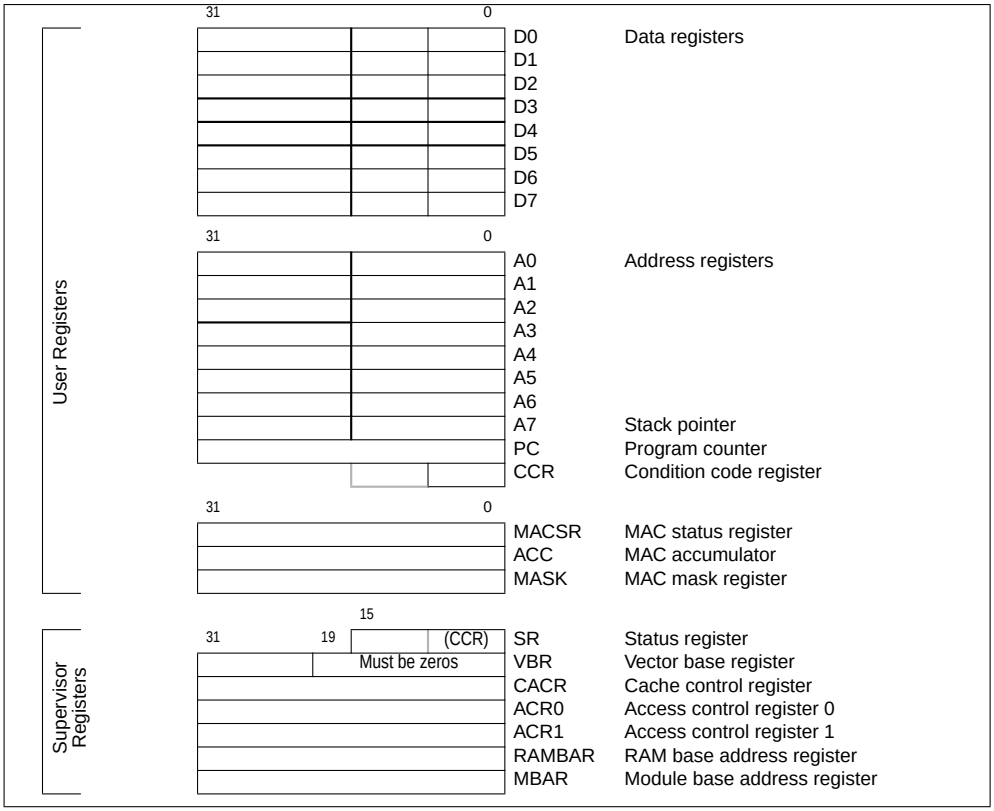


Figure 2-3. ColdFire Programming Model

2.2.1 User Programming Model

As Figure 2-3 shows, the user programming model consists of the following registers:

- 16 general-purpose 32-bit registers, D0–D7 and A0–A7
- 32-bit program counter
- 8-bit condition code register

2.2.1.1 Data Registers (D0–D7)

Registers D0–D7 are used as data registers for bit, byte (8-bit), word (16-bit), and longword (32-bit) operations. They may also be used as index registers.

2.2.1.2 Address Registers (A0–A6)

The address registers (A0–A6) can be used as software stack pointers, index registers, or base address registers and may be used for word and longword operations.

2.2.1.3 Stack Pointer (A7, SP)

The processor core supports a single hardware stack pointer (A7) used during stacking for subroutine calls, returns, and exception handling. The stack pointer is implicitly referenced by certain operations and can be explicitly referenced by any instruction specifying an address register. The initial value of A7 is loaded from the reset exception vector, address 0x0000. The same register is used for user and supervisor modes, and may be used for word and longword operations.

A subroutine call saves the program counter (PC) on the stack and the return restores the PC from the stack. The PC and the status register (SR) are saved on the stack during exception and interrupt processing. The return from exception instruction restores SR and PC values from the stack.

2.2.1.4 Program Counter (PC)

The PC holds the address of the executing instruction. For sequential instructions, the processor automatically increments PC. When program flow changes, the PC is updated with the target instruction. For some instructions, the PC specifies the base address for PC-relative operand addressing modes.

2.2.1.5 Condition Code Register (CCR)

The CCR, Figure 2-4, occupies SR[7–0], as shown in Figure 2-3. CCR[4–0] are indicator flags based on results generated by arithmetic operations.

	7	6	5	4	3	2	1	0
Field	P	—		X	N	Z	V	C
Reset	0	00		Undefined				
R/W	R/W	R		R/W	R/W	R/W	R/W	R/W

Table 2-1. CCR Field Descriptions

Bits	Name	Description
7	P	Branch prediction bit. Alters the static prediction algorithm used by the branch acceleration logic in the IFP on forward conditional branches. 0 Predicted as not-taken. 1 Predicted as taken.
6–5	—	Reserved, should be cleared.
4	X	Extend condition code bit. Assigned the value of the carry bit for arithmetic operations; otherwise not affected or set to a specified result. Also used as an input operand for multiple-precision arithmetic.
3	N	Negative condition code bit. Set if the msb of the result is set; otherwise cleared.
2	Z	Zero condition code bit. Set if the result equals zero; otherwise cleared.

Table 2-1. CCR Field Descriptions (Continued)

Bits	Name	Description
1	V	Overflow condition code bit. Set if an arithmetic overflow occurs, implying that the result cannot be represented in the operand size; otherwise cleared.
0	C	Carry condition code bit. Set if a carry-out of the data operand msb occurs for an addition or if a borrow occurs in a subtraction; otherwise cleared.

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

2.2.2 Supervisor Programming Model

The MCF5307 supervisor programming model is shown in Figure 2-3. Typically, system programmers use the supervisor programming model to implement operating system functions and provide memory and I/O control. The supervisor programming model provides access to the user registers and additional supervisor registers, which include the upper byte of the status register (SR), the vector base register (VBR), and registers for configuring attributes of the address space connected to the Version 3 processor core. Most supervisor-mode registers are accessed by using the MOVEC instruction with the control register definitions in Table 2-2.

Table 2-2. MOVEC Register Map

Rc[11–0]	Register Definition
0x002	Cache control register (CACR)
0x004	Access control register 0 (ACR0)
0x005	Access control register 1 (ACR1)
0x801	Vector base register (VBR)
0xC04	RAM base address register (RAMBAR)
0xC0F	Module base address register (MBAR)

2.2.2.1 Status Register (SR)

The SR stores the processor status, the interrupt priority mask, and other control bits. Supervisor software can read or write the entire SR; user software can read or write only SR[7–0], described in Section 2.2.1.5, “Condition Code Register (CCR).” The control bits indicate processor states—trace mode (T), supervisor or user mode (S), and master or interrupt state (M). SR is set to 0x27*xx* after reset.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	System byte								Condition code register (CCR)							
Field	T	—	S	M	—	I			P	—		X	N	Z	V	C
Reset	0	0	1	0	0	111			0	00		—	—	—	—	—
R/W	R/W	R	R/W	R/W	R	R/W			R/W	R		R/W	R/W	R/W	R/W	R/W

Figure 2-5. Status Register (SR)

Table 2-3 describes SR fields.

Table 2-3. Status Field Descriptions

Bits	Name	Description
15	T	Trace enable. When T is set, the processor performs a trace exception after every instruction.
13	S	Supervisor/user state. Indicates whether the processor is in supervisor or user mode 0 User mode 1 Supervisor mode
12	M	Master/interrupt state. Cleared by an interrupt exception. It can be set by software during execution of the RTE or move to SR instructions so the OS can emulate an interrupt stack pointer.
10–8	I	Interrupt priority mask. Defines the current interrupt priority. Interrupt requests are inhibited for all priority levels less than or equal to the current priority, except the edge-sensitive level-7 request, which cannot be masked.
7–0	CCR	Condition code register. See Table 2-1.

2.2.2.2 Vector Base Register (VBR)

The VBR holds the base address of the exception vector table in memory. The displacement of an exception vector is added to the value in this register to access the vector table. VBR[19–0] are not implemented and are assumed to be zero, forcing the vector table to be aligned on a 0-modulo-1-Mbyte boundary.

	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
Field	Exception vector table base address —
Reset	0000_0000_0000_0000_0000_0000_0000_0000
R/W	Written from a BDM serial command or from the CPU using the MOVEC instruction. VBR can be read from the debug module only. The upper 12 bits are returned, the low-order 20 bits are undefined.
Rc[11–0]	0x801

Figure 2-6. Vector Base Register (VBR)

2.2.2.3 Cache Control Register (CACR)

The CACR controls operation of both the instruction and data cache memory. It includes bits for enabling, freezing, and invalidating cache contents. It also includes bits for defining the default cache mode and write-protect fields. See Section 4.10.1, “Cache Control Register (CACR).”

2.2.2.4 Access Control Registers (ACR0–ACR1)

The access control registers (ACR0–ACR1) define attributes for two user-defined memory regions. Attributes include definition of cache mode, write protect and buffer write enables. See Section 4.10.2, “Access Control Registers (ACR0–ACR1).”

2.2.2.5 RAM Base Address Register (RAMBAR)

The RAMBAR register determines the base address location of the internal SRAM module and indicates the types of references mapped to it. The RAMBAR includes a base address, write-protect bit, address space mask bits, and an enable. The RAM base address must be aligned on a 0-modulo-32-Kbyte boundary. See Section 4.4.1, “SRAM Base Address Register (RAMBAR).”

2.2.2.6 Module Base Address Register (MBAR)

The module base address register (MBAR) defines the logical base address for the memory-mapped space containing the control registers for the on-chip peripherals. See Section 6.2.2, “Module Base Address Register (MBAR).”

2.3 Integer Data Formats

Table 2-4 lists the integer operand data formats. Integer operands can reside in registers, memory, or instructions. The operand size for each instruction is either explicitly encoded in the instruction or implicitly defined by the instruction operation.

Table 2-4. Integer Data Formats

Operand Data Format	Size
Bit	1 bit
Byte integer	8 bits
Word integer	16 bits
Longword integer	32 bits

2.4 Organization of Data in Registers

The following sections describe data organization within the data, address, and control registers.

2.4.1 Organization of Integer Data Formats in Registers

Figure 2-7 shows the integer format for data registers. Each integer data register is 32 bits wide. Byte and word operands occupy the lower 8- and 16-bit portions of integer data registers, respectively. Longword operands occupy the entire 32 bits of integer data registers. A data register that is either a source or destination operand only uses or changes the appropriate lower 8 or 16 bits in byte or word operations, respectively. The remaining

high-order portion does not change. The least significant bit (lsb) of all integer sizes is zero, the most-significant bit (msb) of a longword integer is 31, the msb of a word integer is 15, and the msb of a byte integer is 7.

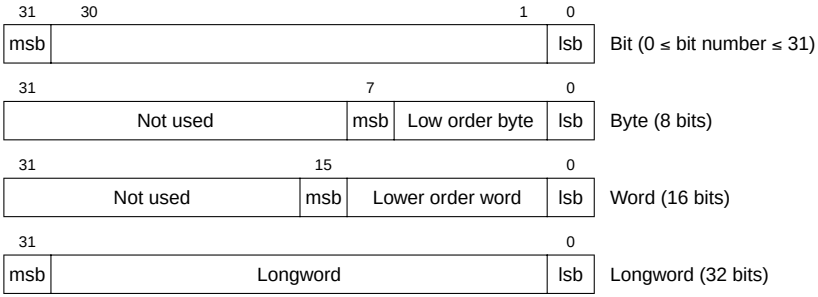


Figure 2-7. Organization of Integer Data Formats in Data Registers

The instruction set encodings do not allow the use of address registers for byte-sized operands. When an address register is a source operand, either the low-order word or the entire longword operand is used, depending on the operation size. Word-length source operands are sign-extended to 32 bits and then used in the operation with anaddress register destination. When an address register is a destination, the entire register is affected, regardless of the operation size. Figure 2-8 shows integer formats for address registers.

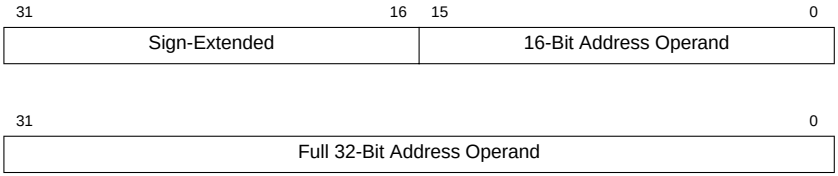


Figure 2-8. Organization of Integer Data Formats in Address Registers

The size of control registers varies according to function. Some have undefined bits reserved for future definition by Motorola. Those particular bits read as zeros and must be written as zeros for future compatibility.

All operations to the SR and CCR are word-size operations. For all CCR operations, the upper byte is read as all zeros and is ignored when written, regardless of privilege mode.

2.4.2 Organization of Integer Data Formats in Memory

All ColdFire processors use a big-endian addressing scheme. The byte-addressable organization of memory allows lower addresses to correspond to higher order bytes. The address N of a longword data item corresponds to the address of the high-order word. The lower order word is located at address N + 2. The address N of a word data item corresponds to the address of the high-order byte. The lower order byte is located at address N + 1. This

organization is shown in Figure 2-9.

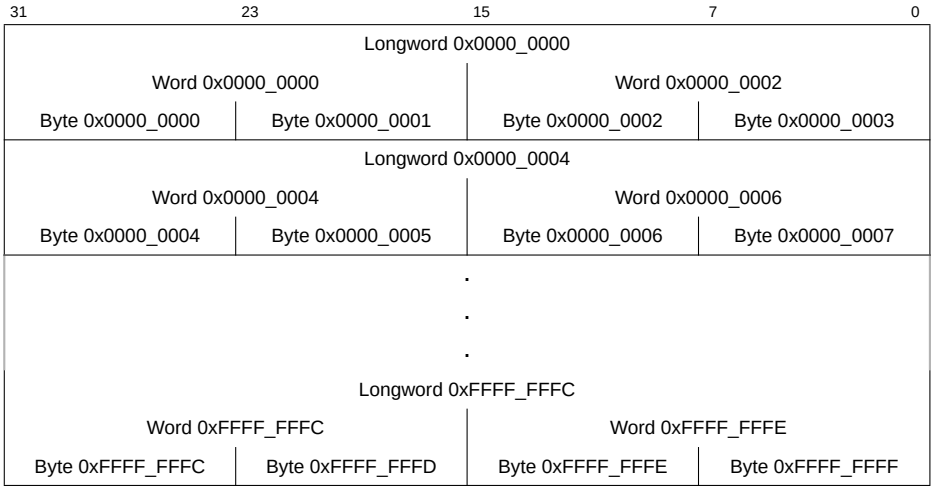


Figure 2-9. Memory Operand Addressing

2.5 Addressing Mode Summary

Addressing modes are categorized by how they are used. Data addressing modes refer to data operands. Memory addressing modes refer to memory operands. Alterable addressing modes refer to alterable (writable) data operands. Control addressing modes refer to memory operands without an associated size.

These categories sometimes combine to form more restrictive categories. Two combined classifications are alterable memory (both alterable and memory) and data alterable (both alterable and data). Twelve of the most commonly used effective addressing modes from the M68000 Family are available on ColdFire microprocessors. Table 2-5 summarizes these modes and their categories;

Table 2-5. ColdFire Effective Addressing Modes

Addressing Modes	Syntax	Mode Field	Reg. Field	Category			
				Data	Memory	Control	Alterable
Register direct Data Address	 Dn An	 000 001	 reg. no. reg. no.	 X —	 — —	 — —	 X X
Register indirect Address Address with Postincrement Address with Predecrement Address with Displacement	 (An) (An)+ –(An) (d_{16}, An)	 010 011 100 101	 reg. no. reg. no. reg. no. reg. no.	 X X X X	 X X X X	 X — — X	 X X X X
Address register indirect with index 8-bit displacement	(d_8, An, Xi)	110	reg. no.	X	X	X	X
Program counter indirect with displacement	(d_{16}, PC)	111	010	X	X	X	—
Program counter indirect with index 8-bit displacement	(d_8, PC, Xi)	111	011	X	X	X	—
Absolute data addressing Short Long	 (xxx).W (xxx).L	 111 111	 000 001	 X X	 X X	 X X	 — —
Immediate	#<xxx>	111	100	X	X	—	—

2.6 Instruction Set Summary

The ColdFire instruction set is a simplified version of the M68000 instruction set. The removed instructions include BCD, bit field, logical rotate, decrement and branch, and integer multiply with a 64-bit result. Nine new MAC instructions have been added.

Table 2-6 lists notational conventions used throughout this manual.

Table 2-6. Notational Conventions

Instruction	Operand Syntax
Opcode Wildcard	
cc	Logical condition (example: NE for not equal)

Table 2-6. Notational Conventions (Continued)

Instruction	Operand Syntax
Register Specifications	
An	Any address register n (example: A3 is address register 3)
Ay,Ax	Source and destination address registers, respectively
Dn	Any data register n (example: D5 is data register 5)
Dy,Dx	Source and destination data registers, respectively
Rc	Any control register (example VBR is the vector base register)
Rm	MAC registers (ACC, MAC, MASK)
Rn	Any address or data register
Rw	Destination register w (used for MAC instructions only)
Ry,Rx	Any source and destination registers, respectively
Xi	index register i (can be an address or data register: Ai, Di)
Register Names	
ACC	MAC accumulator register
CCR	Condition code register (lower byte of SR)
MACSR	MAC status register
MASK	MAC mask register
PC	Program counter
SR	Status register
Port Name	
DDATA	Debug data port
PST	Processor status port
Miscellaneous Operands	
#<data>	Immediate data following the 16-bit operation word of the instruction
Í	Effective address
<ea>y,<ea>x	Source and destination effective addresses, respectively
<label>	Assembly language program label
<list>	List of registers for MOVEM instruction (example: D3–D0)
<shift>	Shift operation: shift left (<<), shift right (>>)
<size>	Operand data size: byte (B), word (W), longword (L)
uc	Unified cache
# <vector>	Identifies the 4-bit vector number for trap instructions
	identifies an indirect data address referencing memory
<xxx>	identifies an absolute address referencing memory
dn	Signal displacement value, n bits wide (example: d16 is a 16-bit displacement)
SF	Scale factor (x1, x2, x4 for indexed addressing mode, <<1n>> for MAC operations)

Table 2-6. Notational Conventions (Continued)

Instruction	Operand Syntax
Operations	
+	Arithmetic addition or postincrement indicator
–	Arithmetic subtraction or predecrement indicator
x	Arithmetic multiplication
/	Arithmetic division
~	Invert; operand is logically complemented
&	Logical AND
|	Logical OR
^	Logical exclusive OR
<<	Shift left (example: D0 << 3 is shift D0 left 3 bits)
>>	Shift right (example: D0 >> 3 is shift D0 right 3 bits)
→	Source operand is moved to destination operand
←→	Two operands are exchanged
sign-extended	All bits of the upper portion are made equal to the high-order bit of the lower portion
If <condition> then <operations> else <operations>	Test the condition. If the condition is true, the operations in the then clause are performed. If the condition is false and the optional else clause is present, the operations in the else claue are performed. If the condition is false and the else clause is omitted, the instruction performs no operation. Refer to the Bcc instruction description as an example.
Subfields and Qualifiers	
{}	Optional operation
()	Identifies an indirect address
d_n	Displacement value, n-bits wide (example: d_{16} is a 16-bit displacement)
Address	Calculated effective address (pointer)
Bit	Bit selection (example: Bit 3 of D0)
lsb	Least significant bit (example: lsb of D0)
LSB	Least significant byte
LSW	Least significant word
msb	Most significant bit
MSB	Most significant byte
MSW	Most significant word
Condition Code Register Bit Names	

Table 2-6. Notational Conventions (Continued)

Instruction	Operand Syntax
P	Branch prediction
C	Carry
N	Negative
V	Overflow
X	Extend
Z	Zero

2.6.1 Instruction Set Summary

Table 2-7 lists implemented user-mode instructions by opcode.

Table 2-7. User-Mode Instruction Set Summary

Instruction	Operand Syntax	Operand Size	Operation
ADD	Dy,<ea>x <ea>y,Dx	.L .L	Source + destination → destination
ADDA	<ea>y,Ax	.L	Source + destination → destination
ADDI	#<data>,Dx	.L	Immediate data + destination → destination
ADDQ	#<data>,<ea>x	.L	Immediate data + destination → destination
ADDX	Dy,Dx	.L	Source + destination + X → destination
AND	Dy,<ea>x <ea>y,Dx	.L .L	Source & destination → destination
ANDI	#<data>,Dx	.L	Immediate data & destination → destination
ASL	Dy,Dx #<data>,Dx	.L .L	X/C ← (Dx << Dy) ← 0 X/C ← (Dx << #<data>) ← 0
ASR	Dy,Dx #<data>,Dx	.L .L	MSB → (Dx >> Dy) → X/C MSB → (Dx >> #<data>) → X/C
Bcc	<label>	.B,.W	If condition true, then PC + 2 + d_n → PC
BCHG	Dy,<ea>x #<data>,<ea-1>x	.B,.L .B,.L	~(<bit number> of destination) → Z, Bit of destination
BCLR	Dy,<ea>x #<data>,<ea-1>x	.B,.L .B,.L	~(<bit number> of destination) → Z; 0 → bit of destination
BRA	<label>	.B,.W	PC + 2 + d_n → PC
BSET	Dy,<ea>x #<data>,<ea-1>x	.B,.L .B,.L	~(<bit number> of destination) → Z; 1→ bit of destination
BSR	<label>	.B,.W	SP – 4 → SP; next sequential PC→ (SP); PC + 2 + d_n → PC
BTST	Dy,<ea>x #<data>,<ea-1>x	.B,.L .B,.L	~(<bit number> of destination) → Z
CLR	<ea>y,Dx	.B,.W,.L	0 → destination
CMP	<ea>y,Ax	.L	Destination – source
CMPA	<ea>y,Dx	.L	Destination – source

Table 2-7. User-Mode Instruction Set Summary (Continued)

Instruction	Operand Syntax	Operand Size	Operation
CMPI	<ea>y,Dx	.L	Destination – immediate data
DIVS	<ea-1>y,Dx <ea>y,Dx	.W .L	Dx /<ea>y → Dx {16-bit remainder; 16-bit quotient} Dx /<ea>y → Dx {32-bit quotient} Signed operation
DIVU	<ea-1>y,Dx Dy,<ea>x	.W .L	Dx /<ea>y → Dx {16-bit remainder; 16-bit quotient} Dx /<ea>y → Dx {32-bit quotient} Unsigned operation
EOR	Dy,<ea>x	.L	Source ^ destination → destination
EORI	#<data>,Dx	.L	Immediate data ^ destination → destination
EXT	#<data>,Dx	.B →.W .W →.L	Sign-extended destination → destination
EXTB	Dx	.B →.L	Sign-extended destination → destination
HALT[1]	None	Unsized	Enter halted state
JMP	<ea-3>y	Unsized	Address of <ea> → PC
JSR	<ea-3>y	Unsized	SP – 4 → SP; next sequential PC → (SP); <ea> → PC
LEA	<ea-3>y,Ax	.L	<ea> → Ax
LINK	Ax,#<d16>	.W	SP – 4 → SP; Ax → (SP); SP → Ax; SP + d16 → SP
LSL	Dy,Dx #<data>,Dx	.L .L	X/C ← (Dx << Dy) ← 0 X/C ← (Dx << #<data>) ← 0
LSR	Dy,Dx #<data>,Dx	.L .L	0 → (Dx >> Dy) → X/C 0 → (Dx >> #<data>) → X/C
MAC	Ry,RxSF	.L + (.W × .W) → .L .L + (.L × .L) → .L	ACC + (Ry × Rx){<< 1 | >> 1} → ACC ACC + (Ry × Rx){<< 1 | >> 1} → ACC; (<ea>y{&MASK}) → Rw
MACL	Ry,RxSF,<ea-1>y,Rw	.L + (.W × .W) → .L, .L .L + (.L × .L) → .L, .L	ACC + (Ry × Rx){<< 1 | >> 1} → ACC ACC + (Ry × Rx){<< 1 | >> 1} → ACC; (<ea-1>y{&MASK}) → Rw
MOVE	<ea>y,<ea>x	.B,.W,.L	<ea>y → <ea>x
MOVE from MAC	MASK,Rx ACC,Rx MACSR,Rx	.L	Rm → Rx
	MACSR,CCR	.L	MACSR → CCR
MOVE to MAC	Ry,ACC Ry,MACSR Ry,MASK	.L	Ry → Rm
	#<data>,ACC #<data>,MACSR #<data>,MASK	.L	#<data> → Rm
MOVE from CCR	CCR,Dx	.W	CCR → Dx
MOVE to CCR	Dy,CCR #<data>,CCR	.B	Dy → CCR #<data> → CCR
MOVEA	<ea>y,Ax	.W,.L → .L	Source → destination

Table 2-7. User-Mode Instruction Set Summary (Continued)

Instruction	Operand Syntax	Operand Size	Operation
MOVEM	#<list>,<ea-2>x <ea-2>y,#<list>	.L .L	Listed registers → destination Source → listed registers
MOVEQ	#<data>,Dx	.B → .L	Sign-extended immediate data → destination
MSAC	Ry,RxSF	.L - (.W × .W) → .L .L - (.L × .L) → .L	ACC – (Ry × Rx){<< 1 | >> 1} → ACC
MSACL	Ry,RxSF,<ea-1>y,Rw	.L - (.W × .W) → .L, .L .L - (.L × .L) → .L, .L	ACC – (Ry × Rx){<< 1 | >> 1} → ACC; (<ea-1>y{&MASK}) → Rw
MULS	<ea>y,Dx	.W X .W → .L .L X .L → .L	Source × destination → destination Signed operation
MULU	<ea>y,Dx	.W X .W → .L .L X .L → .L	Source × destination → destination Unsigned operation
NEG	Dx	.L	0 – destination → destination
NEGX	Dx	.L	0 – destination – X → destination
NOP	none	Unsized	Synchronize pipelines; PC + 2 → PC
NOT	Dx	.L	~ Destination → destination
OR	<ea>y,Dx Dy,<ea>x	.L	Source | destination → destination
ORI	#<data>,Dx	.L	Immediate data | destination → destination
PEA	<ea-3>y	.L	SP – 4 → SP; Address of <ea> → (SP)
PULSE	none	Unsized	Set PST= 0x4
REMS	<ea-1>,Dx	.L	Dx/<ea>y → Dw {32-bit remainder} Signed operation
REMU	<ea-1>,Dx	.L	Dx/<ea>y → Dw {32-bit remainder} Unsigned operation
RTS	none	Unsized	(SP) → PC; SP + 4 → SP
Scc	Dx	.B	If condition true, then 1s — destination; Else 0s → destination
SUB	<ea>y,Dx Dy,<ea>x	.L .L	Destination – source → destination
SUBA	<ea>y,Ax	.L	Destination – source → destination
SUBI	#<data>,Dx	.L	Destination – immediate data → destination
SUBQ	#<data>,<ea>x	.L	Destination – immediate data → destination
SUBX	Dy,Dx	.L	Destination – source – X → destination
SWAP	Dx	.W	MSW of Dx ←→ LSW of Dx
TRAP	#<vector>	Unsized	SP – 4 → SP;PC → (SP); SP – 2 → SP;SR → (SP); SP – 2 → SP; format → (SP); Vector address → PC
TRAPF	None #<data>	Unsized .W .L	PC + 2 → PC PC + 4 → PC PC + 6 → PC

Table 2-7. User-Mode Instruction Set Summary (Continued)

Instruction	Operand Syntax	Operand Size	Operation
TST	<ea>y	.B,.W,.L	Set condition codes
UNLK	Ax	Unsized	Ax →SP; (SP) → Ax; SP + 4 → SP
WDDATA	<ea>y	.B,.W,.L	<ea>y →DDATA port

1 By default the HALT instruction is a supervisor-mode instruction; however, it can be configured to allow user-mode execution by setting CSR[UHE].

Table 2-8 describes supervisor-mode instructions.

Table 2-8. Supervisor-Mode Instruction Set Summary

Instruction	Operand Syntax	Operand Size	Operation
CPUSHL	(An)	Unsized	Invalidate instruction cache line Push and invalidate data cache line Push data cache line and invalidate (I,D)-cache lines
HALT[1]	none	Unsized	Enter halted state
MOVE from SR	SR, Dx	.W	SR → Dx
MOVE to SR	Dy,SR #<data>,SR	.W	Source → SR
MOVEC	Ry,Rc	.L	Ry → Rc **Rc** **Register Definition** 0x002 Cache control register (CACR) 0x004 Access control register 0 (ACR0) 0x005 Access control register 1 (ACR1) 0x006 Access control register 2 (ACR2) 0x007 Access control register 3 (ACR3) 0x801 Vector base register (VBR) 0xC04 RAM base address register 0 (RAMBAR0) 0xC05 RAM base address register 1 (RAMBAR1)
RTE	None	Unsized	(SP+2) → SR; SP+4 → SP; (SP) → PC; SP + formatfield — SP
STOP	#<data>	.W	Immediate data → SR; enter stopped state
WDEBUG	<ea-2>y	.L	<ea-2>y → debug module

1 The HALT instruction can be configured to allow user-mode execution by setting CSR[UHE].

2.7 Instruction Timing

The timing data presented in this section assumes the following:

- The OEP is loaded with the opword and all required extension words at the beginning of each instruction execution. This implies that the OEP spends no time waiting for the IFP to supply opwords and/or extension words.
- The OEP experiences no sequence-related pipeline stalls. For the MCF5307, the most common example of this type of stall involves consecutive store operations, excluding the MOVEM instruction. For all store operations (except MOVEM),

certain hardware resources within the processor are marked as “busy” for two clock cycles after the final DSOC cycle of the store instruction. If a subsequent store instruction is encountered within this two-cycle window, it is stalled until the resource again becomes available. Thus, the maximum pipeline stall involving consecutive store operations is two cycles.

- The OEP can complete all memory accesses without memory causing any stall conditions. Thus, timing details in this section assume an infinite zero-wait state memory attached to the core.
- All operand data accesses are assumed to be aligned on the same byte boundary as the operand size:
  - — 16-bit operands aligned on 0-modulo-2 addresses
  - — 32-bit operands aligned on 0-modulo-4 addresses

  Operands that do not meet these guidelines are misaligned. Table 2-9 shows how the core decomposes a misaligned operand reference into a series of aligned accesses.

Table 2-9. Misaligned Operand References

A[1:0]	Size	Bus Operations	Additional C(R/W) [1]
x1	Word	Byte, Byte	2(1/0) if read 1(0/1) if write
x1	Long	Byte, Word, Byte	3(2/0) if read 2(0/2) if write
10	Long	Word, Word	2(1/0) if read 1(0/1) if write

[1] Each timing entry is presented as C(r/w), described as follows:

C is the number of processor clock cycles, including all applicable operand fetches and writes, as well as 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).

2.7.1 MOVE Instruction Execution Times

The execution times for the MOVE.{B,W,L} instructions are shown in the next tables. Table 2-12 shows the timing for the other generic move operations.

NOTE:

For all tables in this chapter, the execution time of any instruction using the PC-relative effective addressing modes is equivalent to the time using 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 2-10 lists execution times for MOVE.{B,W} instructions.

Table 2-10. 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)	4(1/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)
(Ay)+	4(1/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)
-(Ay)	4(1/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)
(d16,Ay)	4(1/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	—	—
(d8,Ay,Xi*SF)	5(1/0)	5(1/1)	5(1/1)	5(1/1)	—	—	—
(xxx).w	4(1/0)	4(1/1)	4(1/1)	4(1/1)	—	—	—
(xxx).l	4(1/0)	4(1/1)	4(1/1)	4(1/1)	—	—	—
(d16,PC)	4(1/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	—	—
(d8,PC,Xi*SF)	5(1/0)	5(1/1)	5(1/1)	5(1/1)	—	—	—
#<xxx>	1(0/0)	2(0/1)	2(0/1)	2(0/1)	—	—	—

Table 2-11 lists timings for MOVE.L.

Table 2-11. 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)	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)	2(0/1)	2(0/1)	2(0/1)	—	—	—

Table 2-12 gives execution times for MOVE.L instructions accessing program-visible registers of the MAC unit, along with other MOVE.L timings. Execution times for moving contents of the ACC or MACSR into a destination location represent the best-case scenario when the store instruction is executed and there are no load or MAC or MSAC instruction

in the MAC execution pipeline.

Table 2-12. MAC Move Execution Times

Opcode	í	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xi*SF)	(xxx).wl	#<xxx>
move.l	<ea>,ACC	1(0/0)	—	—	—	—	—	—	1(0/0)
move.l	<ea>,MACSR	2(0/0)	—	—	—	—	—	—	2(0/0)
move.l	<ea>,MASK	1(0/0)	—	—	—	—	—	—	1(0/0)
move.l	ACC,Rx	3(0/0)	—	—	—	—	—	—	—
move.l	MACSR,CCR	3(0/0)	—	—	—	—	—	—	—
move.l	MACSR,Rx	3(0/0)	—	—	—	—	—	—	—
move.l	MASK,Rx	3(0/0)	—	—	—	—	—	—	—

2.7.2 Execution Timings—One-Operand Instructions

Table 2-13 shows standard timings for single-operand instructions.

Table 2-13. One-Operand Instruction Execution Times

Opcode	í	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xi*SF)	(xxx).wl	#xxx
clr.b	í	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
clr.w	í	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
clr.l	í	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)	—	—	—	—	—	—	—
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	í	1(0/0)	4(1/0)	4(1/0)	4(1/0)	4(1/0)	5(1/0)	4(1/0)	1(0/0)
tst.w	í	1(0/0)	4(1/0)	4(1/0)	4(1/0)	4(1/0)	5(1/0)	4(1/0)	1(0/0)
tst.l	í	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)

2.7.3 Execution Timings—Two-Operand Instructions

Table 2-14 shows standard timings for two-operand instructions.

Table 2-14. Two-Operand Instruction Execution Times

Opcode	Í	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xi*SF)	(xxx).wl	#<xxx>
add.l	<ea>,Rx	1(0/0)	4(1/0)	4(1/0)	4(1/0)	4(1/0)	5(1/0)	4(1/0)	1(0/0)
add.l	Dy,<ea>	—	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)	—
addi.l	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
addq.l	#imm,<ea>	1(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)	—
addx.l	Dy,Dx	1(0/0)	—	—	—	—	—	—	—
and.l	<ea>,Rx	1(0/0)	4(1/0)	4(1/0)	4(1/0)	4(1/0)	5(1/0)	4(1/0)	1(0/0)
and.l	Dy,<ea>	—	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)	—
andi.l	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
asl.l	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
asr.l	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
bchg	Dy,<ea>	2(0/0)	5(1/1)	5(1/1)	5(1/1)	5(1/1)	6(1/1)	5(1/1)	—
bchg	#imm,<ea>	2(0/0)	5(1/1)	5(1/1)	5(1/1)	5(1/1)	—	—	—
bclr	Dy,<ea>	2(0/0)	5(1/1)	5(1/1)	5(1/1)	5(1/1)	6(1/1)	5(1/1)	—
bclr	#imm,<ea>	2(0/0)	5(1/1)	5(1/1)	5(1/1)	5(1/1)	—	—	—
bset	Dy,<ea>	2(0/0)	5(1/1)	5(1/1)	5(1/1)	5(1/1)	6(1/1)	5(1/1)	—
bset	#imm,<ea>	2(0/0)	5(1/1)	5(1/1)	5(1/1)	5(1/1)	—	—	—
btst	Dy,<ea>	1(0/0)	4(1/0)	4(1/0)	4(1/0)	4(1/0)	5(1/0)	4(1/0)	—
btst	#imm,<ea>	1(0/0)	4(1/0)	4(1/0)	4(1/0)	4(1/0)	—	—	—
cmp.l	<ea>,Rx	1(0/0)	4(1/0)	4(1/0)	4(1/0)	4(1/0)	5(1/0)	4(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)	35(1/0)	35(1/0)	35(1/0)	35(1/0)	—	—	—
divu.l	<ea>,Dx	35(0/0)	35(1/0)	35(1/0)	35(1/0)	35(1/0)	—	—	—
eor.l	Dy,<ea>	1(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(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)
mac.w	Ry,Rx	1(0/0)	—	—	—	—	—	—	—
mac.l	Ry,Rx	3(0/0)	—	—	—	—	—	—	—
msac.w	Ry,Rx	1(0/0)	—	—	—	—	—	—	—
msac.l	Ry,Rx	3(0/0)	—	—	—	—	—	—	—
mac.w	Ry,Rx,ea,Rw	—	3(1/0)	3(1/0)	3(1/0)	3(1/0)	—	—	—

Table 2-14. Two-Operand Instruction Execution Times (Continued)

Opcode	Í	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xi*SF)	(xxx).wl	#<xxx>
mac.l	Ry,Rx,ea,Rw	—	5(1/0)	5(1/0)	5(1/0)	5(1/0)	—	—	—
moveq	#imm,Dx	—	—	—	—	—	—	—	1(0/0)
msac.w	Ry,Rx,ea,Rw	—	3(1/0)	3(1/0)	3(1/0)	3(1/0)	—	—	—
msac.l	Ry,Rx,ea,Rw	—	5(1/0)	5(1/0)	5(1/0)	5(1/0)	—	—	—
muls.w	<ea>,Dx	3(0/0)	6(1/0)	6(1/0)	6(1/0)	6(1/0)	7(1/0)	6(1/0)	3(0/0)
mulu.w	<ea>,Dx	3(0/0)	6(1/0)	6(1/0)	6(1/0)	6(1/0)	7(1/0)	6(1/0)	3(0/0)
muls.l	<ea>,Dx	5(0/0)	8(1/0)	8(1/0)	8(1/0)	8(1/0)	—	—	—
mulu.l	<ea>,Dx	5(0/0)	8(1/0)	8(1/0)	8(1/0)	8(1/0)	—	—	—
or.l	<ea>,Rx	1(0/0)	4(1/0)	4(1/0)	4(1/0)	4(1/0)	5(1/0)	4(1/0)	1(0/0)
or.l	Dy,<ea>	—	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)	—
or.l	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
rems.l	<ea>,Dx	35(0/0)	35(1/0)	35(1/0)	35(1/0)	35(1/0)	—	—	—
remu.l	<ea>,Dx	35(0/0)	35(1/0)	35(1/0)	35(1/0)	35(1/0)	—	—	—
sub.l	<ea>,Rx	1(0/0)	4(1/0)	4(1/0)	4(1/0)	4(1/0)	5(1/0)	4(1/0)	1(0/0)
sub.l	Dy,<ea>	—	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)	—
subi.l	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
subq.l	#imm,<ea>	1(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)	—
subx.l	Dy,Dx	1(0/0)	—	—	—	—	—	—	—

2.7.4 Miscellaneous Instruction Execution Times

Table 2-15 lists timings for miscellaneous instructions.

Table 2-15. Miscellaneous Instruction Execution Times

Opcode	Í	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xi*SF)	(xxx).wl	#<xxx>
cpushl	(Ax)	—	11(0/1)	—	—	—	—	—	—
link.w	Ay,#imm	2(0/1)	—	—	—	—	—	—	—
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	9(0/0)	—	—	—	—	—	—	9(0/0)[1]
movec	Ry,Rc	11(0/1)	—	—	—	—	—	—	—
movem.l [2]	<ea>,&list	—	2+n(n/0)	—	—	2+n(n/0)	—	—	—
movem.l	&list,<ea>	—	2+n(0/n)	—	—	2+n(0/n)	—	—	—

Table 2-15. Miscellaneous Instruction Execution Times (Continued)

Opcode	Í	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xi*SF)	(xxx).wl	#<xxx>
nop		3(0/0)	—	—	—	—	—	—	—
pea	Í	—	2(0/1)	—	—	2(0/1) [3]	3(0/1)[4]	2(0/1)	—
pulse		1(0/0)	—	—	—	—	—	—	—
stop	#imm	—	—	—	—	—	—	—	3(0/0)[5]
trap	#imm	—	—	—	—	—	—	—	18(1/2)
trapf		1(0/0)	—	—	—	—	—	—	—
trapf.w		1(0/0)	—	—	—	—	—	—	—
trapf.l		1(0/0)	—	—	—	—	—	—	—
unlk	Ax	3(1/0)	—	—	—	—	—	—	—
wddata.l	Í	—	7(1/0)	7(1/0)	7(1/0)	7(1/0)	8(1/0)	7(1/0)	—
wdebug.l	Í	—	10(2/0)	—	—	10(2/0)	—	—	—

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

2 n is the number of registers moved by the MOVEM opcode.

3 PEA execution times are the same for (d16,PC).

4 PEA execution times are the same for (d8,PC,Xi*SF).

5 The execution time for STOP is the time required until the processor begins sampling continuously for interrupts.

2.7.5 Branch Instruction Execution Times

Table 2-16 shows general branch instruction timing.

Table 2-16. General Branch Instruction Execution Times

Opcode	Í	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xi*SF)	(xxx).wl	#<xxx>
bra		—	—	—	—	1(0/1)[1]	—	—	—
bsr		—	—	—	—	1(0/1)[1]	—	—	—
jmp	Í	—	5(0/0)	—	—	5(0/0) [1]	6(0/0)	1(0/0)[1]	—
jsr	Í	—	5(0/1)	—	—	5(0/1)	6(0/1)	1(0/1) [1]	—
rte		—	—	14(2/0)	—	—	—	—	—
rts		—	—	8(1/0)	—	—	—	—	—

1 Assumes branch acceleration. Depending on the pipeline status, execution times may vary from 1 to 3 cycles.

For the conditional branch opcodes (bcc), a static algorithm is used to determine the prediction state of the branch. This algorithm is:

```
if bcc is a forward branch && CCR[7] == 0
        then the bcc is predicted as not-taken
```

```
if bcc is a forward branch && CCR[7] == 1
        then the bcc is predicted as taken

        else if bcc is a backward branch
                then the bcc is predicted as taken
```

Table 2-17 shows timing for Bcc instructions.

Table 2-17. Bcc Instruction Execution Times

Opcode	Predicted Correctly as Taken	Predicted Correctly as Not Taken	Predicted Incorrectly
bcc	1(0/0)	1(0/0)	5(0/0)

2.8 Exception Processing Overview

Exception processing for ColdFire processors is streamlined for performance. Differences from previous M68000 Family processors include the following:

- A simplified exception vector table
- Reduced relocation capabilities using the vector base register
- A single exception stack frame format
- Use of a single, self-aligning system stack pointer

ColdFire processors use an instruction restart exception model but require more software support to recover from certain access errors. See Table 2-18 for details.

Exception processing can be defined as the time from the detection of the fault condition until the fetch of the first handler instruction has been initiated. It is comprised of the following four major steps:

1. The processor makes an internal copy of the SR and then enters supervisor mode by setting SR[S] and disabling trace mode by clearing SR[T]. The occurrence of an interrupt exception also forces SR[M] to be cleared and the interrupt priority mask to be set to the level of the current interrupt request.
2. The processor determines the exception vector number. For all faults except interrupts, the processor performs this calculation based on the exception type. For interrupts, the processor performs an interrupt-acknowledge (IACK) bus cycle to obtain the vector number from a peripheral device. The IACK cycle is mapped to a special acknowledge 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. ColdFire processors support a single stack pointer in the A7 address register; therefore, there is no notion of separate supervisor and user stack pointers. As a result, the exception stack frame is created at a 0-modulo-4 address on the top of the current system stack. Additionally, the processor uses a simplified

fixed-length stack frame for all exceptions. The exception type determines whether the program counter in the exception stack frame defines the address of the faulting instruction (fault) or of the next instruction to be executed (next).

4. The processor acquires the address of the first instruction of the exception handler. The exception vector table is aligned on a 1-Mbyte boundary. This instruction address is obtained by fetching a value from the table at the address defined in the vector base register. The index into the exception table is calculated as 4 x vector_number. When the index value is generated, the vector table contents determine the address of the first instruction of the desired handler. After the fetch of the first opcode of the handler is initiated, exception processing terminates and normal instruction processing continues in the handler.

ColdFire processors support a 1024-byte vector table aligned on any 1-Mbyte address boundary; see Table 2-18. The table contains 256 exception vectors where the first 64 are defined by Motorola; the remaining 192 are user-defined interrupt vectors.

Table 2-18. Exception Vector Assignments

Vector Numbers	Vector Offset (Hex)	Stacked Program Counter [1]	Assignment
0	000	—	Initial stack pointer
1	004	—	Initial program counter
2	008	Fault	Access error
3	00C	Fault	Address error
4	010	Fault	Illegal instruction
5	014	Fault	Divide by zero
6–7	018–01C	—	Reserved
8	020	Fault	Privilege violation
9	024	Next	Trace
10	028	Fault	Unimplemented line-a opcode
11	02C	Fault	Unimplemented line-f opcode
12	030	Next	Debug interrupt
13	034	—	Reserved
14	038	Fault	Format error
15	03C	Next	Uninitialized interrupt
16–23	040–05C	—	Reserved
24	060	Next	Spurious interrupt
25–31	064–07C	Next	Level 1–7 autovectored interrupts
32–47	080–0BC	Next	Trap #0–15 instructions
48–60	0C0–0F0	—	Reserved
61	0F4	Fault	Unsupported instruction

Table 2-18. Exception Vector Assignments (Continued)

Vector Numbers	Vector Offset (Hex)	Stacked Program Counter [1]	Assignment
62–63	0F8–0FC	—	Reserved
64–255	100–3FC	Next	User-defined interrupts

1 The term 'fault' refers to the PC of the instruction that caused the exception. The term 'next' refers to the PC of the instruction that immediately follows the instruction that caused the fault.

ColdFire processors inhibit sampling for interrupts during the first instruction of all exception handlers. This allows any handler to effectively disable interrupts, if necessary, by raising the interrupt mask level contained in the status register.

2.8.1 Exception Stack Frame Definition

The exception stack frame is shown in Figure 2-10. The first longword of the exception stack frame contains the 16-bit format/vector word (F/V) and the 16-bit status register. The second longword contains the 32-bit program counter address.

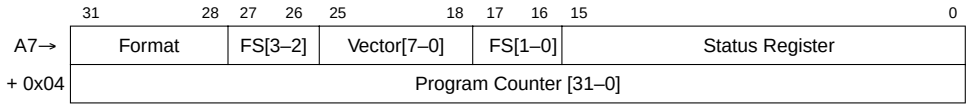


Figure 2-10. Exception Stack Frame Form

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

- Format field—This 4-bit field at the top of the system stack is always written with a value of {4,5,6,7} by the processor indicating a 2-longword frame format. See Table 2-19. This field records any longword misalignment of the stack pointer that may have existed when the exception occurred.

Table 2-19. Format Field Encoding

Original A7 at Time of Exception, Bits 1–0	A7 at First Instruction of Handler	Format Field Bits 31–28
00	Original A[7–8]	0100
01	Original A[7–9]	0101
10	Original A[7–10]	0110
11	Original A[7–11]	0111

- Fault status field—The 4-bit field, FS[3–0], at the top of the system stack is defined for access and address errors along with interrupted debug service routines. See Table 2-20.

Table 2-20. Fault Status Encodings

FS[3–0]	Definition
0000	Not an access or address error
0001-001x	Reserved
0100	Error on instruction fetch
0101–011x	Reserved
1000	Error on operand write
1001	Attempted write to write-protected space
101x	Reserved
1100	Error on operand read
1101–111x	Reserved

- Vector number—This 8-bit field, vector[7–0], defines the exception type. It is calculated by the processor for internal faults and is supplied by the peripheral for interrupts. See Table 2-18.

2.8.2 Processor Exceptions

Table 2-21 describes MCF5307 exceptions.

Table 2-21. MCF5307 Exceptions

Exception	Description
Access Error	Access errors are reported only in conjunction with an attempted store to write-protected memory. Thus, access errors associated with instruction fetch or operand read accesses are not possible.
Address Error	Caused by an attempted execution transferring control to an odd instruction address (that is, if bit 0 of the target address is set), an attempted use of a word-sized index register (Xi.w) or a scale factor of 8 on an indexed effective addressing mode, or attempted execution of an instruction with a full-format indexed addressing mode.
Illegal Instruction	On Version 2 ColdFire implementations, only some illegal opcodes were decoded and generated an illegal instruction exception. The Version 3 processor decodes the full 16-bit opcode and generates this exception if execution of an unsupported instruction is attempted. Additionally, attempting to execute an illegal line A or line F opcode generates unique exception types: vectors 10 and 11, respectively. ColdFire processors do not provide illegal instruction detection on extension words of any instruction, including MOVEC. Attempting to execute an instruction with an illegal extension word causes undefined results.
Divide by Zero	Attempted division by zero causes an exception (vector 5, offset = 0x014) except when the PC points to the faulting instruction (DIVU, DIVS, REMU, REMS).
Privilege Violation	Caused by attempted execution of a supervisor mode instruction while in user mode. The ColdFire Programmer's Reference Manual lists supervisor- and user-mode instructions.

Table 2-21. MCF5307 Exceptions (Continued)

Exception	Description
Trace Exception	ColdFire processors provide instruction-by-instruction tracing. While the processor is in trace mode (SR[T] = 1), instruction completion signals a trace exception. This allows a debugger to monitor program execution. The only exception to this definition is the STOP instruction. If the processor is in trace mode, the instruction before the STOP executes and then generates a trace exception. In the exception stack frame, the PC points to the STOP opcode. When the trace handler is exited, the STOP instruction is executed, loading the SR with the immediate operand from the instruction. The processor then generates a trace exception. The PC in the exception stack frame points to the instruction after STOP, and the SR reflects the just-loaded value. If the processor is not in trace mode and executes a STOP instruction where the immediate operand sets the trace bit in the SR, hardware loads the SR and generates a trace exception. The PC in the exception stack frame points to the instruction after STOP, and the SR reflects the just-loaded value. Because ColdFire processors do not support 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 executing in trace mode. The processor initiates the TRAP exception and passes control to the corresponding handler. If the system requires that a trace exception be processed, the TRAP exception handler must check for this condition (SR[15] in the exception stack frame asserted) and pass control to the trace handler before returning from the original exception.
Debug Interrupt	Caused by a hardware breakpoint register trigger. Rather than generating an IACK cycle, the processor internally calculates the vector number (12). Additionally, the M bit and the interrupt priority mask fields of the SR are unaffected by the interrupt. See Section 2.2.2.1, “Status Register (SR).”
RTE and Format Error Exceptions	When an RTE instruction executes, the processor first examines the 4-bit format field to validate the frame type. For a ColdFire processor, any attempted execution of an RTE 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 exception frame and the stacked PC points to RTE.The selection of the format value provides limited debug support for porting code from M68000 applications. On M68000 Family processors, the SR was at the top of the stack. Bit 30 of the longword addressed by the system stack pointer is typically zero; so, attempting an RTE using this old format generates a format error on a ColdFire processor. If the format field defines a valid type, the processor does the following: 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 first longword fetch. 4 Transfers control to the instruction address defined by the second longword operand in the stack frame.
TRAP	Executing TRAP always forces an exception and is useful for implementing system calls. The trap instruction may be used to change from user to supervisor mode.
Interrupt Exception	Interrupt exception processing, with interrupt recognition and vector fetching, includes uninitialized and spurious interrupts as well as those where the requesting device supplies the 8-bit interrupt vector. Autovectoring may optionally be configured through the system interface module (SIM). See Section 9.2.2, “Autovector Register (AVR).”

Table 2-21. MCF5307 Exceptions (Continued)

Exception	Description
Reset Exception	Asserting the reset input signal ($\overline{\text{RSTI}}$) causes a reset exception. Reset has the highest exception priority; it provides for system initialization and recovery from catastrophic failure. When assertion of $\overline{\text{RSTI}}$ is recognized, current processing is aborted and cannot be recovered. The reset exception places the processor in supervisor mode by setting SR[S] and disables tracing by clearing SR[T]. This exception also clears SR[M] and sets the processor's interrupt priority mask in the SR to the highest level (level 7). Next, the VBR is initialized to 0x0000_0000. Configuration registers controlling the operation of all processor-local memories (cache and RAM modules on the MCF5307) are invalidated, disabling the memories. Note: Other implementation-specific supervisor registers are also affected. Refer to each of the modules in this manual for details on these registers. After $\overline{\text{RSTI}}$ is negated, the processor waits 80 cycles before beginning the actual reset exception process. During this time, certain events are sampled, including the assertion of the debug breakpoint signal. If the processor is not halted, it initiates the reset exception by performing two longword read bus cycles. The longword at address 0 is loaded into the stack pointer and the longword at address 4 is loaded into the PC. 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 executes, the processor enters the fault-on-fault halted state.
Unsupported Instruction Exception	If the MCF5307 attempts to execute a valid instruction but the required optional hardware module is not present in the OEP, a non-supported instruction exception is generated (vector 0x61). Control is then passed to an exception handler that can then process the opcode as required by the system.

If a ColdFire processor encounters any type of fault during the exception processing of another fault, the processor immediately halts execution with the catastrophic fault-on-fault condition. A reset is required to force the processor to exit this halted state.

Chapter 3
Hardware Multiply/Accumulate (MAC) Unit

This chapter describes the MCF5307 multiply/accumulate (MAC) unit, which executes integer multiply, multiply-accumulate, and miscellaneous register instructions. The MAC is integrated into the operand execution pipeline (OEP).

3.1 Overview

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

The MAC unit provides signal processing capabilities for the MCF5307 in a variety of applications including digital audio and servo control. Integrated as an execution unit in the processor's OEP, the MAC unit implements a three-stage arithmetic pipeline optimized for 16 x 16 multiplies. Both 16- and 32-bit input operands are supported by this design in addition to a full set of extensions for signed and unsigned integers plus signed, fixed-point fractional input operands.

The MAC unit provides functionality in three related areas:

- Signed and unsigned integer multiplies
- Multiply-accumulate operations supporting signed, unsigned, and signed fractional operands
- Miscellaneous register operations

Each of the three areas of support is addressed in detail in the succeeding sections. Logic that supports this functionality is contained in a MAC module, as shown in Figure 3-1.

The MAC unit is tightly coupled to the OEP and features a three-stage execution pipeline. To minimize silicon costs, the ColdFire MAC is optimized for 16 x 16 multiply instructions. The OEP can issue a 16 x 16 multiply with a 32-bit accumulation and fetch a 32-bit operand in the same cycle. A 32 x 32 multiply with a 32-bit accumulation takes three cycles before the next instruction can be issued. Figure 3-1 shows the basic functionality of the ColdFire MAC. A full set of instructions is provided for signed and unsigned integers plus signed, fixed-point, fractional input operands.

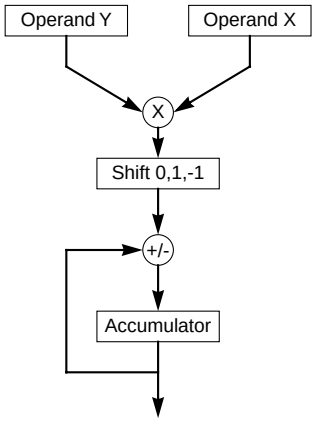


Figure 3-1. ColdFire MAC Multiplication and Accumulation

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

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

3.1.1 MAC Programming Model

Figure 3-2 shows the registers in the MAC portion of the user programming model.

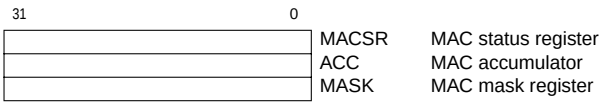


Figure 3-2. MAC Programming Model

These registers are described as follows:

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

3.1.2 General Operation

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

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

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

To maintain compactness, the MAC module is optimized for 16-bit multiplications. Two 16-bit operands produce a 32-bit product. Longword operations are performed by reusing the 16-bit multiplier array at the expense of a small amount of extra control logic. Again, the product of two 32-bit operands is a 32-bit result. For longword integer operations, only the least significant 32 bits of the product are calculated. For fractional operations, the entire 63-bit product is calculated and then either truncated or rounded to a 32-bit result using the round-to-nearest (even) method.

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

during the calculations.

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

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

The three program-visible MAC registers, a 32-bit accumulator (ACC), the MAC mask register (MASK), and MACSR, are described in Section 3.1.1, “MAC Programming Model.”

3.1.3 MAC Instruction Set Summary

The MAC unit supports the integer multiply operations defined by the baseline ColdFire architecture, as well as the new multiply-accumulate instructions. Table 3-1 summarizes the MAC unit instruction set.

Table 3-1. MAC Instruction Summary

Instruction	Mnemonic	Description
Multiply Signed	MULS <ea>y,Dx	Multiplies two signed operands yielding a signed result
Multiply Unsigned	MULU <ea>y,Dx	Multiplies two unsigned operands yielding an unsigned result
Multiply Accumulate	MAC Ry,RxSF MSAC Ry,RxSF	Multiplies two operands, then adds or subtracts the product to/from the accumulator
Multiply Accumulate with Load	MAC Ry,RxSF,Rw MSAC Ry,RxSF,Rw	Multiplies two operands, then adds or subtracts the product to/from the accumulator while loading a register with the memory operand
Load Accumulator	MOV.L {Ry,#imm},ACC	Loads the accumulator with a 32-bit operand
Store Accumulator	MOV.L ACC,Rx	Writes the contents of the accumulator to a register
Load MACSR	MOV.L {Ry,#imm},MACSR	Writes a value to the MACSR
Store MACSR	MOV.L MACSR,Rx	Write the contents of MACSR to a register
Store MACSR to CCR	MOV.L MACSR,CCR	Write the contents of MACSR to the processor's CCR register
Load MASK	MOV.L {Ry,#imm},MASK	Writes a value to MASK
Store MASK	MOV.L MASK,Rx	Writes the contents of MASK to a register

3.1.4 Data Representation

The MAC unit supports three basic operand types:

- Two's complement signed integer: In this format, an N-bit operand represents a number within the range $-2^{(N-1)} \leq$ operand $\leq 2^{(N-1)} - 1$. The binary point is to the right of the least significant bit.
- Two's complement unsigned integer: In this format, an N-bit operand represents a number within the range $0 \leq$ operand $\leq 2^{N} - 1$. The binary point is to the right of the least significant bit.
- Two's complement, signed fractional: In an N-bit number, the first bit is the sign bit. The remaining bits signify the first N-1 bits after the binary point. Given an N-bit number, $a_{N-1}a_{N-2}a_{N-3}... a_2a_1a_0$, its value is given by the following formula:

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

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

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

3.2 MAC Instruction Execution Timings

Table 3-2 shows standard timings for two-operand MAC instructions.

Table 3-2. Two-Operand MAC Instruction Execution Times

Opcode	í	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xi*SF)	(xxx).wl	#<xxx>
mac.w	Ry,Rx	1(0/0)	—	—	—	—	—	—	—
mac.l	Ry,Rx	3(0/0)	—	—	—	—	—	—	—
msac.w	Ry,Rx	1(0/0)	—	—	—	—	—	—	—
msac.l	Ry,Rx	3(0/0)	—	—	—	—	—	—	—
mac.w	Ry,Rx,ea,Rw	—	1(1/0)	1(1/0)	1(1/0)	1(1/0)	—	—	—
mac.l	Ry,Rx,ea,Rw	—	3(1/0)	3(1/0)	3(1/0)	3(1/0)	—	—	—
msac.w	Ry,Rx,ea,Rw	—	1(1/0)	1(1/0)	1(1/0)	1(1/0)	—	—	—
msac.l	Ry,Rx,ea,Rw	—	3(1/0)	3(1/0)	3(1/0)	3(1/0)	—	—	—
muls.w	<ea>,Dx	3(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	3(0/0)
mulu.w	<ea>,Dx	3(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	3(0/0)
muls.l	<ea>,Dx	5(0/0)	5(1/0)	5(1/0)	5(1/0)	5(1/0)	—	—	—
mulu.l	<ea>,Dx	5(0/0)	5(1/0)	5(1/0)	5(1/0)	5(1/0)	—	—	—

Table 3-3 shows standard timings for MAC move instructions.

Table 3-3. MAC Move Instruction Execution Times

Opcode	í	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xi*SF)	(xxx).wl	#<xxx>
move.l	<ea>,ACC	1(0/0)	—	—	—	—	—	—	1(0/0)
move.l	<ea>,MACSR	6(0/0)	—	—	—	—	—	—	6(0/0)
move.l	<ea>,MASK	5(0/0)	—	—	—	—	—	—	5(0/0)
move.l	ACC,Rx	1(0/0)	—	—	—	—	—	—	—
move.l	MACSR,CCR	1(0/0)	—	—	—	—	—	—	—
move.l	MACSR,Rx	1(0/0)	—	—	—	—	—	—	—
move.l	MASK,Rx	1(0/0)	—	—	—	—	—	—	—

Chapter 4
Local Memory

This chapter describes the MCF5307 implementation of the ColdFire Version 3 local memory specification. It consists of two major sections.

- Section 4.2, “SRAM Overview,” describes the MCF5307 on-chip static RAM (SRAM) implementation. It covers general operations, configuration, and initialization. It also provides information and examples showing how to minimize power consumption when using the SRAM.
- Section 4.7, “Cache Overview,” describes the MCF5307 cache implementation, including organization, configuration, and coherency. It describes cache operations and how the cache interfaces with other memory structures.

4.1 Interactions between Local Memory Modules

Depending on configuration information, instruction fetches and data read accesses may be sent simultaneously to the RAM and cache controllers. This approach is required because both controllers are memory-mapped devices and the hit/miss determination is made concurrently with the read data access. Power dissipation can be minimized by configuring the RAMBARs to mask unused address spaces whenever possible.

If the access address is mapped into the region defined by the RAM (and this region is not masked), the RAM provides the data back to the processor, and the cache data is discarded. Accesses from the RAM module are never cached. The complete definition of the processor’s local bus priority scheme for read references is as follows:

```
        if (RAM “hits”
)        RAM supplies data to the processor
               else if (cache “hits”)
                         cache supplies data to the processor
                  else system memory reference to access data
```

For data write references, the memory mapping into the local memories is resolved before the appropriate destination memory is accessed. Accordingly, only the targeted local memory is accessed for data write transfers.

4.2 SRAM Overview

The 4-Kbyte on-chip SRAM module is connected to the internal bus and provides pipelined, single-cycle access to memory mapped to the module. Memory can be mapped to any

0-modulo-32K location in the 4-Gbyte address space and configured to respond to either instruction or data accesses.Time-critical functions can be mapped into instruction the system stack. Other heavily-referenced data can be mapped into memory.

The following summarizes features of the MCF5307 SRAM implementation:

- 4-Kbyte SRAM, organized as 1024 x 32 bits
- Single-cycle throughput. When the pipeline is full, one access can occur per clock cycle.
- Physical location on the processor's high-speed local bus
- Memory location programmable on any 0-modulo-32K address boundary
- Byte, word, and longword address capabilities
- The RAM base address register (RAMBAR) defines the logical base address, attributes, and access types for the SRAM module.

4.3 SRAM Operation

The SRAM module provides a general-purpose memory block that the ColdFire processor can access with single-cycle throughput. The location of the memory block can be specified to any word-aligned address in the 4-Gbyte address space by RAMBAR[BA], described in Section 4.4.1, "SRAM Base Address Register (RAMBAR)." The memory is ideal for storing critical code or data structures or for use as the system stack. Because the SRAM module connects physically to the processor's high-speed local bus, it can service processor-initiated accesses or memory-referencing debug module commands.

Instruction fetches and data reads can be sent to both the cache and SRAM blocks simultaneously. If the reference is mapped into a region defined by the SRAM, the SRAM provides data to the processor and any cache data is discarded. Data accessed from the SRAM module are not cached.

Note also that the SRAM cannot be accessed by the on-chip DMAs. The on-chip system configuration allows concurrent core and DMA execution, where the core can reference code or data from the internal SRAM or cache while performing a DMA transfer.

Accesses are attempted in the following order:

1. SRAM
2. Cache (if space is defined as cacheable)
3. External access

4.4 SRAM Programming Model

The SRAM programming model consists of RAMBAR.

4.4.1 SRAM Base Address Register (RAMBAR)

The SRAM modules are configured through the RAMBAR, shown in Figure 4-1.

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

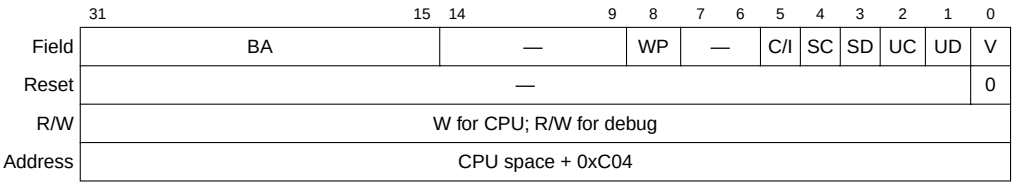


Figure 4-1. SRAM Base Address Register (RAMBAR)

RAMBAR fields are described in detail in Table 4-1.

Table 4-1. RAMBAR Field Description

Bits	Name	Description
31–15	BA	Base address. Defines the SRAM module's word-aligned base address. The SRAM module occupies a 4-Kbyte space defined by the contents of BA. SRAM may reside on any 32-Kbyte boundary in the 4-Gbyte address space.
14–9	—	Reserved, should be cleared.
8	WP	Write protect. Controls read/write properties of the SRAM. 0 Allows read and write accesses to the SRAM module 1 Allows only read accesses to the SRAM module. Any attempted write reference generates an access error exception to the ColdFire processor core.
7–6	—	Reserved, should be cleared.

Table 4-1. RAMBAR Field Description (Continued)

Bits	Name	Description
5–1	C/I, SC, SD, UC, UD	Address space masks (ASn). These fields allow certain types of accesses to be masked, or inhibited from accessing the SRAM module. These bits are useful for power management as described in Section 4.6, “Power Management.” In particular, C/I is typically set. The address space mask bits are follows: C/I = CPU space/interrupt acknowledge cycle mask. Note that C/I must be set if BA = 0. 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 ASn 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.
0	V	Valid. Enables/disables the SRAM module. V is cleared at reset. 0 RAMBAR contents are not valid. 1 RAMBAR contents are valid.

The mapping of a given access into the RAM uses the following algorithm to determine if the access hits in the memory:

```
if (RAMBAR[0] = 1)

if (requested address[31:15] = RAMBAR[31:15])
        if (requested address[14:12] = 0)
                if (ASn of the requested type = 0)
                        Access is mapped to the RAM module
                        if (access = read)
                                Read the RAM and return the data
                        if (access = write)
                                if (RAMBAR[8] = 0)
                                        Write the data into the RAM
                                else Signal a write-protect access error
```

AS*n* refers to the five address space mask bits: C/I, SC, SD, UC, and UD.

4.5 SRAM Initialization

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

1. Read the source data and write it to the SRAM. Various instructions support this function, including memory-to-memory move instructions and the move multiple instruction (MOVEM). MOVEM is optimized to generate line-sized burst fetches on line-aligned addresses, so it generally provides maximum performance.
2. After the data is loaded into the SRAM, it may be appropriate to revise the RAMBAR attribute bits, including the write-protect and address space mask fields.

Remember that the SRAM cannot be accessed by the on-chip DMAs. The on-chip system configuration allows concurrent core and DMA execution where the core can execute code

out of internal SRAM or cache during DMA access.

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

4.5.1 SRAM Initialization Code

The code segment below initializes the SRAM. The code sets the base address of the SRAM at 0x2000_0000 and then initializes the RAM to zeros.

```
RAMBASE         EQU     0x20000000      ;set this variable to 0x20000000
RAMVALID        EQU     0x00000035
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          #1024,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          ;exit if done; else continue looping
```

The following function copies the number of bytesToMove from the source (*src) to the processor's local RAM at an offset relative to the SRAM base address defined by destinationOffset. The bytesToMove must be a multiple of 16. For best performance, source and destination SRAM addresses should be line-aligned (0-modulo-16).

```
; copyToCpuRam (*src, destinationOffset, bytesToMove)


RAMBASE         EQU      0x20000000     ;SRAM base address
RAMFLAGS        EQU      0x00000035     ;RAMBAR valid + mask bits

        lea.l       -12(a7),a7          ;allocate temporary space
        movem.l     #0x1c,(a7)          ;store D2/D3/D4 registers

; stack arguments and locations
; +0     saved d2
; +4     saved d3
; +8     saved d4
; +12    returnPc
; +16    pointer to source operand
; +20    destinationOffset
; +24    bytesToMove

        move.l      RAMBASE+RAMFLAGS,a0 ;define RAMBAR contents
        movec.l     a0,rambar           ;load it

        move.l      16(a7),a0           ;load argument defining *src

        lea.l       RAMBASE,a1          ;memory pointer to RAM base
        add.l       20(a7),a1           ;include destinationOffset
```

```
        move.l      24(a7),d4           ;load byte count
        asr.l       #4,d4               ;divide by 16 to convert to loop count

        .align      4                   ;force loop on 0-mod-4 address
loop:   movem.l     (a0),#0xf           ;read 16 bytes from source
        movem.l     #0xf,(a1)           ;store into RAM destination
        lea.l       16(a0),a0           ;increment source pointer
        lea.l       16(a1),a1           ;increment destination pointer
        subq.l      #1,d4               ;decrement loop counter
        bne.b       loop                ;if done, then exit, else continue

        movem.l     (a7),#0x1c          ;restore d2/d3/d4 registers
        lea.l       12(a7),a7           ;deallocate temporary space
        rts
```

4.6 Power Management

Because processor memory references may be simultaneously sent to an SRAM module and cache, power can be minimized by configuring RAMBAR address space masks as precisely as possible. For example, if an SRAM is mapped to the internal instruction bus and contains instruction data, setting the AS*n* mask bits associated with operand references can decrease power dissipation. Similarly, if the SRAM contains data, setting AS*n* bits associated with instruction fetches minimizes power.

Table 4-2 shows typical RAMBAR configurations.

Table 4-2. Examples of Typical RAMBAR Settings

Data Contained in SRAM	RAMBAR[5–0]
Code only	0x2B
Data only	0x35
Both code and data	0x21

4.7 Cache Overview

This section describes the MCF5307 cache implementation, including organization, configuration, and coherency. It describes cache operations and how the cache interacts with other memory structures.

The MCF5307 processor contains a nonblocking, 8-Kbyte, 4-way set-associative, unified (instruction and data) cache with a 16-byte line size. The cache improves system performance by providing low-latency access to the instruction and data pipelines. This decouples processor performance from system memory performance, increasing bus availability for on-chip DMA or external devices. Figure 4-2 shows the organization and integration of the data cache.

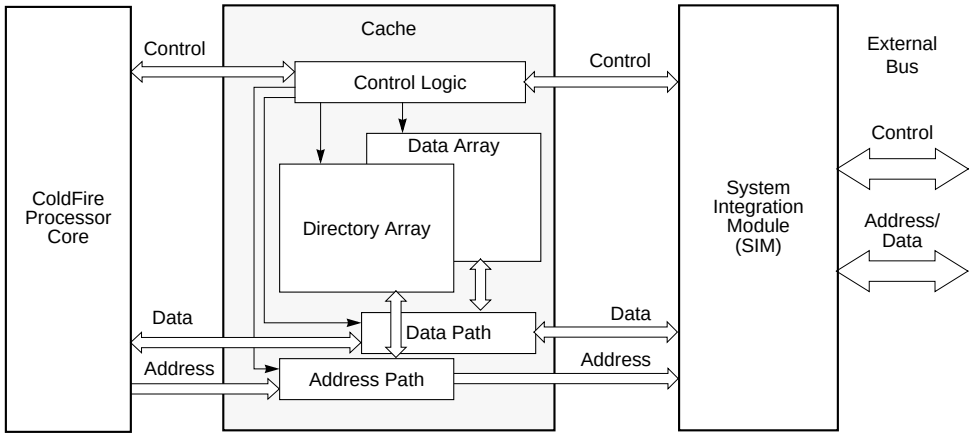


Figure 4-2. Unified Cache Organization

The cache supports operation of copyback, write-through, or cache-inhibited modes. The cache lock feature can be used to guarantee deterministic response for critical code or data areas.

A nonblocking cache services read hits or write hits from the processor while a fill (caused by a cache allocation) is in progress. As Figure 4-2 shows, instruction and data accesses use a single bus connected to the cache.

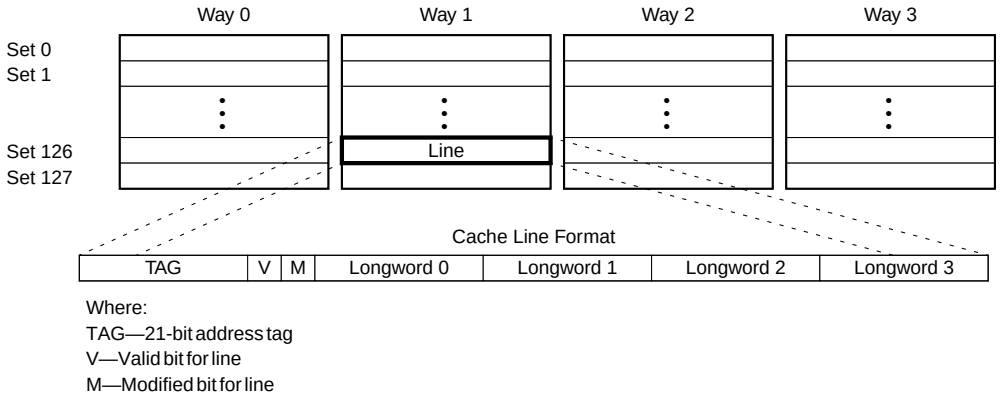
All addresses from the processor to the cache are physical addresses. A cache hit occurs when an address matches a cache entry. For a read, the cache supplies data to the processor. For a write, the processor updates the cache. If an access does not match a cache entry (misses the cache) or if a write access must be written through to memory, the cache performs a bus cycle on the internal bus and correspondingly on the external bus by way of the system integration module (SIM).

The SRAM module does not implement bus snooping; cache coherency with other possible bus masters must be maintained in software.

4.8 Cache Organization

A four-way set associative cache is organized as four ways (levels). There are 128 sets in the 8-Kbyte cache with each line containing 16 bytes (4 longwords). Entire cache lines are loaded from memory by burst-mode accesses that cache 4 longwords of data or instructions. All 4 longwords must be loaded for the cache line to be valid.

Figure 4-3 shows cache organization as well as terminology used.



Where:
TAG—21-bit address tag
V—Valid bit for line
M—Modified bit for line

Figure 4-3. Cache Organization and Line Format

A set is a group of four lines (one from each level, or way), corresponding to the same index into the cache array.

4.8.1 Cache Line States: Invalid, Valid-Unmodified, and Valid-Modified

As shown in Table 4-3, a cache line can be invalid, valid-unmodified (often called exclusive), or valid-modified.

Table 4-3. Valid and Modified Bit Settings

V	M	Description
0	x	Invalid. Invalid lines are ignored during lookups.
1	0	Valid, unmodified. Cache line has valid data that matches system memory.
1	1	Valid, modified. Cache line contains most recent data, data at system memory location is stale.

A valid line can be explicitly invalidated by executing a CPUSHL instruction.

4.8.2 The Cache at Start-Up

As Figure 4-4 (A) shows, after power-up, cache contents are undefined; V and M may be set on some lines even though the cache may not contain the appropriate data for start up. Because reset and power-up do not invalidate cache lines automatically, the cache should be cleared explicitly by setting CACR[CINVA] before the cache is enabled (B).

After the entire cache is flushed, cacheable entries are loaded first in way 0. If way 0 is occupied, the cacheable entry is loaded into the same set in way 1, as shown in Figure 4-4 (D). This process is described in detail in Section 4.9, “Cache Operation.”

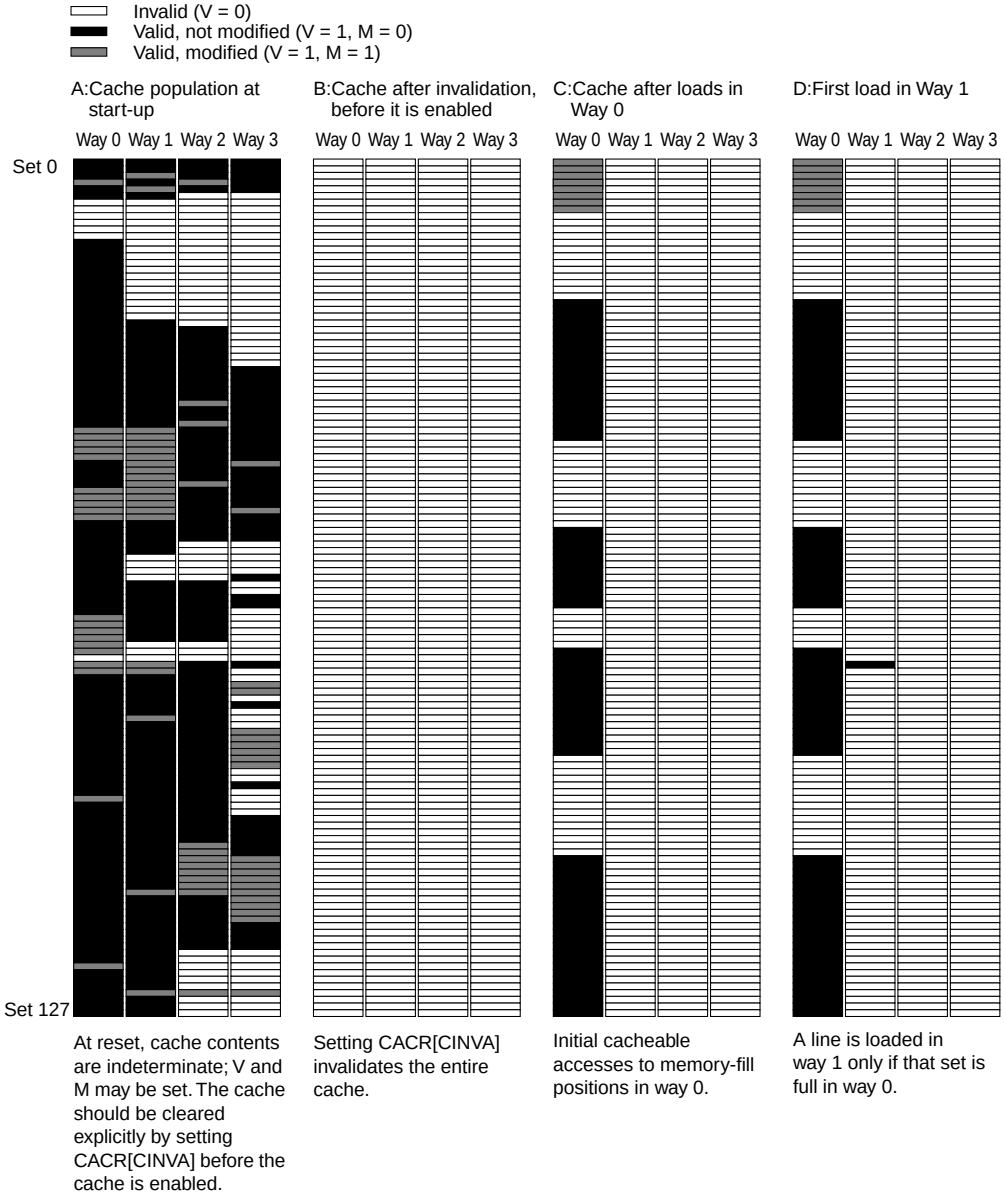


Figure 4-4. Cache—A: at Reset, B: after Invalidation, C and D: Loading Pattern

4.9 Cache Operation

Figure 4-5 shows the general flow of a caching operation.

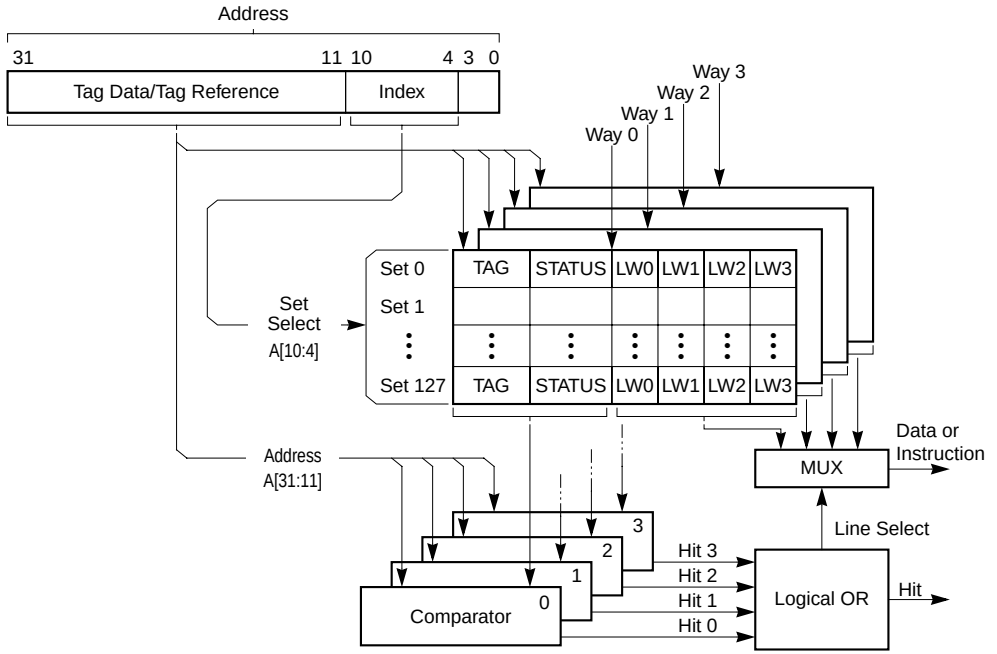


Figure 4-5. Caching Operation

The following steps determine if a cache line is allocated for a given address:

1. The cache set index, A[10:4], selects one cache set.
2. A[31:11] and the cache set index are used as a tag reference or are used to update the cache line tag field. Note that A[31:11] can specify 21 possible addresses that can be mapped to one of the four ways.
3. The four tags from the selected cache set are compared with the tag reference. A cache hit occurs if a tag matches the tag reference and the V bit is set, indicating that the cache line contains valid data. If a cacheable write access hits in a valid cache line, the write can occur to the cache line without having to load it from memory.

   If the memory space is copyback, the updated cache line is marked modified (M = 1), because the new data has made the data in memory out of date. If the memory location is write-through, the write is passed on to system memory and the M bit is never used. Note that the tag does not have TT or TM bits.

To allocate a cache entry, the cache set index selects one of the cache's 128 sets. The cache control logic looks for an invalid cache line to use for the new entry. If none is available, the cache controller uses a pseudo-round-robin replacement algorithm to choose the line to

be deallocated and replaced. First the cache controller looks for an invalid line, with way 0 the highest priority. If all lines have valid data, a 2-bit replacement counter is used to choose the way. After a line is allocated, the pointer increments to point to the next way.

Cache lines from ways 0 and 1 can be protected from deallocation by enabling half-cache locking. If CACR[HLCK] = 1, the replacement pointer is restricted to way 2 or 3.

As part of deallocation, a valid, unmodified cache line is invalidated. It is consistent with system memory, so memory does not need to be updated. To deallocate a modified cache line, data is placed in a push buffer (for an external cache line push) before being invalidated. After invalidation, the new entry can replace it. The old cache line may be written after the new line is read.

When a cache line is selected to host a new cache entry, the following three things happen:

1. The new address tag bits A[31:11] are written to the tag.
2. The cache line is updated with the new memory data.
3. The cache line status changes to a valid state (V = 1).

Read cycles that miss in the cache allocate normally as previously described.

Write cycles that miss in the cache do not allocate on a cacheable write-through region, but do allocate for addresses in a cacheable copyback region**.**

A copyback byte, word, longword, or line write miss causes the following:

1. The cache initiates a line fill or flush.
2. Space is allocated for a new line.
3. V and M are both set to indicate valid and modified.
4. Data is written in the allocated space. No write to memory occurs.

Note the following:

- Read hits cannot change the status bits and no deallocation or replacement occurs; the data or instructions are read from the cache.
- If the cache hits on a write access, data is written to the appropriate portion of the accessed cache line. Write hits in cacheable, write-through regions generate an external write cycle and the cache line is marked valid, but is never marked modified. Write hits in cacheable copyback regions do not perform an external write cycle; the cache line is marked valid and modified (V = 1 and M = 1).
- Misaligned accesses are broken into at least two cache accesses.
- Validity is provided only on a line basis. Unless a whole line is loaded on a cache miss, the cache controller does not validate data in the cache line.

Write accesses designated as cache-inhibited by the CACR or ACR bypass the cache and perform a corresponding external write.

Normally, cache-inhibited reads bypass the cache and are performed on the external bus. The exception to this normal operation occurs when all of the following conditions are true during a cache-inhibited read:

- The cache-inhibited fill buffer bit, CACR[DNFB], is set.
- The access is an instruction read.
- The access is normal (that is, transfer type (TT) equals 0).

In this case, an entire line is fetched and stored in the fill buffer. It remains valid there, and the cache can service additional read accesses from this buffer until either another fill or a cache-invalidate-all operation occurs.

Valid cache entries that match during cache-inhibited address accesses are neither pushed nor invalidated. Such a scenario suggests that the associated cache mode for this address space was changed. To avoid this, it is generally recommended to use the CPUSHL instruction to push or invalidate the cache entry or set CACR[CINVA] to invalidate the cache before switching cache modes.

4.9.1 Caching Modes

For every memory reference generated by the processor or debug module, a set of effective attributes is determined based on the address and the ACRs. Caching modes determine how the cache handles an access. An access can be cacheable in either write-through or copyback mode; it can be cache-inhibited in precise or imprecise modes. For normal accesses, the ACR*n*[CM] bit corresponding to the address of the access specifies the caching modes. If an address does not match an ACR, the default caching mode is defined by CACR[DCM]. 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
```

Addresses matching an ACR can also be write-protected using ACR[W]. Addresses that do not match either ACR can be write-protected using CACR[DW].

Reset disables the cache and clears all CACR bits. As shown in Figure 4-4, reset does not automatically invalidate cache entries; they must be invalidated through software.

The ACRs allow the defaults selected in the CACR to be overridden. In addition, some instructions (for example, CPUSHL) and processor core operations perform accesses that have an implicit caching mode associated with them. The following sections discuss the different caching accesses and their associated cache modes.

4.9.1.1 Cacheable Accesses

If ACR*n*[CM] or the default field of the CACR indicates write-through or copyback, the access is cacheable. A read access to a write-through or copyback region is read from the

cache if matching data is found. Otherwise, the data is read from memory and the cache is updated. When a line is being read from memory for either a write-through or copyback read miss, the longword within the line that contains the core-requested data is loaded first and the requested data is given immediately to the processor, without waiting for the three remaining longwords to reach the cache.

The following sections describe write-through and copyback modes in detail.

4.9.1.2 Write-Through Mode

Write accesses to regions specified as write-through are always passed on to the external bus, although the cycle can be buffered, depending on the state of CACR[ESB]. Writes in write-through mode are handled with a no-write-allocate policy—that is, writes that miss in the cache are written to the external bus but do not cause the corresponding line in memory to be loaded into the cache. Write accesses that hit always write through to memory and update matching cache lines. The cache supplies data to data-read accesses that hit in the cache; read misses cause a new cache line to be loaded into the cache.

4.9.1.3 Copyback Mode

Copyback regions are typically used for local data structures or stacks to minimize external bus use and reduce write-access latency. Write accesses to regions specified as copyback that hit in the cache update the cache line and set the corresponding M bit without an external bus access.

Be sure to flush the cache using the CPUSHL instruction before invalidating the cache in copyback mode. Modified cache data is written to memory only if the line is replaced because of a miss or a CPUSHL instruction pushes the line. If a byte, word, longword, or line write access misses in the cache, the required cache line is read from memory, thereby updating the cache. When a miss selects a modified cache line for replacement, the modified cache data moves to the push buffer. The replacement line is read into the cache and the push buffer contents are then written to memory.

4.9.2 Cache-Inhibited Accesses

Memory regions can be designated as cache-inhibited, which is useful for memory containing targets such as I/O devices and shared data structures in multiprocessing systems. It is also important to not cache the MCF5307 memory mapped registers. If the corresponding ACR*n*[CM] or CACR[DCM] indicates cache-inhibited, precise or imprecise, the access is cache-inhibited. The caching operation is identical for both cache-inhibited modes, which differ only regarding recovery from an external bus error.

In determining whether a memory location is cacheable or cache-inhibited, the CPU checks memory-control registers in the following order:

1. RAMBAR
2. ACR0

3. ACR1
4. If an access does not hit in the RAMBAR or the ACRs, the default is provided for all accesses in CACR.

Cache-inhibited write accesses bypass the cache and a corresponding external write is performed. Cache-inhibited reads bypass the cache and are performed on the external bus, except when all of the following conditions are true:

- The cache-inhibited fill-buffer bit, CACR[DNFB], is set.
- The access is an instruction read.
- The access is normal (that is, TT = 0).

In this case, a fetched line is stored in the fill buffer and remains valid there; the cache can service additional read accesses from this buffer until another fill occurs or a cache-invalidate-all operation occurs.

If ACR*n*[CM] indicates cache-inhibited mode, precise or imprecise, the controller bypasses the cache and performs an external transfer. If a line in the cache matches the address and the mode is cache-inhibited, the cache does not automatically push the line if it is modified, nor does it invalidate the line if it is valid. Before switching cache mode, execute a CPUSHL instruction or set CACR[CINVA] to invalidate the entire cache.

If ACR*n*[CM] indicates precise mode, the sequence of read and write accesses to the region is guaranteed to match the instruction sequence. In imprecise mode, the processor core allows read accesses that hit in the cache to occur before completion of a pending write from a previous instruction. Writes are not deferred past data-read accesses that miss the cache (that is, that must be read from the bus).

Precise operation forces data-read accesses for an instruction to occur only once by preventing the instruction from being interrupted after data is fetched. Otherwise, if the processor is not in precise mode, an exception aborts the instruction and the data may be accessed again when the instruction is restarted. These guarantees apply only when ACR*n*[CM] indicates precise mode and aligned accesses.

CPU space-register accesses, such as MOVEC, are treated as cache-inhibited and precise.

4.9.3 Cache Protocol

The following sections describe the cache protocol for processor accesses and assumes that the data is cacheable (that is, write-through or copyback).

4.9.3.1 Read Miss

A processor read that misses in the cache requests the cache controller to generate a bus transaction. This bus transaction reads the needed line from memory and supplies the required data to the processor core. The line is placed in the cache in the valid state.

4.9.3.2 Write Miss

The cache controller handles processor writes that miss in the cache differently for write-through and copyback regions. Write misses to copyback regions cause the cache line to be read from system memory, as shown in Figure 4-6.

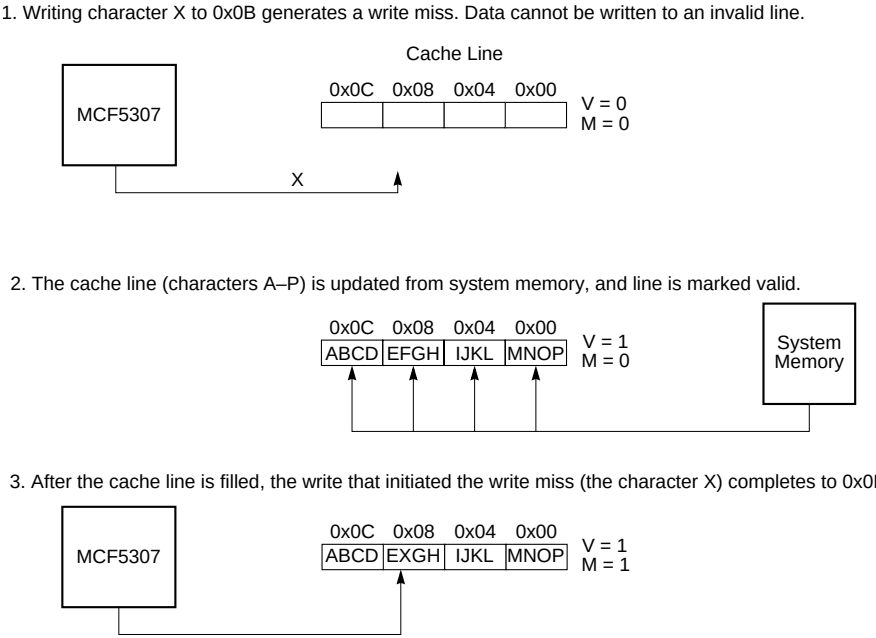


Figure 4-6. Write-Miss in Copyback Mode

The new cache line is then updated with write data and the M bit is set for the line, leaving it in modified state. Write misses to write-through regions write directly to memory without loading the corresponding cache line into the cache.

4.9.3.3 Read Hit

On a read hit, the cache provides the data to the processor core and the cache line state remains unchanged. If the cache mode changes for a specific region of address space, lines in the cache corresponding to that region that contain modified data are not pushed out to memory when a read hit occurs within that line. First execute a CPUSHL instruction or set CACR[CINVA] before switching the cache mode.

4.9.3.4 Write Hit

The cache controller handles processor writes that hit in the cache differently for write-through and copyback regions. For write hits to a write-through region, portions of cache lines corresponding to the size of the access are updated with the data. The data is

also written to external memory. The cache line state is unchanged. For copyback accesses, the cache controller updates the cache line and sets the M bit for the line. An external write is not performed and the cache line state changes to (or remains in) the modified state.

4.9.4 Cache Coherency

The MCF5307 provides limited cache coherency support in multiple-master environments. Both write-through and copyback memory update techniques are supported to maintain coherency between the cache and memory.

The cache does not support snooping (that is, cache coherency is not supported while external or DMA masters are using the bus). Therefore, on-chip DMAs of the MCF5307 cannot access local memory and do not maintain coherency with the unified cache.

4.9.5 Memory Accesses for Cache Maintenance

The cache controller performs all maintenance activities that supply data from the cache to the core, including requests to the SIM for reading new cache lines and writing modified lines to memory. The following sections describe memory accesses resulting from cache fill and push operations. Chapter 18, “Bus Operation,” describes required bus cycles in detail.

4.9.5.1 Cache Filling

When a new cache line is required, a line read is requested from the SIM, which generates a burst-read transfer by indicating a line access with the size signals, SIZ[1:0].

The responding device supplies 4 consecutive longwords of data. Burst operations can be inhibited or enabled through the burst read/write enable bits (BSTR/BSTW) in the chip-select control registers (CSCR0–CSCR7).

SIM line accesses implicitly request burst-mode operations from memory. For more information regarding external bus burst-mode accesses, see Chapter 18, “Bus Operation.”

The first cycle of a cache-line read loads the longword entry corresponding to the requested address. Subsequent transfers load the remaining longword entries.

A burst operation is aborted by an a write-protection fault, which is the only possible access error. Exception processing proceeds immediately. Because the write cycle can be decoupled from the processor’s issuing of the operation, error signaling appears to be decoupled from the instruction that generated the write. Accordingly, the PC in the exception stack frame represents the program location when the access error was signaled. See Section 2.8.2, “Processor Exceptions.”

4.9.5.2 Cache Pushes

Cache pushes occur for line replacement and as required for the execution of the CPUSHL instruction. To reduce the requested data's latency in the new line, the modified line being replaced is temporarily placed in the push buffer while the new line is fetched from memory. After the bus transfer for the new line completes, the modified cache line is written back to memory and the push buffer is invalidated.

4.9.5.2.1 Push and Store Buffers

The 16-byte push buffer reduces latency for requested new data on a cache miss by holding a displaced modified cache line while the new data is read from memory.

If a cache miss displaces a modified line, a miss read reference is immediately generated. While waiting for the response, the current contents of the cache location load into the push buffer. When the burst-read bus transaction completes, the cache controller can generate the appropriate line-write bus transaction to write the push buffer contents into memory.

In imprecise mode, the FIFO store buffer can defer pending writes to maximize performance. The store buffer can support as many as four entries (16 bytes maximum) for this purpose.

Data writes destined for the store buffer cannot stall the core. The store buffer effectively provides a measure of decoupling between the pipeline's ability to generate writes (one per cycle maximum) and the external bus's ability to retire those writes. In imprecise mode, writes stall only if the store buffer is full and a write operation is on the internal bus. The internal write cycle is held, stalling the data execution pipeline.

If the store buffer is not used (that is, store buffer disabled or cache-inhibited precise mode), external bus cycles are generated directly for each pipeline write operation. The instruction is held in the pipeline until external bus transfer termination is received. Therefore, each write is stalled for 5 cycles, making the minimum write time equal to 6 cycles when the store buffer is not used. See Section 2.1.2.2, "Operand Execution Pipeline (OEP)."

The store buffer enable bit, CACR[ESB], controls the enabling of the store buffer. This bit can be set and cleared by the MOVEC instruction. ESB is zero at reset and all writes are performed in order (precise mode). ACR*n*[CM] or CACR[DCM] generates the mode used when ESB is set. Cacheable write-through and cache-inhibited imprecise modes use the store buffer.

The store buffer can queue data as much as 4 bytes wide per entry. Each entry matches the corresponding bus cycle it generates; therefore, a misaligned longword write to a write-through region creates two entries if the address is to an odd-word boundary. It creates three entries if it is to an odd-byte boundary—one per bus cycle.

4.9.5.2.2 Push and Store Buffer Bus Operation

As soon as the push or store buffer has valid data, the internal bus controller uses the next available external bus cycle to generate the appropriate write cycles. In the event that

another cache fill is required (for example, cache miss to process) during the continued instruction execution by the processor pipeline, the pipeline stalls until the push and store buffers are empty, then generate the required external bus transaction.

Supervisor instructions, the NOP instruction, and exception processing synchronize the processor core and guarantee the push and store buffers are empty before proceeding. Note that the NOP instruction should be used only to synchronize the pipeline. The preferred no-operation function is the TPF instruction.

4.9.6 Cache Locking

Ways 0 and 1 of the cache can be locked by setting CACR[HLCK]. If the cache is locked, cache lines in ways 0 and 1 are not subject to being deallocated by normal cache operations.

As Figure 4-7 (B and C) shows, the algorithm for updating the cache and for identifying cache lines to be deallocated is otherwise unchanged. If ways 2 and 3 are entirely invalid, cacheable accesses are first allocated in way 2. Way 3 is not used until the location in way 2 is occupied.

Ways 0 and 1 are still updated on write hits (D in Figure 4-7) and may be pushed or cleared only explicitly by using specific cache push/invalidate instructions. However, new cache lines cannot be allocated in ways 0 and 1.

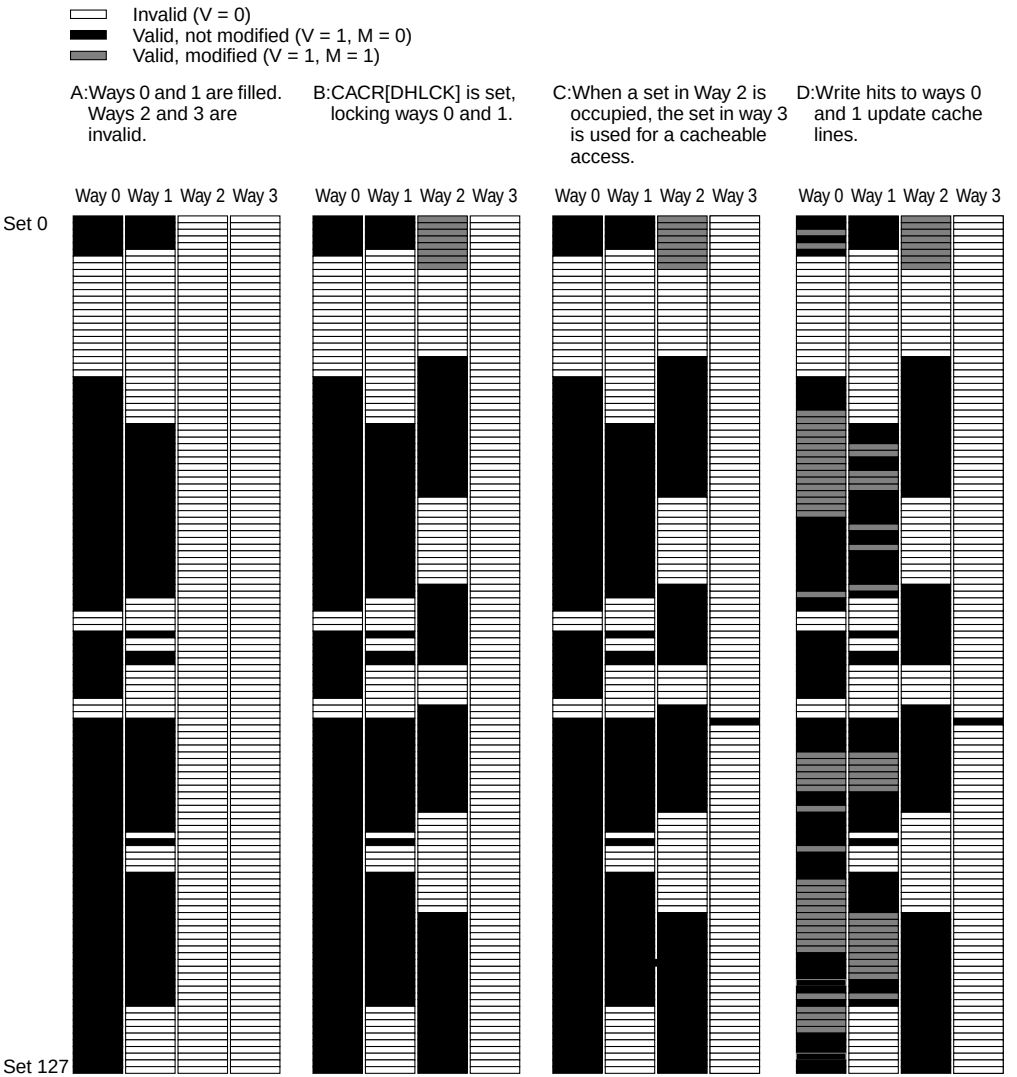


Figure 4-7. Cache Locking

4.10 Cache Registers

This section describes the MCF5307 implementation of the Version 3 cache registers.

4.10.1 Cache Control Register (CACR)

The CACR in Figure 4-8 contains bits for configuring the cache. It can be written by the MOVEC register instruction and can be read or written from the debug facility. A hardware reset clears CACR, which disables the cache; however, reset does not affect the tags, state information, or data in the cache.

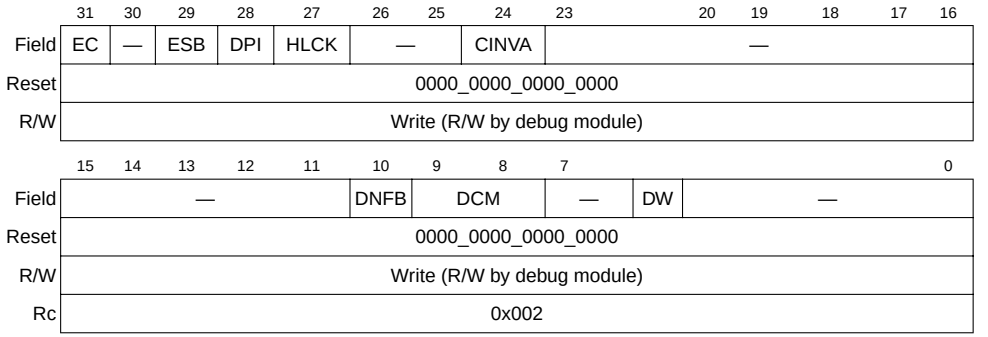


Figure 4-8. Cache Control Register (CACR)

Table 4-4 describes CACR fields.

Table 4-4. CACR Field Descriptions

Bits	Name	Description
31	EC	Enable cache. 0 Cache disabled. The cache is not operational, but data and tags are preserved. 1 Cache enabled.
30	—	Reserved, should be cleared.
29	ESB	Enable store buffer. 0 Writes to write-through or noncachable in imprecise mode bypass the store buffer and generate bus cycles directly. Section 4.9.5.2.1, “Push and Store Buffers,” describes the performance penalty for this. 1 The four-entry FIFO store buffer is enabled; when imprecise mode is used, this buffer defers pending writes to write-through or cache-inhibited regions to maximize performance. Cache-inhibited, precise-mode accesses always bypass the store buffer.
28	DPI	Disable CPUSHL invalidation. 0 Normal operation. A CPUSHL instruction causes the selected line to be pushed if modified and then invalidated. 1 No clear operation. A CPUSHL instruction causes the selected line to be pushed if modified, then left valid.

Table 4-4. CACR Field Descriptions (Continued)

Bits	Name	Description
27	HLCK	Half-cache lock mode 0 Normal operation. The cache allocates the lowest invalid way. If all ways are valid, the cache allocates the way pointed at by the counter and then increments this counter modulo-4. 1 Half-cache operation. The cache allocates to the lower invalid way of levels 2 and 3; if both are valid, the cache allocates to way 2 if the high-order bit of the round-robin counter is zero; otherwise, it allocates way 3 and increments the round-robin counter modulo-2. This locks the content of ways 0 and 1. Ways 0 and 1 are still updated on write hits and may be pushed or cleared by specific cache push/invalidate instructions. This implementation allows maximum use of available cache memory and provides the flexibility of setting HLCK before, during, or after allocations occur.
26–25	—	Reserved, should be cleared.
24	CINVA	Cache invalidate all. Writing a 1 to this bit initiates entire cache invalidation. Once invalidation is complete, this bit automatically returns to 0; it is not necessary to clear it explicitly. Note the caches are not cleared on power-up or normal reset, as shown in Figure 4-4. 0 No invalidation is performed. 1 Initiate invalidation of the entire cache. The cache controller sequentially clears V and M bits in all sets. Subsequent accesses stall until the invalidation is finished, at which point, this bit is automatically cleared. In copyback mode, the cache should be flushed using a CPUSHL instruction before setting this bit.
23–11	—	Reserved, should be cleared.
10	DNFB	Default noncacheable fill buffer. Determines if the fill buffer can store noncacheable accesses 0 Fill buffer not used to store noncacheable instruction accesses (16 or 32 bits). 1 Fill buffer used to store noncacheable accesses. The fill buffer is used only for normal (TT = 0) instruction reads of a noncacheable region. Instructions are loaded into the fill buffer by a burst access (same as a line fill). They stay in the buffer until they are displaced, so subsequent accesses may not appear on the external bus. Note that this feature can cause a coherency problem for self-modifying code. If DNFB = 1 and a cache-inhibited access uses the fill buffer, instructions remain valid in the fill buffer until a cache-invalidate-all instruction, another cache-inhibited burst, or a miss that initiates a fill. A write to the line in the fill buffer goes to the external bus without updating or invalidating the buffer. Subsequent reads of that written data are serviced by the fill buffer and receive stale information.
9–8	DCM	Default cache mode. Selects the default cache mode and access precision as follows: 00 Cacheable, write-through 01 Cacheable, copy-back 10 Cache-inhibited, precise exception model 11 Cache-inhibited, imprecise exception model. Precise and imprecise modes are described in Section 4.9.2, “Cache-Inhibited Accesses.”
7–6	—	Reserved, should be cleared.
5	DW	Default write protect. Use of this bit is described in Section 4.9.1, “Caching Modes.” 0 Read and write accesses permitted 1 Write accesses not permitted
4–0	—	Reserved, should be cleared.

4.10.2 Access Control Registers (ACR0–ACR1)

The ACRs, Figure 4-9, assign control attributes, such as cache mode and write protection, to specified memory regions. Registers are accessed with the MOVEC instruction with the Rc encodings in Figure 4-9.

For overlapping regions, ACR0 takes priority. Data transfers to and from these registers are longword transfers. Bits 12–7, 4, 3, 1, and 0 are always read as zeros.

NOTE:

The SIM MBAR region should be mapped as cache-inhibited through an ACR.

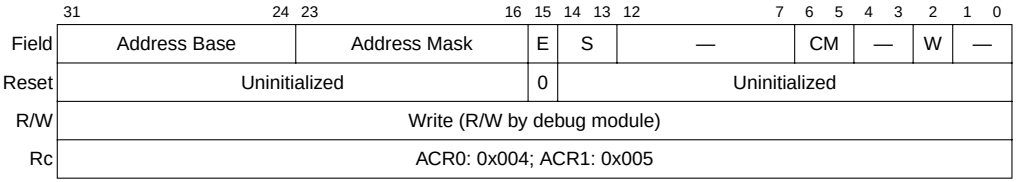


Figure 4-9. Access Control Register Format (ACR*n*)

Table 4-5 describes ACR*n* fields.

Table 4-5. ACR*n* Field Descriptions

Bits	Name	Description
31–24	Address base	Address base. Compared with address bits A[31:24]. Eligible addresses that match are assigned the access control attributes of this register.
23–16	Address mask	Address mask. Setting a mask bit causes the corresponding address base bit to be ignored. The low-order mask bits can be set to define contiguous regions larger than 16 Mbytes. The mask can define multiple noncontiguous regions of memory.
15	E	Enable. Enables or disables the other ACRn bits. 0 Access control attributes disabled 1 Access control attributes enabled
14–13	S	Supervisor mode. Specifies whether only user or supervisor accesses are allowed in this address range or if the type of access is a don't care. 00 Match addresses only in user mode 01 Match addresses only in supervisor mode 1x Execute cache matching on all accesses
12–7	—	Reserved; should be cleared.
6–5	CM	Cache mode. Selects the cache mode and access precision. Precise and imprecise modes are described in Section 4.9.2, "Cache-Inhibited Accesses." 00 Cacheable, write-through 01 Cacheable, copyback 10 Cache-inhibited, precise 11 Cache-inhibited, imprecise
4–3	—	Reserved, should be cleared.
2	W	Write protect. Selects the write privilege of the memory region. 0 Read and write accesses permitted 1 Write accesses not permitted
1–0	—	Reserved, should be cleared.

4.11 Cache Management

The cache can be enabled and configured by using a MOVEC instruction to access CACR. A hardware reset clears CACR, disabling the cache and removing all configuration information; however, reset does not affect the tags, state information, and data in the cache.

Set CACR[CINVA] to invalidate the cache before enabling it.

The privileged CPUSHL instruction supports cache management by selectively pushing and invalidating cache lines. The address register used with CPUSHL directly addresses the cache's directory array. The CPUSHL instruction flushes a cache line.

The value of CACR[DPI] determines whether CPUSHL invalidates a cache line after it is pushed. To push the entire cache, implement a software loop to index through all sets and through each of the four lines within each set (a total of 512 lines). The state of CACR[EC] does not affect the operation of CPUSHL or CACR[CINVA]. Disabling the cache by setting CACR[EC] makes the cache nonoperational without affecting tags, state information, or contents.

The contents of A*n* used with CPUSHL specify cache row and line indexes. This differs from the MC68040 where a physical address is specified. Figure 4-10 shows the A*n* format.

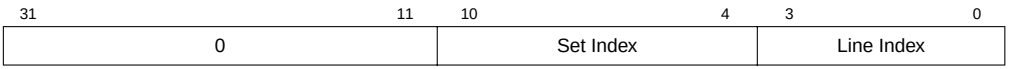


Figure 4-10. A*n* Format

The following code example flushes the entire cache:

```
_cache_disable:
        nop
        move.w          #0x2700,SR      ;mask off IRQs
        jsr             _cache_flush    ;flush the cache completely
        clr.l           d0
        movec           d0,ACR0         ;ACR0 off
        movec           d0,ACR1         ;ACR1 off
        move.l          #0x01000000,d0  ;Invalidate and disable cache
        movec           d0,CACR
        rts

_cache_flush:
        nop                             ;synchronize—flush store buffer
        moveq.l         #0,d0           ;initialize way counter
        moveq.l         #0,d1           ;initialize set counter
        move.l          d0,a0           ;initialize cpushl pointer

setloop:
        cpushl          bc,(a0)         ;push cache line a0
        add.l           #0x0010,a0      ;increment set index by 1
        addq.l          #1,d1           ;increment set counter
        cmpi.l          #128,d1         ;are sets for this way done?
        bne             setloop

        moveq.l         #0,d1           ;set counter to zero again
        addq.l          #1,d0           ;increment to next way
```

```
        move.l          d0,a0           ;set = 0, way = d0
        cmpi.l          #4,d0           ;flushed all the ways?
        bne             setloop
        rts
```

The following CACR loads assume the default cache mode is copyback.

```
CacheLoadAndLock:

        move.l  #0xA1000100,d0; enable and invalidate cache ...
        movec   d0,cacr ; ... in the CACR
```

The following code preloads half of the cache (4 Kbytes). It assumes a contiguous block of data is to be mapped into the cache, starting at a 0-modulo-4K address.

```
        move.l  #256,d0         ;256 16-byte lines in 4K space
        lea     data_,a0        ; load pointer defining data area
CacheLoop:
        tst.b   (a0)            ;touch location + load into data cache
        lea     16(a0),a0       ;increment address to next line
        subq.l  #1,d0           ;decrement loop counter
        bne.b   CacheLoop       ;if done, then exit, else continue

; A 4K region has been loaded into levels 0 and 1 of the 8K cache. lock it!

        move.l  #0xA8000100,d0  ;set the cache lock bit ...
        movec   d0,cacr         ; ... in the CACR
        rts

        align   16
```

4.12 Cache Operation Summary

This section gives operational details for the cache and presents cache-line state diagrams.

4.12.1 Cache State Transitions

Using the V and M bits, the cache supports a line-based protocol allowing individual cache lines to be invalid, valid, or modified. To maintain memory coherency, the cache supports both write-through and copyback modes, specified by the corresponding ACR[CM], or CACR[DCM] if no ACR matches.

Read or write misses to copyback regions cause the cache controller to read a cache line from memory into the cache. If available, tag and data from memory update an invalid line in the selected set. The line state then changes from invalid to valid by setting the V bit. If all lines in the row are already valid or modified, the pseudo-round-robin replacement algorithm selects one of the four lines and replaces the tag and data. Before replacement, modified lines are temporarily buffered and later copied back to memory after the new line has been read from memory.

Figure 4-11 shows the three possible cache line states and possible processor-initiated transitions for memory configured as copyback. Transitions are labeled with a capital letter indicating the previous state and a number indicating the specific case listed in Table 4-11.

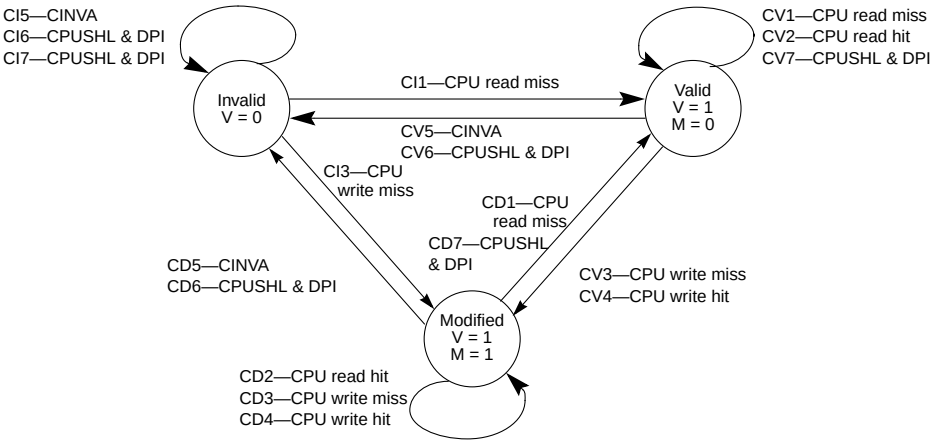


Figure 4-11. Cache Line State Diagram—Copyback Mode

Figure 4-12 shows the two possible states for a cache line in write-through mode.

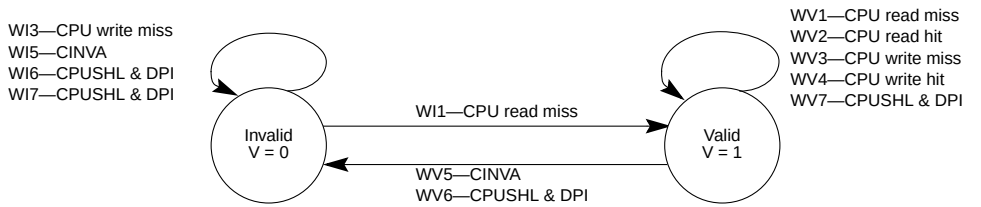


Figure 4-12. Cache Line State Diagram—Write-Through Mode

Table 4-6 describes cache line transitions and the accesses that cause them.

Table 4-6. Cache Line State Transitions

Access	Current State					
	Invalid (V = 0)		Valid (V = 1, M = 0)		Modified (V = 1, M = 1)	
Read miss	(C,W)I1	Read line from memory and update cache; supply data to processor; go to valid state.	(C,W)V1	Read new line from memory and update cache; supply data to processor; stay in valid state.	CD1	Push modified line to buffer; read new line from memory and update cache; supply data to processor; write push buffer contents to memory; go to valid state.
Read hit	(C,W)I2	Not possible.	(C,W)V2	Supply data to processor; stay in valid state.	CD2	Supply data to processor; stay in modified state.
Write miss (copy-back)	CI3	Read line from memory and update cache; write data to cache; go to modified state.	CV3	Read new line from memory and update cache; write data to cache; go to modified state.	CD3	Push modified line to buffer; read new line from memory and update cache; write push buffer contents to memory; stay in modified state.
Write miss (write-through)	WI3	Write data to memory; stay in invalid state.	WV3	Write data to memory; stay in valid state.	WD3	Write data to memory; stay in modified state. Cache mode changed for the region corresponding to this line. To avoid this state, execute a CPUSHL instruction or set CACR[CINVA] before switching modes.
Write hit (copy-back)	CI4	Not possible.	CV4	Write data to cache; go to modified state.	CD4	Write data to cache; stay in modified state.
Write hit (write-through)	WI4	Not possible.	WV4	Write data to memory and to cache; stay in valid state.	WD4	Write data to memory and to cache; go to valid state. Cache mode changed for the region corresponding to this line. To avoid this state, execute a CPUSHL instruction or set CACR[CINVA] before switching modes.
Cache invalidate	(C,W)I5	No action; stay in invalid state.	(C,W)V5	No action; go to invalid state.	CD5	No action (modified data lost); go to invalid state.
Cache push	(C,W)I6 (C,W)I7	No action; stay in invalid state.	(C,W)V6	No action; go to invalid state.	CD6	Push modified line to memory; go to invalid state.
			(C,W)V7	No action; stay in valid state.	CD7	Push modified line to memory; go to valid state.

The following tables present the same information as Table 4-6, organized by the current state of the cache line. In Table 4-7 the current state is invalid.

Table 4-7. Cache Line State Transitions (Current State Invalid)

Access	Response	
Read miss	(C,W)I1	Read line from memory and update cache; supply data to processor; go to valid state.
Read hit	(C,W)I2	Not possible
Write miss (copyback)	CI3	Read line from memory and update cache; write data to cache; go to modified state.
Write miss (write-through)	WI3	Write data to memory; stay in invalid state.
Write hit (copyback)	CI4	Not possible
Write hit (write-through)	WI4	Not possible
Cache invalidate	(C,W)I5	No action; stay in invalid state.
Cache push	(C,W)I6	No action; stay in invalid state.
Cache push	(C,W)I7	No action; stay in invalid state.

In Table 4-8 the current state is valid.

Table 4-8. Cache Line State Transitions (Current State Valid)

Access	Response	
Read miss	(C,W)V1	Read new line from memory and update cache; supply data to processor; stay in valid state.
Read hit	(C,W)V2	Supply data to processor; stay in valid state.
Write miss (copyback)	CV3	Read new line from memory and update cache; write data to cache; go to modified state.
Write miss (write-through)	WV3	Write data to memory; stay in valid state.
Write hit (copyback)	CV4	Write data to cache; go to modified state.
Write hit (write-through)	WV4	Write data to memory and to cache; stay in valid state.
Cache invalidate	(C,W)V5	No action; go to invalid state.
Cache push	(C,W)V6	No action; go to invalid state.
Cache push	(C,W)V7	No action; stay in valid state.

In the current state is modified.

Table 4-9. Cache Line State Transitions (Current State Modified)

Access	Response	
Read miss	CD1	Push modified line to buffer; read new line from memory and update cache; supply data to processor; write push buffer contents to memory; go to valid state.
Read hit	CD2	Supply data to processor; stay in modified state.
Write miss (copyback)	CD3	Push modified line to buffer; read new line from memory and update cache; write push buffer contents to memory; stay in modified state.
Write miss (write-through)	WD3	Write data to memory; stay in modified state. Cache mode changed for the region corresponding to this line. To avoid this state, execute a CPUSHL instruction or set CACR[CINVA] before switching modes.
Write hit (copyback)	CD4	Write data to cache; stay in modified state.
Write hit (write-through)	WD4	Write data to memory and to cache; go to valid state. Cache mode changed for the region corresponding to this line. To avoid this state, execute a CPUSHL instruction or set CACR[CINVA] before switching modes.
Cache invalidate	CD5	No action (modified data lost); go to invalid state.
Cache push	CD6	Push modified line to memory; go to invalid state.
Cache push	CD7	Push modified line to memory; go to valid state.

4.13 Cache Initialization Code

The following example sets up the cache for FLASH or ROM space only.

```
move.l#0x81000300,D0 //enable cache, invalidate it,
                     //default mode is cache-inhibited imprecise
movecD0, CACR

move.l #0xFF00C000,D0//cache FLASH space, enable,
                     //ignore FC2, cacheable, writethrough
movecD0,ACR0
```

Chapter 5
Debug Support

This chapter describes the Revision B enhanced hardware debug support in the MC5307. This revision of the ColdFire debug architecture encompasses the earlier revision.

5.1 Overview

The debug module is shown in Figure 5-1.

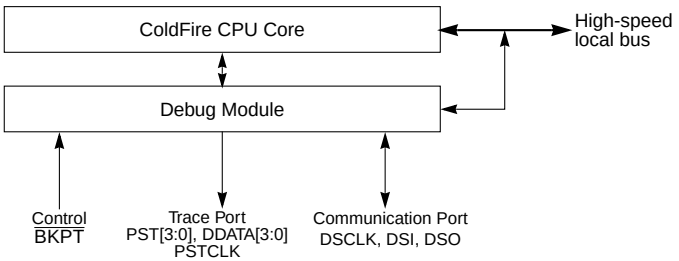


Figure 5-1. Processor/Debug Module Interface

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 5.3, “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 5.5, “Background Debug Mode (BDM),” and Section 5.4, “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. The emulator can access saved data because the hardware supports concurrent operation of the processor and BDM-initiated commands. See Section 5.6, “Real-Time Debug Support.”

The Version 2 ColdFire core implemented the original debug architecture, now called Revision A. Based on feedback from customers and third-party developers, enhancements have been added to succeeding generations of ColdFire cores. The Version 3 core implements Revision B of the debug architecture, providing more flexibility for configuring the hardware breakpoint trigger registers and removing the restrictions involving concurrent BDM processing while hardware breakpoint registers are active.

5.2 Signal Description

Table 5-1 describes debug module signals. All ColdFire debug signals are unidirectional and related to a rising edge of the processor core's clock signal. The standard 26-pin debug connector is shown in Section 5.7, "Motorola-Recommended BDM Pinout."

Table 5-1. 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 CLKIN edges.) Clocks the serial communication port to the debug module. Maximum frequency is 1/5 the processor CLK 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/debug data signals (PST[3:0]) as the value 0xF. If CSR[BKD] is set (disabling normal $\overline{\text{BKPT}}$ functionality), asserting $\overline{\text{BKPT}}$ generates a debug interrupt exception in the processor.
Processor Status Clock (PSTCLK)	Delayed version of the processor clock. Its rising edge appears in the center of valid PST and DDATA output. See Figure 5-2. PSTCLK indicates when the development system should sample PST and DDATA values. If real-time trace is not used, setting CSR[PCD] keeps PSTCLK, PST and DDATA outputs from toggling without disabling triggers. Non-quiescent operation can be reenabled by clearing CSR[PCD], although the emulator must resynchronize with the PST and DDATA outputs. PSTCLK starts clocking only when the first non-zero PST value (0xC, 0xD, or 0xF) occurs during system reset exception processing. Table 5-2 describes PST values. Chapter 7, "Phase-Locked Loop (PLL)," describes PSTCLK generation.
Debug Data (DDATA[3:0])	These output signals display the hardware 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 5-2 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 5-2 shows PSTCLK timing with respect to PST and DDATA.

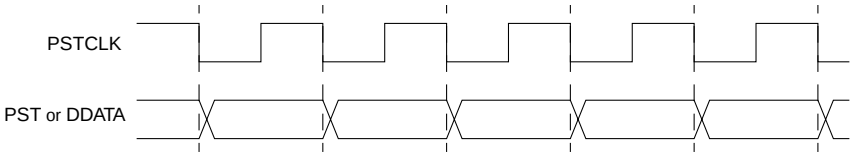


Figure 5-2. PSTCLK Timing

5.3 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 5.3.1, “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 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 5-2 shows the encoding of these signals.

Table 5-2. 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 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 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 5.3.1, "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 clock cycles. The value is driven onto the PST port one clock PSTCLK cycle before the data is displayed on DDATA. 0x8 Begin 1-byte transfer on DDATA. 0x9 Begin 2-byte transfer on DDATA. 0xA Begin 3-byte transfer on DDATA. 0xB Begin 4-byte transfer on DDATA.
0xC	1100	Exception processing. Exceptions that enter emulation mode (debug interrupt or optionally trace) generate a different encoding, as described below. Because the 0xC encoding defines a multiple-cycle mode, PST outputs are driven with 0xC until exception processing completes.
0xD	1101	Entry into emulator mode. Displayed during emulation mode (debug interrupt or optionally trace). Because this encoding defines a multiple-cycle mode, PSToutputs are driven with 0xD until exception processing completes.
0xE	1110	Processor is stopped. Appears in multiple-cycle format when the MCF5307 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 5.5.1, "CPU Halt")

5.3.1 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 MCF5307 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 5-3 shows when the PST and DDATA outputs that indicate when a JMP (A0) executed, assuming the CSR was programmed to display the lower 2 bytes of an address.

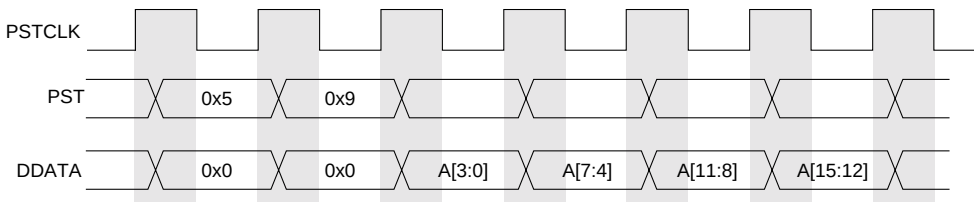


Figure 5-3. Example JMP Instruction Output on PST/DDATA

PST is driven with a 0x5 in the first cycle and 0x9 in the second. The 0x5 indicates a taken branch and the marker value 0x9 indicates a 2-byte address. Thus, the 4 subsequent DDATA nibbles 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.

5.4 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 nine registers to support the required functionality. These registers are also accessible from the processor's supervisor

programming model by executing the WDEBUG instruction. Thus, the breakpoint hardware in the debug module can be accessed 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 MCF5307 is using the WDEBUG instruction to access debug module registers or the resulting behavior is undefined.

These registers, shown in Figure 5-4, 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.

Figure 5-4. Debug Programming Model

These registers are accessed through the BDM port by new BDM commands, WDMREG and RDMREG, described in Section 5.5.3.3, "Command Set Descriptions." These commands contain a 5-bit field, DRc, that specifies the register, as shown in Table 5-3.

Table 5-3. BDM/Breakpoint Registers

DRc[4–0]	Register Name	Abbreviation	Initial State	Page
0x00	Configuration/status register	CSR	0x0010_0000	p. 5-10
0x01–0x04	Reserved	—	—	—
0x05	BDM address attribute register	BAAR	0x0000_0005	p. 5-9
0x06	Address attribute trigger register	AATR	0x0000_0005	p. 5-7
0x07	Trigger definition register	TDR	0x0000_0000	p. 5-14
0x08	Program counter breakpoint register	PBR	—	p. 5-13
0x09	Program counter breakpoint mask register	PBMR	—	p. 5-13
0x0A–0x0B	Reserved	—	—	—
0x0C	Address breakpoint high register	ABHR	—	p. 5-8
0x0D	Address breakpoint low register	ABLR	—	p. 5-8
0x0E	Data breakpoint register	DBR	—	p. 5-12
0x0F	Data breakpoint mask register	DBMR	—	p. 5-12

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.

5.4.1 Address Attribute Trigger Register (AATR)

The address attribute trigger register (AATR) defines address attributes and a mask to be matched in the trigger. The register value is compared with address attribute signals from the processor's local high-speed bus, as defined by the setting of the trigger definition register (TDR).

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	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															

Figure 5-5. Address Attribute Trigger Register (AATR)

Table 5-4 describes AATR fields.

Table 5-4. 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.
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).

5.4.2 Address Breakpoint Registers (ABLR, ABHR)

The address breakpoint low and high registers (ABLR, ABHR), Figure 5-6, 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. identically 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 5-6. Address Breakpoint Registers (ABLR, ABHR)

Table 5-5 describes ABLR fields.

Table 5-5. 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 5-6 describes ABHR fields.

Table 5-6. 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.

5.4.3 BDM Address Attribute Register (BAAR)

The BAAR defines the address space for memory-referencing BDM commands. See Figure 5-7. The reset value of 0x5 sets supervisor data as the default address space.

	7	6	5	4	3	2	1	0
Field	R	SZ		TT		TM		
Reset	0000_0101							
R/W	Write only. BAAR[R,SZ] are loaded directly from the BDM command; BAAR[TT,TM] can be programmed as debug control register 0x05 from the external development system. For compatibility with Rev. A, BAAR is loaded each time AATR is written.							
DRc[4–0]	0x05							

Figure 5-7. BDM Address Attribute Register (BAAR)

Table 5-7 describes BAAR fields.

Table 5-7. BAAR Field Descriptions

Bits	Name	Description
7	R	Read/write 0 Write 1 Read
6–5	SZ	Size 00 Longword 01 Byte 10 Word 11 Reserved
4–3	TT	Transfer type. See the TT definition in Table 5-4.
2–0	TM	Transfer modifier. See the TM definition in Table 5-4.

5.4.4 Configuration/Status Register (CSR)

The configuration/status register (CSR) defines the debug configuration for the processor and memory subsystem and contains status information from the breakpoint logic.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	BSTAT				FOF	TRG	HALT	BKPT	HRL				—	BKD	—	IPW
Reset	0000				0	0	0	0	0001				—	—	—	0
R/W[1]	R				R	R	R	R	R				—	—	—	R/W

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	MAP	TRC	EMU	DDC		UHE	BTB		—[2]	NPL	IPI	SSM	—			
Reset	0	0	0	00		0	00		0	0	—	0	—			
R/W	R/W	R/W	R/W	R/W		R/W	R/W		R	R/W	—	R/W	—			
DRc[4–0]	0x00															

1 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.

2 Bit 7 is reserved for Motorola use and must be written as a zero.

Figure 5-8. Configuration/Status Register (CSR)

Table 5-8 describes CSR fields.

Table 5-8. 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.
26	TRG	Hardware breakpoint trigger. If TRG is set, a hardware breakpoint halted the processor core and forced entry into BDM. Reset and the debug GO command clear TRG.
25	HALT	Processor halt. If HALT is set, the processor executed a HALT and forced entry into BDM. Reset and the debug GO command reset HALT.
24	BKPT	Breakpoint assert. If BKPT is set, $\overline{\text{BKPT}}$ was asserted, forcing the processor into BDM. Reset and the debug GO command clears this bit.
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) 0001 Revision B (this is the only valid value for the MCF5307)
19	—	Reserved, should be cleared.
18	BKD	Breakpoint disable. Used to disable the normal $\overline{\text{BKPT}}$ input functionality and to allow the assertion of $\overline{\text{BKPT}}$ to generate a debug interrupt. 0 Normal operation 1 $\overline{\text{BKPT}}$ is edge-sensitive: a high-to-low edge on $\overline{\text{BKPT}}$ signals a debug interrupt to the processor. The processor makes this interrupt request pending until the next sample point, when the exception is initiated. In the ColdFire architecture, the interrupt sample point occurs once per instruction. There is no support for nesting debug interrupts.
17	PCD	PSTCLK disable. Setting PCD disables generation of PSTCLK, PST and DDATA outputs and forces them to remain quiescent.
16	IPW	Inhibit processor writes. Setting IPW inhibits processor-initiated writes to the debug module's programming model registers. IPW can be modified only by commands from the external development system.
15	MAP	Force processor references in emulator mode. 0 All emulator-mode references are mapped into supervisor code and data spaces. 1 The processor maps all references while in emulator mode to a special address space, TT = 10, TM = 101 or 110.
14	TRC	Force emulation mode on trace exception. If TRC = 1, the processor enters emulator mode when a trace exception occurs.
13	EMU	Force emulation mode. If EMU = 1, the processor begins executing in emulator mode. See Section 5.6.1.1, “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 clock cycles). See Table 5-2. 00 No operand data is displayed. 01 Capture all write data. 10 Capture all read data. 11 Capture all read and write data.

Table 5-8. CSR Field Descriptions (Continued)

Bit	Name	Description
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.
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 5.3.1, “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. Instruction folding is disabled. Given an average execution latency of 1.6, throughput in non-pipeline mode would be 6.6, 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. An address or data breakpoint should always occur before the next instruction begins execution. Therefore the occurrence of the address/data breakpoints should be guaranteed.
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.

5.4.5 Data Breakpoint/Mask Registers (DBR, DBMR)

The data breakpoint registers, Figure 5-9, specify data patterns used as part of the trigger into debug mode. Only DBR bits not masked with a corresponding zero in DBMR are compared with the data from the processor’s local bus, 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 5-9. Data Breakpoint/Mask Registers (DBR and DBMR)

Table 5-9 describes DBR fields.

Table 5-9. 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 5-10 describes DBMR fields.

Table 5-10. 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 5-11 shows relationships between processor address, access size, and location within the 32-bit data bus.

Table 5-11. 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]

5.4.6 Program Counter Breakpoint/Mask Registers (PBR, PBMR)

The PC breakpoint register (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 clearing corresponding PBMR bits. Results are compared with the processor's program counter register, as defined in TDR. Figure 5-10 shows the PC breakpoint register.

	31 … 1	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 5.5.3.3, “Command Set Descriptions.”	
DRc[4–0]	0x08	

Figure 5-10. Program Counter Breakpoint Register (PBR)

Table 5-12 describes PBR fields.

Table 5-12. 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 5-11 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 5-11. Program Counter Breakpoint Mask Register (PBMR)

Table 5-13 describes PBMR fields.

Table 5-13. 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.

5.4.7 Trigger Definition Register (TDR)

The TDR, shown in Table 5-12, 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].

Section Table 5-14., "TDR Field Descriptions," describes how to handle multiple breakpoint conditions.

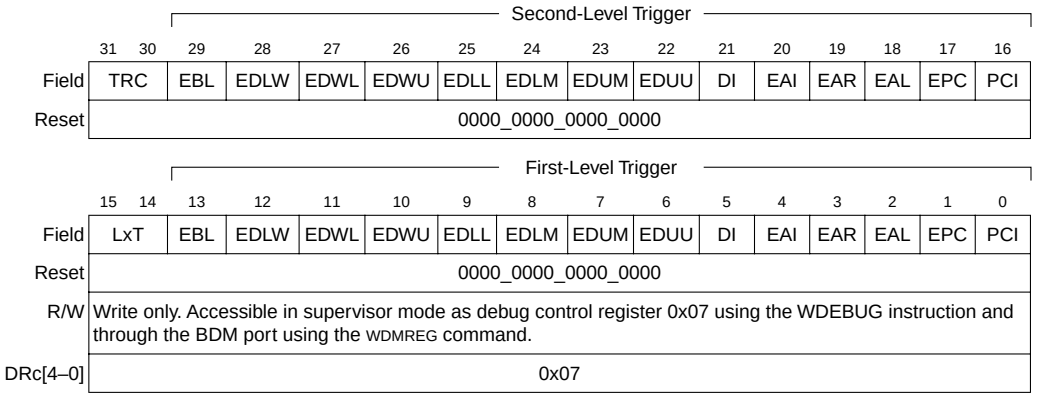


Figure 5-12. Trigger Definition Register (TDR)

Table 5-14 describes TDR fields.

Table 5-14. 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. 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.

Table 5-14. TDR Field Descriptions (Continued)

Bits	Name	Description	
28–22 12–6	EDx	Setting an EDx bit enables the corresponding data breakpoint condition based on the size and placement on the processor's local data bus. Clearing all EDx bits disables data breakpoints.	
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	EAx	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.	

5.5 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.

5.5.1 CPU Halt

Although many 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 5.6.1, “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.

5.5.2 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 DSCLK and

DSI must meet the required input setup and hold timings and the DSO is specified as a delay relative to the rising edge of the processor clock. See Table 5-1. 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 processor 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 5-13, all state transitions are enabled on a rising edge of the processor clock when DSCLK is high; that is, DSI is sampled and DSO is driven.

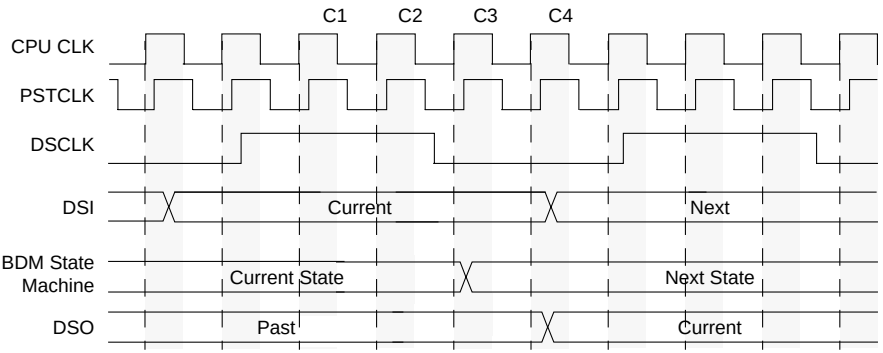


Figure 5-13. BDM Serial Interface Timing

DSCLK and DSI are synchronized inputs. DSCLK acts as a pseudo clock enable and is sampled on the rising edge of the processor CLK 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.

5.5.2.1 Receive Packet Format

The basic receive packet, Figure 5-14, consists of 16 data bits and 1 status bit.

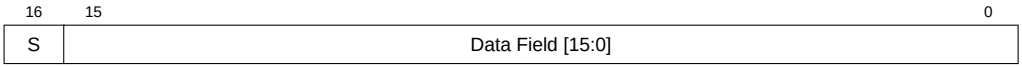


Figure 5-14. Receive BDM Packet

Table 5-15 describes receive BDM packet fields.

Table 5-15. 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 Data Message 0 xxxx Valid data transfer 0 0xFFFF Status OK 1 0x0000 Not ready with response; come again 1 0x0001 Error—Terminated bus cycle; data invalid 1 0xFFFF Illegal command
15–0	Data	Data. Contains the message to be sent from the debug module to the development system. The response message is always a single word, with the data field encoded as shown above.

5.5.2.2 Transmit Packet Format

The basic transmit packet, Figure 5-15, consists of 16 data bits and 1 control bit.

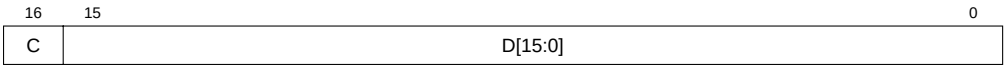


Figure 5-15. Transmit BDM Packet

Table 5-16 describes transmit BDM packet fields.

Table 5-16. 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	Data	Contains the data to be sent from the development system to the debug module.

5.5.3 BDM Command Set

Table 5-17 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 5-17. 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	5.5.3.3.1	0x218 {A/D, Reg[2:0]}
Write A/D register	WAREG/ WDREG	Write the data operand to the specified address or data register.	Halted	5.5.3.3.2	0x208 {A/D, Reg[2:0]}
Read memory location	READ	Read the data at the memory location specified by the longword address.	Steal	5.5.3.3.3	0x1900—byte 0x1940—word 0x1980—lword
Write memory location	WRITE	Write the operand data to the memory location specified by the longword address.	Steal	5.5.3.3.4	0x1800—byte 0x1840—word 0x1880—lword
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	5.5.3.3.5	0x1D00—byte 0x1D40—word 0x1D80—lword
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	5.5.3.3.6	0x1C00—byte 0x1C40—word 0x1C80—lword
Resume execution	GO	The pipeline is flushed and refilled before resuming instruction execution at the current PC.	Halted	5.5.3.3.7	0x0C00
No operation	NOP	Perform no operation; may be used as a null command.	Parallel	5.5.3.3.8	0x0000
Output the current PC	SYNC_PC	Capture the current PC and display it on the PST/DDATA output pins.	Parallel	5.5.3.3.9	0x0001
Read control register	RCREG	Read the system control register.	Halted	5.5.3.3.10	0x2980
Write control register	WCREG	Write the operand data to the system control register.	Halted	5.5.3.3.11	0x2880
Read debug module register	RDMREG	Read the debug module register.	Parallel	5.5.3.3.12	0x2D {0x4[2] DRc[4:0]}
Write debug module register	WDMREG	Write the operand data to the debug module register.	Parallel	5.5.3.3.13	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 Motorola. All unused command formats within any revision level perform a NOP and return the illegal command response.

5.5.3.1 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 5-16.

15 10	9	8	7 6	5	4	3	2 0
Operation	0	R/W	Op Size	0	0	A/D	Register
Extension Word(s)							

Figure 5-16. BDM Command Format

Table 5-18 describes BDM fields.

Table 5-18. BDM Field Descriptions

Bit	Name	Description
15–10	Operation	Specifies the command. These values are listed in Table 5-17.
9	0	Reserved
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	Operand 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
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.

5.5.3.1.1 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.

5.5.3.2 Command Sequence Diagrams

The command sequence diagram in Figure 5-17 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.

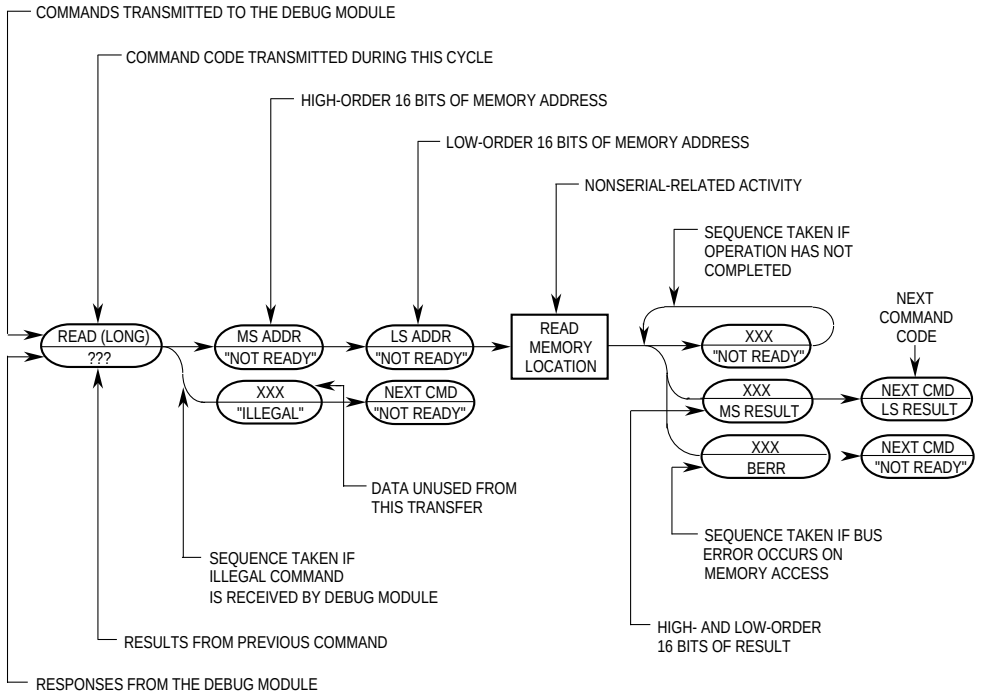


Figure 5-17. Command Sequence Diagram

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.

5.5.3.3 Command Set Descriptions

The following sections describe the commands summarized in Table 5-17.

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 5.5.2, "BDM Serial Interface," describes the receive packet format.

Motorola reserves unassigned command opcodes for future expansion. Unused command formats in any revision level perform a NOP and return an illegal command response.

5.5.3.3.1 Read A/D Register (RAREG/RDREG)

Read the selected address or data register and return the 32-bit result. A bus error response is returned if the CPU core is not halted.

Command/Result Formats:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Command	0x2				0x1				0x8				A/D	Register		
Result	D[31:16]															
	D[15:0]															

Figure 5-18. RAREG/RDREG Command Format

Command Sequence:

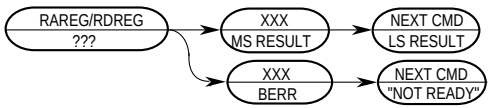


Figure 5-19. RAREG/RDREG Command Sequence

Operand Data: None

Result Data: The contents of the selected register are returned as a longword value, most-significant word first.

5.5.3.3.2 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:

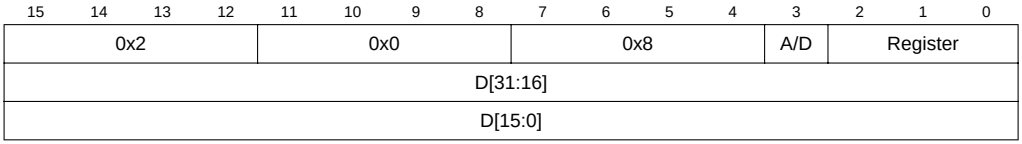


Figure 5-20. WAREG/WDREG Command Format

Command Sequence

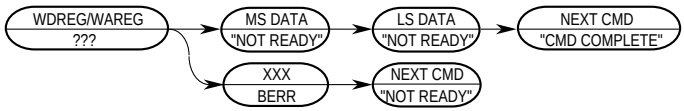


Figure 5-21. WAREG/WDREG Command Sequence

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.

5.5.3.3.3 Read Memory Location (READ)

Read data at the longword address. Address space is defined by BAAR[TT,TM]. Hardware forces low-order address bits to zeros for word and longword accesses to ensure that word addresses are word-aligned and longword addresses are longword-aligned.

Command/Result Formats:

		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte	Command	0x1				0x9				0x0				0x0			
		A[31:16]															
		A[15:0]															
	Result	X	X	X	X	X	X	X	X	D[7:0]							
Word	Command	0x1				0x9				0x4				0x0			
		A[31:16]															
		A[15:0]															
	Result	D[15:0]															
Longword	Command	0x1				0x9				0x8				0x0			
		A[31:16]															
		A[15:0]															
	Result	D[31:16]															
		D[15:0]															

Figure 5-22. READ Command/Result Formats

Command Sequence:

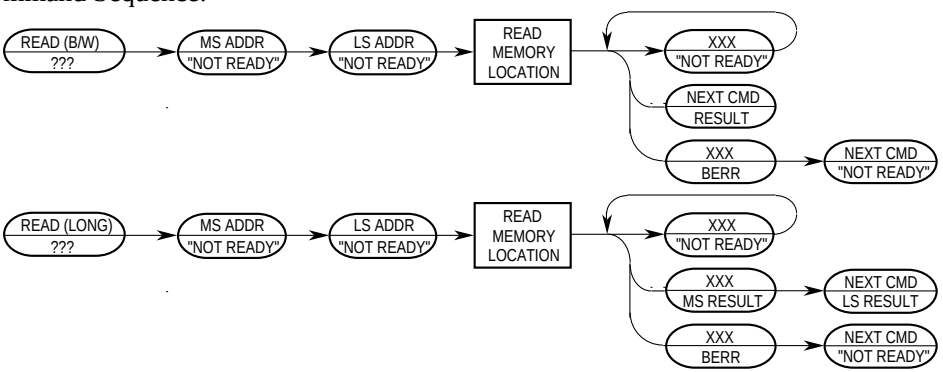


Figure 5-23. READ Command Sequence

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.

5.5.3.3.4 Write Memory Location (WRITE)

Write data to the memory location specified by the longword address. The address space is defined by BAAR[TT,TM]. Hardware forces low-order address bits to zeros for word and longword accesses to ensure that word addresses are word-aligned and longword addresses are longword-aligned.

Command Formats:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
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 5-24. WRITE Command Format

Command Sequence:

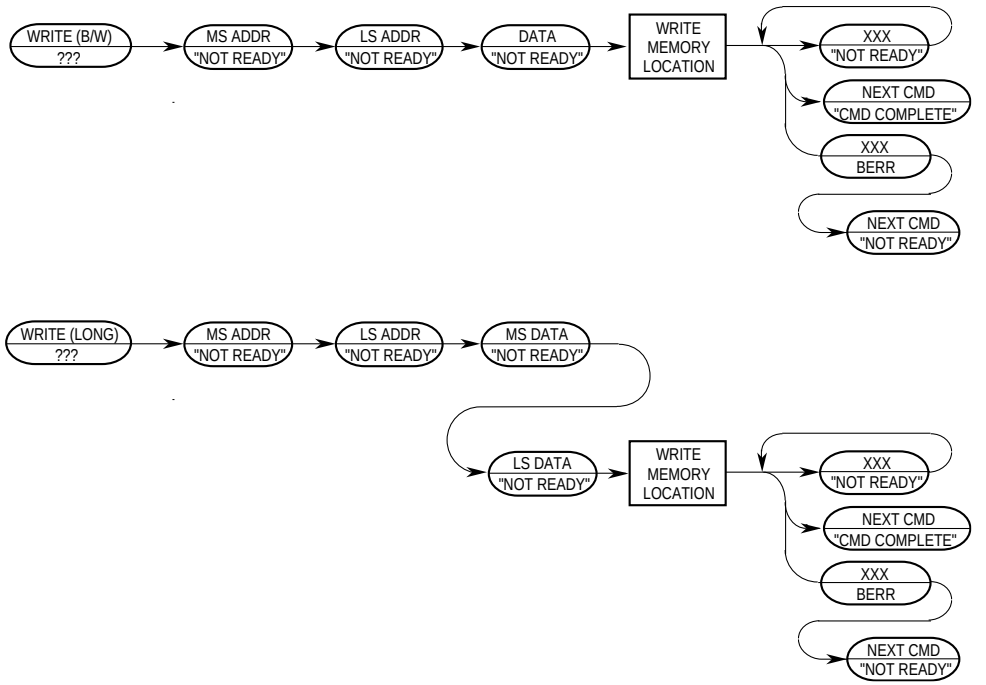


Figure 5-25. WRITE Command Sequence

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.

5.5.3.3.5 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	14	13	12	11	10	9	8	7	6	5	4	3	2	1	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 5-26. DUMP Command/Result Formats

Command Sequence:

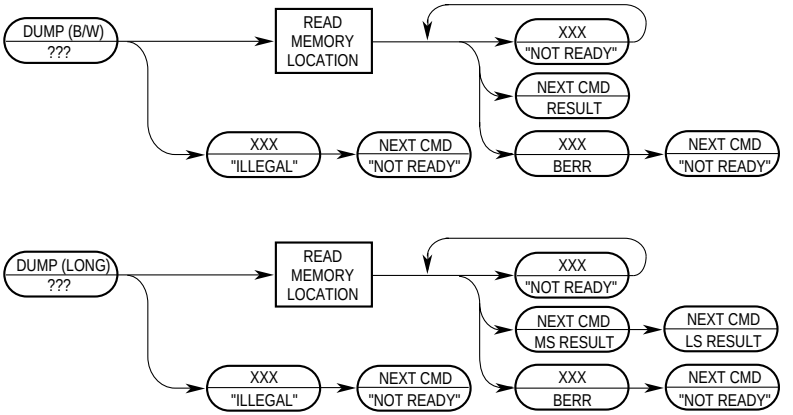


Figure 5-27. DUMP Command Sequence

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.

5.5.3.3.6 Fill Memory Block (FILL)

A FILL command is used with the WRITE command to access large blocks of memory. An initial WRITE is executed to set up the starting address of the block and to supply the first operand. The FILL command writes subsequent operands. The initial address is incremented by the operand size (1, 2, or 4) and saved in a temporary register after the memory write. Subsequent FILL commands use this address, perform the write, increment it by the current operand size, and store the updated address in the temporary register.

If an initial WRITE is not executed preceding the first FILL command, the illegal command response is returned.

NOTE:

The FILL command does not check for a valid address—FILL is a valid command only when preceded by another FILL, a NOP, or a WRITE command. Otherwise, an illegal command response is returned. The NOP command can be used for intercommand padding without corrupting the address pointer.

The size field is examined each time a FILL command is processed, allowing the operand size to be altered dynamically.

Command Formats:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte	0x1				0xC				0x0				0x0			
	X	X	X	X	X	X	X	X	D[7:0]							
Word	0x1				0xC				0x4				0x0			
	D[15:0]															
Longword	0x1				0xC				0x8				0x0			
	D[31:16]															
	D[15:0]															

Figure 5-28. FILL Command Format

Command Sequence:

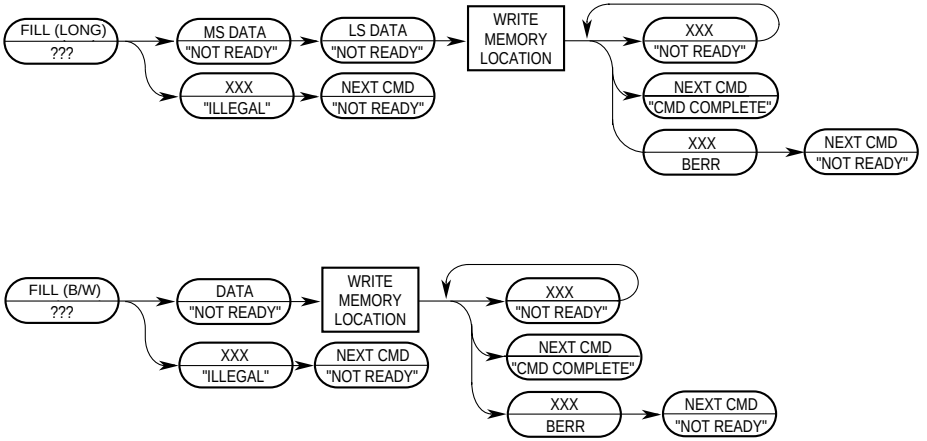


Figure 5-29. FILL Command Sequence

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.

5.5.3.3.7 Resume Execution (GO)

The pipeline is flushed and refilled before normal instruction execution resumes. Prefetching begins at the current address in the PC and at the current privilege level. If any register (such as the PC or SR) is altered by a BDM command while the processor is halted, the updated value is used when prefetching resumes. If a GO command is issued and the CPU is not halted, the command is ignored.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0				0xC				0x0				0x0			

Figure 5-30. GO Command Format

Command Sequence:

Figure 5-31. GO Command Sequence

Operand Data: None

Result Data: The command-complete response (0xFFFF) is returned during the next shift operation.

5.5.3.3.8 No Operation (NOP)

NOP performs no operation and may be used as a null command where required.

Command Formats:

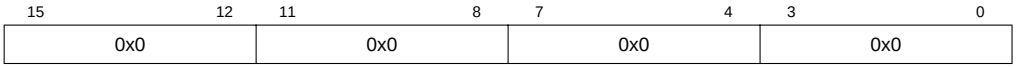


Figure 5-32. NOP Command Format

Command Sequence:

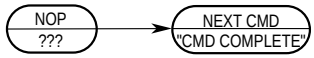


Figure 5-33. NOP Command Sequence

Operand Data: None

Result Data: The command-complete response, 0xFFFF (with S cleared), is returned during the next shift operation.

5.5.3.3.9 Synchronize PC to the PST/DDATA Lines (SYNC_PC)

The SYNC_PC command captures the current PC and displays it on the PST/DDATA outputs. After the debug module receives the command, it sends a signal to the ColdFire processor that the current PC must be displayed. The processor then forces an instruction fetch at the next PC with the address being captured in the DDATA logic under control of CSR[BTB]. The specific sequence of PST and DDATA values is as follows:

1. Debug signals a SYNC_PC command is pending.
2. CPU completes the current instruction.
3. CPU forces an instruction fetch to the next PC, generates a PST = 0x5 value indicating a taken branch and signals the capture of DDATA.
4. The instruction address corresponding to the PC is captured.
5. The PST marker (0x9–0xB) is generated and displayed as defined by CSR[BTB] followed by the captured PC address.

The SYNC_PC command can be used to dynamically access the PC for performance monitoring. The execution of this command is considerably less obtrusive to the real-time operation of an application than a HALT-CPU/READ-PC/RESUME command sequence.

Command Formats:

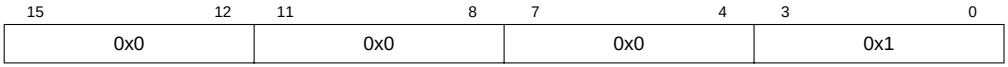


Figure 5-34. SYNC_PC Command Format

Command Sequence:

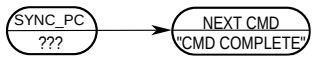


Figure 5-35. SYNC_PC Command Sequence

Operand Data: None

Result Data: Command complete status (0xFFFF) is returned when the register write is complete.

5.5.3.3.10 Read Control Register (RCREG)

Read the selected control register and return the 32-bit result. Accesses to the processor/memory control registers are always 32 bits wide, regardless of register width. The second and third words of the command form a 32-bit address, which the debug module uses to generate a special bus cycle to access the specified control register. The 12-bit Rc field is the same as that used by the MOVEC instruction.

Command/Result Formats:

	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Command	0x2	0x9	0x8	0x0
	0x0	0x0	0x0	0x0
	0x0	Rc		
Result	D[31:16]			
	D[15:0]			

Figure 5-36. RCREG Command/Result Formats

Rc encoding:

Table 5-19. Control Register Map

Rc	Register Definition
0x002	Cache control register (CACR)
0x004	Access control register 0 (ACR0)
0x005	Access control register 1 (ACR1)
0x801	Vector base register (VBR)
0x804	MAC status register (MACSR)[1]

Rc	Register Definition
0x805	MAC mask register (MASK)[1]
0x806	MAC accumulator (ACC)[1]
0x80E	Status register (SR)
0x80F	Program register (PC)
0xC04	RAM base address register (RAMBAR)

1 Available if the optional MAC unit is present.

Command Sequence:

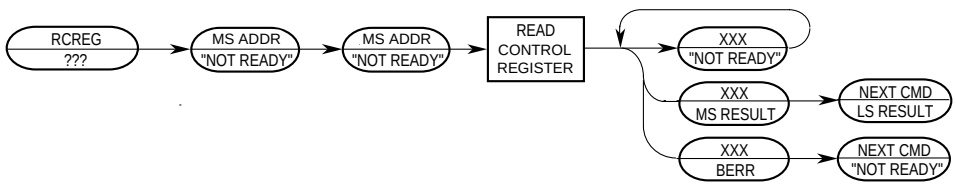


Figure 5-37. RCREG Command Sequence

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.

5.5.3.3.11 Write Control Register (WCREG)

The operand (longword) data is written to the specified control register. The write alters all 32 register bits.

Command/Result Formats:

	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Command	0x2	0x8	0x8	0x0
	0x0	0x0	0x0	0x0
	0x0	Rc		
Result	D[31:16]			
	D[15:0]			

Figure 5-38. WCREG Command/Result Formats

Command Sequence:

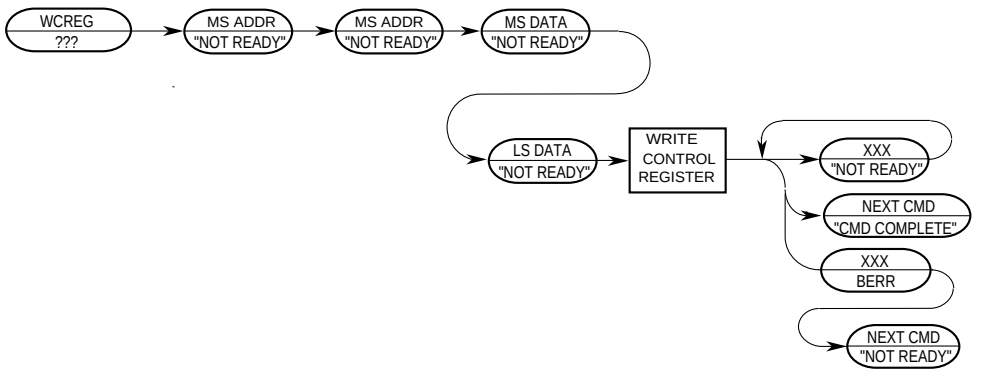


Figure 5-39. WCREG Command Sequence

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.

5.5.3.3.12 Read Debug Module Register (RDMREG)

Read the selected debug module register and return the 32-bit result. The only valid register selection for the RDMREG command is CSR (DRc = 0x00). Note that this read of the CSR clears 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:

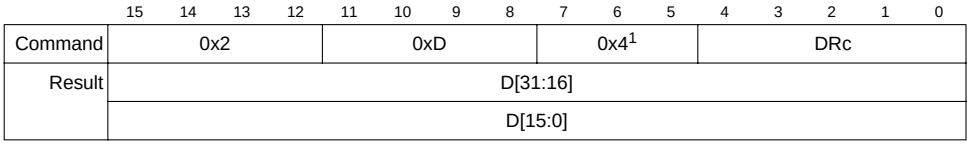


Figure 5-40. RDMREG BDM Command/Result Formats

[1] Note 0x4 is a 3-bit field

Table 5-20 shows the definition of DRc encoding.

Table 5-20. Definition of DRc Encoding—Read

DRc[4:0]	Debug Register Definition	Mnemonic	Initial State	Page
0x00	Configuration/Status	CSR	0x0	p. 5-10
0x01–0x1F	Reserved	—	—	—

Command Sequence:

Figure 5-41. 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.

5.5.3.3.13 Write Debug Module Register (WDMREG)

The operand (longword) data is written to the specified debug module register. All 32 bits of the register are altered by the write. DSCLK must be inactive while the debug module register writes from the CPU accesses are performed using the WDEBUG instruction.

Command Format:

Figure 5-42. WDMREG BDM Command Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2				0xC				0x4[1]			DRc				
D[31:16]															
D[15:0]															

1 Note: 0x4 is a three-bit field

Table 5-3 shows the definition of the DRc write encoding.

Command Sequence:

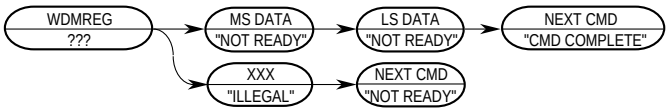


Figure 5-43. WDMREG Command Sequence

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.

5.6 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.

5.6.1 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 5-21, 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 5-21. 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 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.

The generation of another debug interrupt during the first instruction after the RTE exits emulator mode is inhibited. This behavior is consistent with the existing logic involving trace mode where the first instruction executes before another trace exception is generated. Thus, all hardware breakpoints are disabled until the first instruction after the RTE completes execution, regardless of the programmed trigger response.

5.6.1.1 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 5.5.1, “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).

5.6.2 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

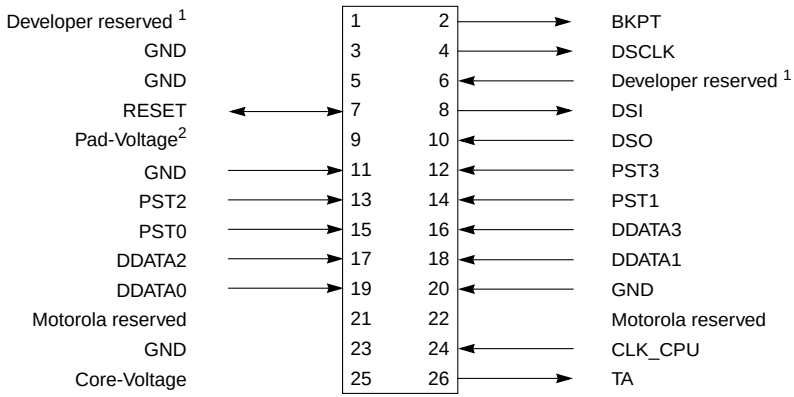
For BDM commands that access memory, the debug module requests the processor's local bus. The processor responds by stalling the instruction fetch pipeline and waiting for current bus activity to complete before freeing the local bus for the debug module to perform its access. After the debug module 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).

5.7 Motorola-Recommended BDM Pinout

The ColdFire BDM connector, Figure 5-44, is a 26-pin Berg connector arranged 2 x 13.



[1]Pins reserved for BDM developer use.
[2]Supplied by target

Figure 5-44. Recommended BDM Connector

5.8 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). Additionally, for certain change-of-flow branch instructions, another CSR field provides the capability to display {0x2, 0x3, 0x4} bytes of the target instruction address. For both situations, an optional PST value {0x8, 0x9, 0xB} provides the marker identifying the size and presence of valid data on the DDATA output.

5.8.1 User Instruction Set

Table 5-22 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 5-22. 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
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
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}

Table 5-22. PST/DDATA Specification for User-Mode Instructions (Continued)

Instruction	Operand Syntax	PST/DDATA
cmp.l	<ea>y,Rx	PST = 0x1, {PST = 0xB, DD = source operand}
cmpi.l	#imm,Dx	PST = 0x1
divs.l	<ea>y,Dx	PST = 0x1, {PST = 0xB, DD = source operand}
divs.w	<ea>y,Dx	PST = 0x1, {PST = 0x9, DD = source operand}
divu.l	<ea>y,Dx	PST = 0x1, {PST = 0xB, DD = source operand}
divu.w	<ea>y,Dx	PST = 0x1, {PST = 0x9, DD = source operand}
eor.l	Dy,<ea>x	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}
eori.l	#imm,Dx	PST = 0x1
ext.l	Dx	PST = 0x1
ext.w	Dx	PST = 0x1
extb.l	Dx	PST = 0x1
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
mac.l		PST = 0x1
mac.l	Ry,Rx	PST = 0x1
mac.l	Ry,Rx,ea,Rw	PST = 0x1, {PST = 0xB, DD = source operand}
mac.w		PST = 0x1
mac.w	Ry,Rx	PST = 0x1
mac.w	Ry,Rx,ea,Rw	PST = 0x1, {PST = 0xB, DD = source operand}
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.l	<ea>y,ACC	PST = 0x1
move.l	<ea>y,MACSR	PST = 0x1
move.l	<ea>y,MASK	PST = 0x1
move.l	ACC,Rx	PST = 0x1
move.l	MACSR,CCR	PST = 0x1
move.l	MACSR,Rx	PST = 0x1
move.l	MASK,Rx	PST = 0x1
move.w	<ea>y,<ea>x	PST = 0x1, {PST = 0x9, DD = source}, {PST = 0x9, DD = destination}
move.w	CCR,Dx	PST = 0x1
move.w	{Dy,#imm},CCR	PST = 0x1

Table 5-22. PST/DDATA Specification for User-Mode Instructions (Continued)

Instruction	Operand Syntax	PST/DDATA
movem.l	#list,<ea>x	PST = 0x1, {PST = 0xB, DD = destination},... [2]
movem.l	<ea>y,#list	PST = 0x1, {PST = 0xB, DD = source},... [2]
moveq	#imm,Dx	PST = 0x1
msac.l	Ry,Rx	PST = 0x1
msac.l	Ry,Rx,ea,Rw	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}
msac.w	Ry,Rx	PST = 0x1
msac.w	Ry,Rx,ea,Rw	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}
muls.l	<ea>y,Dx	PST = 0x1, {PST = 0xB, DD = source operand}
muls.w	<ea>y,Dx	PST = 0x1, {PST = 0x9, DD = source operand}
mulu.l	<ea>y,Dx	PST = 0x1, {PST = 0xB, DD = source operand}
mulu.w	<ea>y,Dx	PST = 0x1, {PST = 0x9, DD = source operand}
neg.l	Dx	PST = 0x1
negx.l	Dx	PST = 0x1
nop		PST = 0x1
not.l	Dx	PST = 0x1
or.l	<ea>y,Dx	PST = 0x1, {PST = 0xB, DD = source operand}
or.l	Dy,<ea>x	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}
ori.l	#imm,Dx	PST = 0x1
pea	<ea>y	PST = 0x1, {PST = 0xB, DD = destination operand}
pulse		PST = 0x4
rems.l	<ea>y,Dx:Dw	PST = 0x1, {PST = 0xB, DD = source operand}
remu.l	<ea>y,Dx:Dw	PST = 0x1, {PST = 0xB, DD = source operand}
rts		PST = 0x1, {PST = 0xB, DD = source operand}, PST = 0x5, {PST = [0x9AB], DD = target address}
scc	Dx	PST = 0x1
sub.l	<ea>y,Rx	PST = 0x1, {PST = 0xB, DD = source operand}
sub.l	Dy,<ea>x	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}
subi.l	#imm,Dx	PST = 0x1
subq.l	#imm,<ea>x	PST = 0x1, {PST = 0xB, DD = source}, {PST = 0xB, DD = destination}
subx.l	Dy,Dx	PST = 0x1
swap	Dx	PST = 0x1
trap	#imm	PST = 0x1 [3]
trapf		PST = 0x1
tst.b	<ea>x	PST = 0x1, {PST = 0x8, DD = source operand}
tst.l	<ea>x	PST = 0x1, {PST = 0xB, DD = source operand}
tst.w	<ea>x	PST = 0x1, {PST = 0x9, DD = source operand}

Table 5-22. PST/DDATA Specification for User-Mode Instructions (Continued)

Instruction	Operand Syntax	PST/DDATA
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 Processing  PST = 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}// PC of handler
```

The PST/DDATA specification for the reset exception is shown below:

```
Exception Processing PST = 0xC,
                     PST = 0x5,{PST = [0x9AB], DD = target}// PC of handler
```

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).

5.8.2 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 5-23.

Table 5-23. PST/DDATA Specification for Supervisor-Mode Instructions

Instruction	Operand Syntax	PSTDDATA
cpushl		PST = 0x1
halt		PST = 0x1, PST = 0xF
move.w	SR,Dx	PST = 0x1
move.w	{Dy,#imm},SR	PST = 0x1, {PST = 3}

Table 5-23. PST/DDATA Specification for Supervisor-Mode Instructions

Instruction	Operand Syntax	PSTDDATA
movec	Ry,Rc	PST = 0x1
rte		PST = 0x7, {PST = 0xB, DD = source operand}, {PST = 3},{ PST =0xB, DD =source operand}, PST = 0x5, {[PST = 0x9AB], DD = target address}
stop	#imm	PST = 0x1, PST = 0xE
wdebug	<ea>y	PST = 0x1, {PST = 0xB, DD = source, PST = 0xB, DD = source}

The move-to-SR 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.

Part II
System Integration Module (SIM)

Intended Audience

Part II is intended for users who need to understand the interface between the ColdFire core processor complex, described in Part I, and internal peripheral devices, described in Part III. It includes a general description of the SIM and individual chapters that describe components of the SIM, such as the phase-lock loop (PLL) timing source, interrupt controller for both on-chip and external peripherals, configuration and operation of chip selects, and the SDRAM controller.

Contents

Part II contains the following chapters:

- Chapter 6, “SIM Overview,” describes the SIM programming model, bus arbitration, and system-protection functions for the MCF5307.
- Chapter 7, “Phase-Locked Loop (PLL),” describes configuration and operation of the PLL module. It describes in detail the registers and signals that support the PLL implementation.
- Chapter 8, “I2C Module,” describes the MCF5307 I2C module, including I2C protocol, clock synchronization, and the registers in the I2C programing model. It also provides extensive programming examples.
- Chapter 9, “Interrupt Controller,” describes operation of the interrupt controller portion of the SIM. Includes descriptions of the registers in the interrupt controller memory map and the interrupt priority scheme.
- Chapter 10, “Chip-Select Module,” describes the MCF5307 chip-select implementation, including the operation and programming model, which includes the chip-select address, mask, and control registers.
- Chapter 11, “Synchronous/Asynchronous DRAM Controller Module,” describes configuration and operation of the synchronous/asynchronous DRAM controller component of the SIM. It begins with a general description and brief glossary, and

includes a description of signals involved in DRAM operations. The remainder of the chapter is divided between descriptions of asynchronous and synchronous operations.

Suggested Reading

The following literature may be helpful with respect to the topics in Part II:

- *The I^2C Bus Specification, Version 2.1* (January 2000)

Acronyms and Abbreviations

Table II-i contains acronyms and abbreviations are used in Part II.

Table II-i. Acronyms and Abbreviated Terms

Term	Meaning
ADC	Analog-to-digital conversion
BDM	Background debug mode
CODEC	Code/decode
DAC	Digital-to-analog conversion
DMA	Direct memory access
DSP	Digital signal processing
EDO	Extended data output (DRAM)
FIFO	First-in, first-out
GPIO	
I^2C	Inter-integrated circuit
IEEE	Institute for Electrical and Electronics Engineers
IPL	Interrupt priority level
JEDEC	Joint Electron Device Engineering Council
LIFO	Last-in, first-out
LRU	Least recently used
LSB	Least-significant byte
lsb	Least-significant bit
MBAR	Memory base address register
MSB	Most-significant byte
msb	Most-significant bit
Mux	Multiplex

Table II-i. Acronyms and Abbreviated Terms (Continued)

Term	Meaning
NOP	No operation
PCLK	Processor clock
PLL	Phase-locked loop
POR	Power-on reset
Rx	Receive
SIM	System integration module
SOF	Start of frame
TAP	Test access port
TTL	Transistor-to-transistor logic
Tx	Transmit
UART	Universal asynchronous/synchronous receiver transmitter

Chapter 6
SIM Overview

This chapter provides detailed operation information regarding the system integration module (SIM). It describes the SIM programming model, bus arbitration, and system-protection functions for the MCF5307.

6.1 Features

The SIM, shown in Figure 6-1, provides overall control of the bus and serves as the interface between the ColdFire core processor complex and the internal peripheral devices.

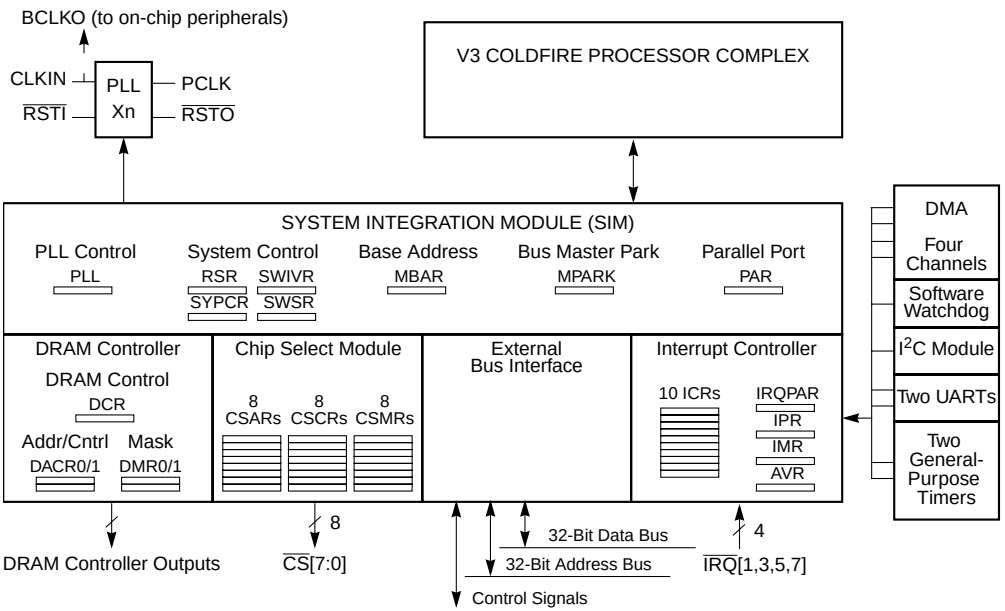


Figure 6-1. SIM Block Diagram

The following is a list of the key SIM features:

- Module base address register (MBAR)
  - — Base address location of all internal peripherals and SIM resources
  - — Address space masking to internal peripherals and SIM resources
- Phase-locked loop (PLL) clock control register (PLLCR) for CPU STOP instruction
  - — Control for turning off clocks to core and interrupt levels that turn clocks back on

  Chapter 7, “Phase-Locked Loop (PLL).”
- Interrupt controller
  - — Programmable interrupt level (1–7) for internal peripheral interrupts
  - — Programmable priority level (0–3) within each interrupt level
  - — Four external interrupts; one set to interrupt level 7; three others programmable to two interrupt levels

  See Chapter 9, “Interrupt Controller.”
- Chip select module
  - — Eight independent, user-programmable chip-select signals ($\overline{CS}$[7:0]) that can interface with SRAM, PROM, EPROM, EEPROM, Flash, and peripherals
  - — Address masking for 64-Kbyte to 4-Gbyte memory block sizes
  - — Programmable wait states and port sizes
  - — External master access to chip selects

  See Chapter 10, “Chip-Select Module.”
- System protection and reset status
  - — Reset status indicating the cause of last reset
  - — Software watchdog timer with programmable secondary bus monitor

  See Section 6.2.4, “Software Watchdog Timer.”
- Pin assignment register (PAR) configures the parallel port. See Section 6.2.9, “Pin Assignment Register (PAR).”
- Bus arbitration
  - — Default bus master park register (MPARK) controls internal and external bus arbitration and enables display of internal accesses on the external bus for debugging
  - — Supports several arbitration algorithms

  See Section 6.2.10, “Bus Arbitration Control.”

6.2 Programming Model

The following sections describe the registers incorporated into the SIM.

6.2.1 SIM Register Memory Map

Table 6-1 shows the memory map for the SIM registers. The internal registers in the SIM are memory-mapped registers offset from the MBAR address pointer defined in MBAR[BA]. This supervisor-level register is described in Section 6.2.2, “Module Base Address Register (MBAR).” Because SIM registers depend on the base address defined in MBAR[BA], MBAR must be programmed before SIM registers can be accessed.

NOTE:

Although external masters cannot access the MCF5307’s on-chip memories or MBAR, they can access any of the SIM memory map and peripheral registers, such as those belonging to the interrupt controller, chip-select module, UARTs, timers, DMA, and I^2C.

Table 6-1. SIM Registers

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x000	Reset status register (RSR) [p. 6-5]	System protection control register (SYPCR) [p. 6-8]	Software watchdog interrupt vector register (SWIVR) [p. 6-9]	Software watchdog service register (SWSR) [p. 6-9]
0x004	Pin assignment register (PAR) [p. 6-10]		Interrupt port assignment register (IRQPAR) [p. 9-7]	Reserved
0x008	PLL control (PLLCR) [p. 7-3]	Reserved		
0x00C	Default bus master park register (MPARK) [p. 6-11]	Reserved		
0x010–0x03C	Reserved			
Interrupt Controller Registers [p. 9-2]				
0x040	Interrupt pending register (IPR) [p. 9-6]			
0x044	Interrupt mask register (IMR) [p. 9-6]			
0x048	Reserved			Autovector register (AVR) [p. 9-5]
Interrupt Control Registers (ICRs) [p. 9-3]				

Table 6-1. SIM Registers (Continued)

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x04C	Software watchdog timer (ICR0) [p. 9-3]	Timer0 (ICR1) [p. 9-3]	Timer1 (ICR2) [p. 9-3]	I^2C (ICR3) [p. 9-3]
0x050	UART0 (ICR4) [p. 9-3]	UART1 (ICR5) [p. 9-3]	DMA0 (ICR6) [p. 9-3]	DMA1 (ICR7) [p. 9-3]
0x054	DMA2 (ICR8) [p. 9-3]	DMA3 (ICR9) [p. 9-3]	Reserved	

6.2.2 Module Base Address Register (MBAR)

The supervisor-level MBAR, Figure 6-2, specifies the base address and allowable access types for all internal peripherals. It is written with a MOVEC instruction using the CPU address 0xC0F. (See the *ColdFire Family Programmer's Reference Manual*.) MBAR can be read or written through the debug module as a read/write register, as described in Chapter 5, "Debug Support." Only the debug module can read MBAR.

The valid bit, MBAR[V], is cleared at system reset to prevent incorrect references before MBAR is written; other MBAR bits are uninitialized at reset. To access internal peripherals, write MBAR with the appropriate base address (BA) and set MBAR[V] after system reset.

All internal peripheral registers occupy a single relocatable memory block along 4-Kbyte boundaries. If MBAR[V] is set, MBAR[BA] is compared to the upper 20 bits of the full 32-bit internal address to determine if an internal peripheral is being accessed. MBAR masks specific address spaces using the address space fields. Attempts to access a masked address space generate an external bus access.

Addresses hitting overlapping memory spaces take the following priority:

1. MBAR
2. SRAM and caches
3. Chip select

NOTE:

The MBAR region must be mapped to non-cacheable space.

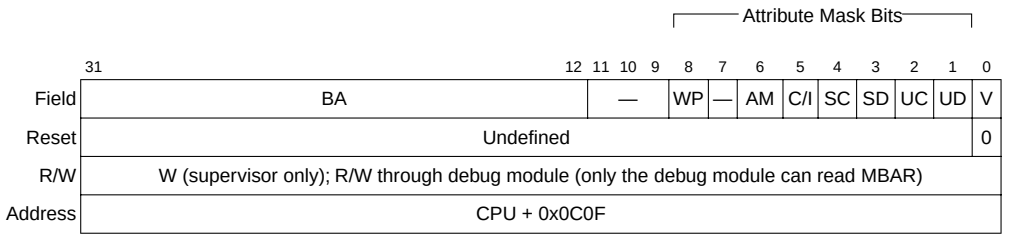


Figure 6-2. Module Base Address Register (MBAR)

Table 6-2 describes MBAR fields.

Table 6-2. MBAR Field Descriptions

Bits	Field	Description
31–12	BA	Base address. Defines the base address for a 4-Kbyte address range.
11–9	—	Reserved, should be cleared.
8	WP	Write protect. Mask bit for write cycles in the MBAR-mapped register address range. 0 Module address range is read/write. 1 Module address range is read only.
7	—	Reserved, should be cleared.
6	AM	Alternate master mask. When AM = 0 and an alternate master (external master or DMA) accesses MBAR-mapped registers, MBAR[SC,SD,UC,UD] are ignored in address decoding. These fields mask address space, placing the MBAR-mapped register in a specific address space or spaces.
5	C/I	Mask CPU space and interrupt acknowledge cycles. 0 Activates the corresponding MBAR-mapped register 1 Regular external bus access
4	SC	Setting masks supervisor code space in MBAR address range
3	SD	Setting masks supervisor data space in MBAR address range
2	UC	Setting masks user code space in MBAR address range
1	UD	Setting masks user data space in MBAR address range
0	V	Valid. Determines whether MBAR settings are valid. 0 MBAR contents are invalid. 1 MBAR contents are valid.

The following example shows how to set the MBAR to location 0x1000_0000 using the D0 register. Setting MBAR[V] validates the MBAR location. This example assumes all accesses are valid:

```
move.1 #0x10000001,DO
movec DO,MBAR
```

6.2.3 Reset Status Register (RSR)

The reset status register (RSR), Figure 6-3, contains two status bits, HRST and SWTR. Reset control logic sets one of the bits depending on whether the last reset was caused by an external device asserting $\overline{\text{RSTI}}$ (HRST = 1) or by the software watchdog timer (SWTR = 1). Only one RSR bit can be set at any time. If a reset occurs, reset control logic sets only the bit that indicates the cause of reset.

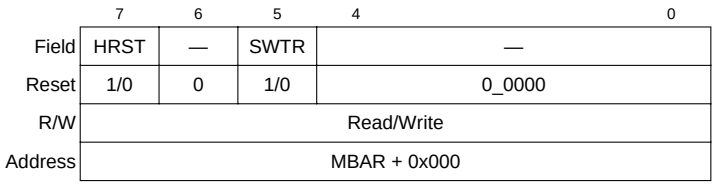


Figure 6-3. Reset Status Register (RSR)

Table 6-3 describes RSR fields.

Table 6-3. RSR Field Descriptions

Bits	Name	Description
7	HRST	Hardware or system reset 1 An external device driving $\overline{\text{RSTI}}$ caused the last reset. Assertion of reset by an external device causes the core processor to take a reset exception. All registers in internal peripherals and the SIM are reset.
6	—	Reserved, should be cleared.
5	SWTR	Software watchdog timer reset 1 The last reset was caused by the software watchdog timer. If SYPCR[SWRI] = 1 and the software watchdog timer times out, a hardware reset occurs.
4–0	—	Reserved, should be cleared.

6.2.4 Software Watchdog Timer

The software watchdog timer prevents system lockup should the software become trapped in loops with no controlled exit. The software watchdog timer can be enabled or disabled through SYPCR[SWE]. If enabled, the watchdog timer requires the periodic execution of a software watchdog servicing sequence. If this periodic servicing action does not occur, the timer times out, resulting in a watchdog timer $\overline{\text{IRQ}}$ or hardware reset with $\overline{\text{RSTO}}$ driven low, as programmed by SYPCR[SWRI].

If the timer times out and the software watchdog transfer acknowledge enable bit (SYPCR[SWTA]) is set, a watchdog timer $\overline{\text{IRQ}}$ is asserted. Note that the software watchdog timer IACK cycle cannot be autovectored.

If a software watchdog timer IACK cycle has not occurred after another timeout, SWT $\overline{\text{TA}}$ is asserted in an attempt to terminate the bus cycle and allow the IACK cycle to proceed. The setting of SYPCR[SWTAVAL] indicates that the watchdog timer $\overline{\text{TA}}$ was asserted. Figure 6-4 shows termination of a locked bus.

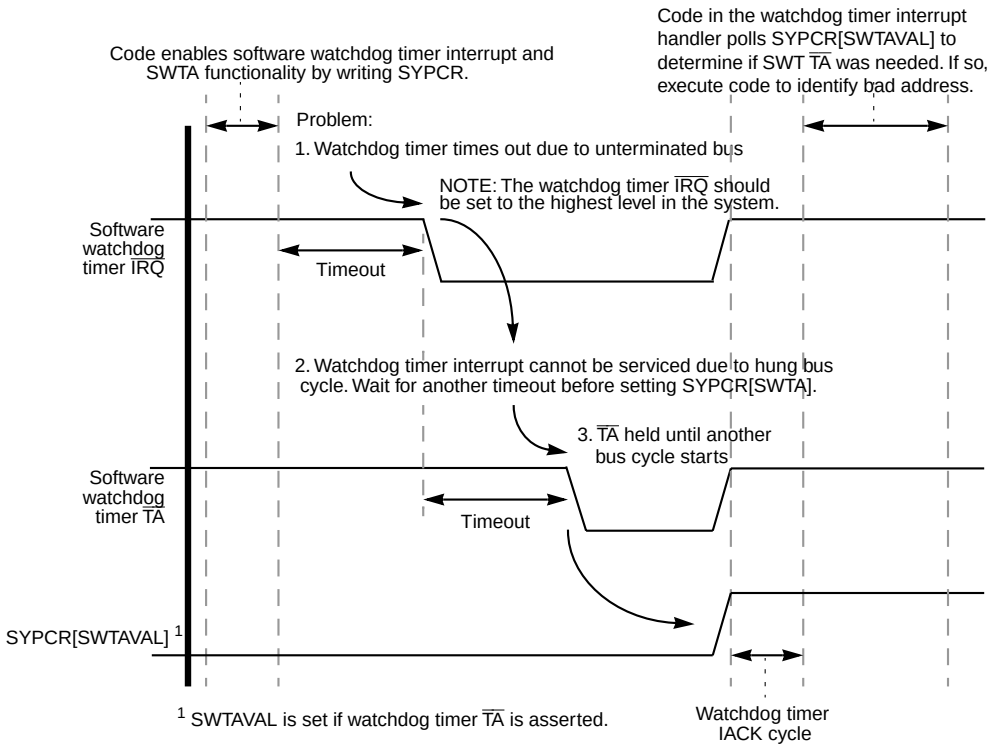


Figure 6-4. MCF5307 Embedded System Recovery from Unterminated Access

When the watchdog timer times out and SYPCR[SWRI] is programmed for a software reset, an internal reset is asserted and RSR[SWTR] is set.

To prevent the watchdog timer from interrupting or resetting, the SWSR must be serviced by performing the following sequence:

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

Both writes must occur in order before the timeout, but any number of instructions or SWSR accesses 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 SYPCR values after the software watchdog timer has been enabled with the setting of SYPCR[SWE], because it is difficult to determine the state of the watchdog timer while it is running. The countdown value is constantly compared with the timeout period specified by SYPCR[SWP,SWT]. Therefore, altering SWP and SWT improperly causes unpredictable processor behavior. The following steps must be taken to change SWP or SWT:

1. Disable the software watchdog timer by clearing SYPCR[SWE].
2. Reset the counter by writing 0x55 and then 0xAA to SWSR.
3. Update SYPCR[SWT,SWP].
4. Reenable the watchdog timer by setting SYPCR[SWE]. This can be done in step 3.

6.2.5 System Protection Control Register (SYPCR)

The SYPCR, Figure 6-5, controls the software watchdog timer, timeout periods, and software watchdog timer transfer acknowledge. The SYPCR can be read at any time, but can be written only if a software watchdog timer $\overline{\text{IRQ}}$ is not pending. At system reset, the software watchdog timer is disabled.

	7	6	5	4	3	2	1	0
Field	SWE	SWRI	SWP	SWT		SWTA	SWTAVAL	—
Reset	0000_0000							
R/W	R/W							
Address	MBAR + 0x01							

Figure 6-5. System Protection Control Register (SYPCR)

Table 6-4 describes SYPCR fields.

Table 6-4. SYPCR Field Descriptions

Bits	Name	Description
7	SWE	Software watchdog timer enable 0 Software watchdog timer disabled 1 Software watchdog timer enabled
6	SWRI	Software watchdog reset/interrupt select 0 If a timeout occurs, the watchdog timer generates an interrupt to the core processor at the level programmed into ICR0[IL]. 1 The software watchdog timer causes soft reset to be asserted for all modules of the part except for the PLL (reset mode selects, such as PP_RESET_SEL or chip-select settings, should not change).
5	SWP	Software watchdog prescaler. This bit interacts with SYPCR[SWT]. 0 Software watchdog timer clock not prescaled. 1 Software watchdog timer clock prescaled by 8192.
4–3	SWT	Software watchdog timing delay. SWT and SWP select the timeout period for the watchdog timer. At system reset, the software watchdog timer is set to the minimum timeout period. SWP = 0 / SWP = 1 00 2^9/system frequency / 00 2^{22}/system frequency 01 2^{11}/system frequency / 01 2^{24}/system frequency 10 2^{13}/system frequency / 10 2^{26}/system frequency 11 2^{15}/system frequency / 11 2^{28}/system frequency Note that if SWP and SWT are modified to select a new software timeout, the software service sequence must be performed (0x55 followed by 0xAA written to the SWSR) before the new timeout period takes effect.

Table 6-4. SYPCR Field Descriptions (Continued)

Bits	Name	Description
2	SWTA	Software watchdog transfer acknowledge enable 0 SWTA transfer acknowledge disabled 1 SWTA asserts transfer acknowledge enabled. After one timeout period of the unacknowledged assertion of the software watchdog timer interrupt, the software watchdog transfer acknowledge asserts, which allows the watchdog timer to terminate a bus cycle and allow the IACK to occur.
1	SWTAVAL	Software watchdog transfer acknowledge valid 0 SWTA transfer acknowledge has not occurred. 1 SWTA transfer acknowledge has occurred. Write a 1 to clear this flag bit.

6.2.6 Software Watchdog Interrupt Vector Register (SWIVR)

The SWIVR, shown in Figure 6-6, contains the 8-bit interrupt vector (SWIV) that the SIM returns during an interrupt-acknowledge cycle in response to a software watchdog timer-generated interrupt. SWIVR is set to the uninitialized vector 0x0F at system reset.

	7 ... 0
Field	SWIV
Reset	0000_1111
R/W	Supervisor write only
Address	MBAR + 0x002

Figure 6-6. Software Watchdog Interrupt Vector Register (SWIVR)

Note that the software watchdog interrupt cannot be autovectored.

6.2.7 Software Watchdog Service Register (SWSR)

The SWSR, shown in Figure 6-7, is where the software watchdog timer servicing sequence should be written. To prevent a watchdog timer timeout, the software service sequence must be performed (0x55 followed by 0xAA written to the SWSR). Both writes must be performed in order before the timeout, but any number of instructions or accesses to the SWSR can be executed between the two writes. If the timer has timed out, writing to SWSR does not cancel the interrupt (that is, IPR[SWT] remains set). The interrupt is cancelled (and SWT is cleared) automatically when the IACK cycle is run.

	7 ... 0
Field	SWSR
Reset	Undetermined
R/W	Supervisor write only
Address	MBAR + 0x003

Figure 6-7. Software Watchdog Service Register (SWSR)

6.2.8 PLL Clock Control for CPU STOP Instruction

The SIM contains the PLL clock control register, which is described in detail in Section 7.2.4, “PLL Control Register (PLLCR).” PLLCR[ENBSTOP,PLLIPL] are significant to the operation of the SIM, and are described as follows:

- PLLCR[ENBSTOP] must be set for the ColdFire CPU STOP instruction to be acknowledged. This bit is cleared at reset and must be set for the MCF5307 to enter low-power modes. The CPU STOP instruction stops only clocks to the core processor. All internal modules remain clocked and can generate interrupts to restart the ColdFire core. For example, the on-chip timer can be used to interrupt the processor after a given timer countdown.
- PLLCR[PLLIPL] determines the minimum level at which an interrupt (decoded as an interrupt priority level or IPL) must occur to awaken the PLL. The PLL then turns clocks back on to the core processor and interrupt exception processing takes place. Table 6-5 describes PLLIPL settings to be compared against the interrupt ranges that awaken the core processor from a CPU STOP instruction.

Table 6-5. PLLIPL Settings

PLLIPL	Description
000	Any interrupts can wake core
001	Interrupts 2–7
010	Interrupts 3–7
011	Interrupts 4–7
100	Interrupts 5–7
101	Interrupts 6–7
110	Interrupt 7 only
111	No interrupts can wake core

6.2.9 Pin Assignment Register (PAR)

The pin assignment register (PAR), Figure 6-8, allows the selection of pin assignments.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	PAR15	PAR14	PAR13	PAR12	PAR11	PAR10	PAR9	PAR8	PAR7	PAR6	PAR5	PAR4	PAR3	PAR2	PAR1	PAR0
PARn = 0	PP15	PP14	PP13	PP12	PP11	PP10	PP9	PP8	PP7	PP6	PP5	PP4	PP3	PP2	PP1	PP0
PARn = 1	A31	A30	A29	A28	A27	A26	A25	A24	$\overline{\text{TIP}}$	$\overline{\text{DREQ0}}$	$\overline{\text{DREQ1}}$	TM2	TM1	TM0	TT1	TT0
Reset	Determined by driving D4/ADDR_CONFIG with a 1 or 0 when $\overline{\text{RSTI}}$ negates. The system is configured as PP[15:0] if D4 is low; otherwise alternate pin functions selected by PAR = 1 are used.															
R/W	R/W															
Address	Address MBAR + 0x004															

Figure 6-8. Pin Assignment Register (PAR)

6.2.10 Bus Arbitration Control

This section describes the bus arbitration register and the four arbitration schemes.

6.2.10.1 Default Bus Master Park Register (MPARK)

The MPARK, shown in Figure 6-9, determines the default bus master arbitration between internal transfers (core and DMA module) and between internal and external transfers to internal resources. This arbitration is needed because external masters can access internal registers within the MCF5307 peripherals.

	7	6	5	4	3	2	0
Field	PARK		IARBCTRL	EARBCTRL	SHOWDATA	—	BCR24BIT
Reset	0000_0000						
R/W	R/W						
Address	MBAR + 0x0C						

Figure 6-9. Default Bus Master Register (MPARK)

Table 6-6 describes MPARK bits.

Table 6-6. MPARK Field Descriptions

Bits	Name	Description
7–6	PARK	Park. Indicates the arbitration priority of internal transfers among MCF5307 resources. 00 Round-robin between DMA and ColdFire core 01 Park on master ColdFire core 10 Park on master DMA module 11 Park on current master Use of this field is described in detail in Section 6.2.10.1.1, “Arbitration for Internally Generated Transfers (MPARK[PARK]).”
5	IARBCTRL	Internal bus arbitration control. Controls external device access to the MCF5307 internal bus. 0 Arbitration disabled (single-master system) 1 Arbitration enabled. IARBCTRL must be set if external masters are using internal resources like the DRAM controller or chip selects. Use of this bit depends on whether the system has single or multiple masters, as follows: • In a single-master system, IARBCTRL should stay cleared, disabling internal arbitration by external masters. In this scenario, MPARK[PARK] applies only to priority of internal masters over one another. Note that the internal DMA (master 3) has priority over the ColdFire core (master 2), if internal DMA bandwidth is at its maximum (BWC = 000). • In multiple master systems that expect to use internal resources like the DRAM controller or chip selects, internal arbitration should be enabled. The external master defaults to the highest priority internal master anytime $\overline{BG}$ is negated.
4	EARBCTRL	External bus arbitration control. Enables internal register memory space to external bus arbitration. Internal registers are those accessed at offsets to the MBAR. These include the SIM, DMA, chip selects, timers, UARTs, I^2C, and parallel port registers. These registers do not include the MBAR; only the core can access the MBAR. 0 Arbitration disabled 1 Arbitration enabled The use of this field is described in detail in Section 6.2.10.1.2, “Arbitration between Internal and External Masters for Accessing Internal Resources.”

Table 6-6. MPARK Field Descriptions (Continued)

Bits	Name	Description
3	SHOWDATA	Enable internal register data bus to be driven on external bus. EARBCTRL must be set for this function to work. Section 6.2.10.1.2, “Arbitration between Internal and External Masters for Accessing Internal Resources,” describes the proper use of SHOWDATA. 0 Do not drive internal register data bus values to external bus. 1 Drive internal register data bus values to external bus.
2–1	—	Reserved, should be cleared.
0	BCR24BIT	Controls the BCR and address mapping for DMA. Allows the BCR to be used as a 24-bit register. Chapter 12, “DMA Controller Module,” describes the BCRs. 0 DMA BCRs function as 16-bit counters. 1 DMA BCRs function as 24-bit counters.

6.2.10.1.1 Arbitration for Internally Generated Transfers (MPARK[PARK])

MPARK[PARK] prioritizes internal transfers, which can be initiated by the core and the on-chip DMA module, which contains all four DMA channels. Priority among the four DMA channels in the module is determined by the BWC bits in their respective DMA control registers (see Chapter 12, “DMA Controller Module”).

The four arbitration schemes for internally generated transfers are described as follows:

- Round-robin scheme (PARK = 00)—Figure 6-10 shows round-robin arbitration between the core and DMA module. Bus mastership alternates between the core and DMA module.

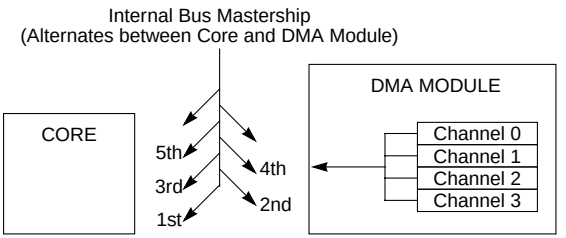


Figure 6-10. Round Robin Arbitration (PARK = 00)

The DMA module presents only the highest-priority DMA request, and bus mastership alternates between the core and DMA channel as long as both are requesting bus mastership. Section 12.5.4.1, “External Request and Acknowledge Operation,” includes a timing diagram showing a lower-priority DMA transfer.

When the processor is initialized, the core has first priority. If DMA channels 0 and 1 (both set to BWC = 010) assert an internal bus request during a core-generated bus transfer, DMA channel 0 would gain bus mastership next. However, if the core requests the bus during this DMA transfer, bus mastership returns to the core rather than being granted to DMA channel 1.

Note that the internal DMA has higher priority than the core if the internal DMA has its bandwidth BWC bits set to 000 (maximum bandwidth).

- Park on master core priority (PARK = 01)—The core retains bus mastership as long as it needs it. After it negates its internal bus request, the core does not have to rearbitrate for the bus unless the DMA module has requested the bus when it is idle. The DMA module can be granted bus mastership only when the core is not asserting its bus request. See Figure 6-11.

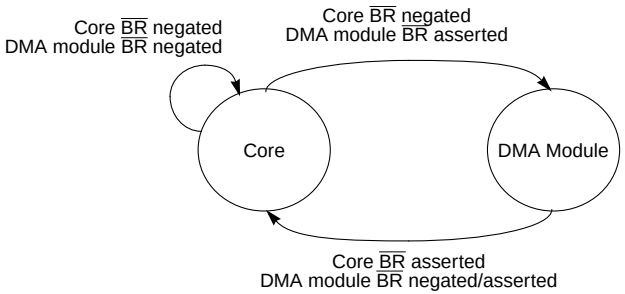


Figure 6-11. Park on Master Core Priority (PARK = 01)

- Park on master DMA priority (PARK = 10)—The DMA module retains bus mastership as long as it needs it. After it negates its internal bus request, the DMA module does not have to rearbitrate for the bus unless the core has requested the bus when it is idle. The core can be granted bus mastership only when the DMA module is not asserting its bus request. See Figure 6-12.

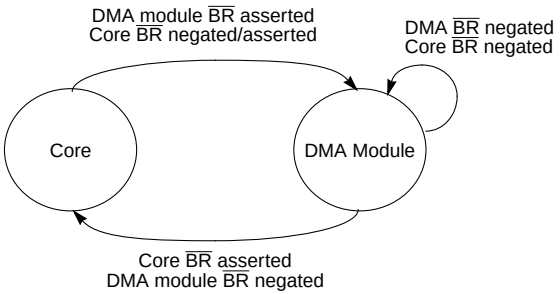


Figure 6-12. Park on DMA Module Priority (PARK = 10)

- Park on current master priority (PARK = 11)—The current bus master retains mastership as long as it needs the bus. The other device can become the bus master only when the bus is idle. For example, if the core is bus master out of reset, it retains mastership as long as it needs the bus. It loses mastership only when it negates its bus request signal and the DMA asserts its internal bus request signal. At this point the DMA module is the bus master, and retains bus mastership as long as it needs the bus. See Figure 6-13.

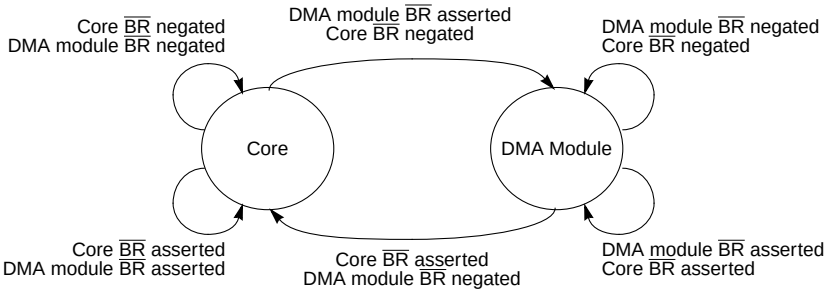


Figure 6-13. Park on Current Master Priority (PARK = 01)

6.2.10.1.2 Arbitration between Internal and External Masters for Accessing Internal Resources

If an external device is programmed to access internal MCF5307 resources (EARBCTRL = 1), the external device can gain bus mastership only when $\overline{BG}$ is negated. This means neither the core nor the DMA controller can access the external bus until the external device asserts $\overline{BG}$. After the external master finishes its bus transfer and asserts $\overline{BG}$, the core has priority on the next available bus cycle regardless of the value of PARK. Thus if the core asserts its internal bus request on this first bus cycle, it executes a bus cycle even if PARK indicates the DMA should have priority. Then, after the bus transfer, the PARK scheme returns to programmed functioning and the DMA is given bus mastership.

NOTE:

In all arbitration modes, if $\overline{BG}$ is negated, the external master interface has highest priority. In this case, the ColdFire core has second-highest priority, until the internal bus grant is asserted.

- In a single-master system, the setting of EARBCTRL does not affect arbitration performance. Typically, $\overline{BG}$ is tied low and the MCF5307 always owns the external bus and internal register transfers are already shown on the external bus. In a system where MCF5307 is the only master, this bit may remain cleared.

  If the system needs external visibility of the data bus values during internal register transfers for system debugging, both EARBCTRL and SHOWDATA must be set.

  Note that when an internal register transfer is driven externally, $\overline{TA}$ becomes an output, which is asserted (normally an input) to prevent external devices and

memories from responding to internal register transfers that go to the external bus. The $\overline{AS}$ signal and all chip-select-related strobe signals are not asserted.

Do not immediately follow a cycle in which SHOWDATA is set with a cycle using fast termination.

- In multiple-master systems, disabling arbitration with EARBCTRL allows performance improvement because internal register bus transfer cycles do not interfere with the external bus.

  Having internal transfers go external may affect performance in two ways:

  — If the internal device does not control the bus immediately, the core stalls until it wins arbitration of the external bus.

  — If the core wins arbitration instantly, it may kick the external master off of the external bus unnecessarily for a transfer that did not need the external bus. For debug, where this performance penalty is not a concern, setting EARBCTRL and SHOWDATA provides external visibility of the internal bus cycles.

Chapter 7
Phase-Locked Loop (PLL)

This chapter describes configuration and operation of the phase-locked loop (PLL) module. It describes in detail the registers and signals that support the PLL implementation.

7.1 Overview

The basic features of the MCF5307 PLL implementation are as follows:

- The PLL locks to the clock input (CLKIN) frequency. It provides a processor clock (PCLK) that is twice the input clock frequency and a programmable system bus clock output (BCLKO) that is 1/2, 1/3, or 1/4 the PCLK frequency.
- A buffered processor status clock (PSTCLK) is equal to the PCLK frequency, as indicated in Figure 7-1. This signal is made available for system development.

The PLL module has the following three modes of operation:

- Reset mode—In reset mode, the core/bus frequency ratio and other configuration information is sampled. At reset, the PLL asserts the reset out signal, $\overline{\text{RSTO}}$.
- Normal mode—During normal operations, the divide ratio is programmed at reset and is clock-multiplied to provide a maximum frequency of 90 MHz
- Reduced-power mode—In reduced-power mode, the high-speed processor core clocks are turned off without losing the register contents so that the system can be reenabled by an unmasked interrupt or reset.

Figure 7-1 shows the frequency relationships of PLL module clock signals.

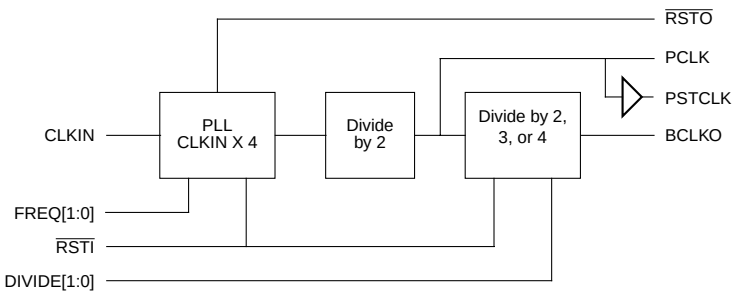


Figure 7-1. PLL Module Block Diagram

7.1.1 PLL:PCLK Ratios

The specifications for the clocks in the PLL module are summarized in Table 0-1.

Table 0-1. PLL Clock Specifications

Symbol	Description	Frequency		
—	PLL lock time	2.2 mS with CLKIN running at 45 MHz		
CLKIN	Input clock	16.67 MHz—45 MHz		
PCLK	Internal processor clock	33.34 MHz—90 MHz (CLKIN x 2)		
PSTCLK	Processor status clock	33.34 MHz—90 MHz (CLKIN x 2)		
BCLKO	Output clock	16.67 MHz—45 MHz	11.11 MHz—30 MHz	8.24 MHz—22.5 MHz
BCLKO/PCLK ratio		**1/2**	**1/3**	**1/4**

7.2 PLL Operation

The following sections provide detailed information about the three PLL modes.

7.2.1 Reset/Initialization

The PLL receives $\overline{\text{RSTI}}$ as an input directly from the pin. Additionally, signals are multiplexed with D[3:0]/FREQ[1:0]:DIVIDE[1:0] while $\overline{\text{RSTI}}$ is asserted. These signals are sampled during reset and registered by the PLL on the negation of $\overline{\text{RSTI}}$ to provide initialization information. FREQ[1:0] and DIVIDE[1:0] are used by the PLL to select the CLKIN frequency range and set the CLKIN/PCLK ratio, respectively.

7.2.2 Normal Mode

PCLK is divided to create the system bus clock, BCLKO. At reset, the logic level of DIVIDE[1:0]/D[1:0] determines the BCLKO divisor. The bus clock can be 1/2, 1/3, or 1/4 of the PCLK frequency.

7.2.3 Reduced-Power Mode

The PCLK can be turned off in a predictable manner to conserve system power. To allow fast restart of the MCF5307 processor core, the PLL continues to operate at the frequency configured at reset. PCLK is disabled using the CPU STOP instruction and resumes normal operation on interrupt, as described in Section 7.2.4, “PLL Control Register (PLLCR).”

7.2.4 PLL Control Register (PLLCR)

The PLL control register (PLLCR), Figure 7-2, provides control over the PLL.

	7	6	5	4	3	2	1	0
Field	ENBSTOP	PLLIPL			—			
Reset	0000_0000							
R/W	R/W							
Address	MBAR + 0x08							

Figure 7-2. PLL Control Register (PLLCR)

Table 7-1 describes PLLCR bits.

Table 7-1. PLLCR Field Descriptions

Bit	Name	Description
7	ENBSTOP	Enable CPU STOP instruction. Must be set for the ColdFire CPU STOP instruction to be acknowledged. Cleared at reset and must be subsequently set for the processor to enter low-power modes. Only clocks to the core are turned off because of the CPU STOP instruction. Internal modules remain clocked and can generate interrupts to restart the ColdFire core. 0 Disable CPU STOP 1 Enable CPU STOP; STOP instruction turns off clocks to the ColdFire core.
6–4	PLLIPL	PLL interrupt priority level to wake up from CPU STOP. Determines the minimum level an interrupt (decoded as an interrupt priority level) must be to waken the PLL. The PLL then turns clocks back on to the core processor and interrupt exception processing occurs. 000 Any interrupts can wake core 001 Interrupts 2–7 010 Interrupts 3–7 011 Interrupts 4–7 100 Interrupts 5–7 101 Interrupts 6–7 110 Interrupt 7 only 111 No interrupts can wake core. Any reset, including a watchdog reset, can wake the core. No PLL phase lock time is required.
3–0	—	Reserved, should be cleared.

7.3 PLL Port List

Table 7-2 describes PLL module inputs.

Table 7-2. PLL Module Input SIgnals

SIgnal	Description
CLKIN	Input clock to the PLL. Input frequency must not be changed during operation. Changes are recognized only at reset.
$\overline{\text{RSTI}}$	Active-low asynchronous input that, when asserted, indicates PLL is to enter reset mode. As long as $\overline{\text{RSTI}}$ is asserted, the PLL is held in reset and does not begin to lock.

Table 7-2. PLL Module Input SIgnals

SIgnal	Description
FREQ[1:0]	Input bus indicating the CLKIN frequency range. FREQ[1:0] are multiplexed with D[3:2] and are sampled while $\overline{\text{RSTI}}$ is asserted. FREQ[1:0] must be correctly set for proper operation. These signals do not affect CLKIN frequency but are required to set up the analog PLL to handle the input clock frequency. 00 16.6—27.999 MHz 01 28—38.999 MHz 10 39—45 MHz 11 Not used
DIVIDE[1:0]	The MCF5307 samples clock ratio encodings on the lower data bits of the bus to determine the CLKIN-to-processor clock ratio. D[1:0]/DIVIDE[1:0] support the divide-ratio combinations. 00 1/4 01 Not used 10 1/2 11 1/3

Table 7-3 describes PLL module outputs.

Table 7-3. PLL Module Output Signals

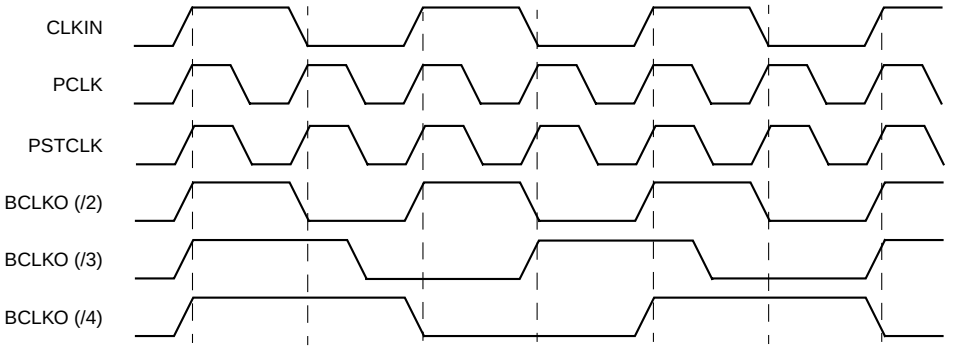
Output	Description
BCLKO	This bus clock output provides a divided version of the processor clock frequency, determined by DIVIDE[1:0].
PSTCLK	Provides a buffered processor status clock at 2X the CLKIN frequency. PSTCLK is a delayed version of PCLK. See Section 7.4.1, “PCLK, PSTCLK, and BCLKO,” and Figure 7-1.
$\overline{\text{RSTO}}$	This output provides an external reset for peripheral devices.

7.4 Timing Relationships

The MCF5307 uses CLKIN and BCLKO, which is generated by the PLL and may be used as the bus timing reference for external devices. The MCF5307 BCLKO frequency can be 1/2, 1/3, or 1/4 the processor clock. In this document, bus timings are referenced from BCLKO. Furthermore, depending on the user configuration, the BCLKO-to-processor clock ratio may differ from the CLKIN-to-processor clock ratio.

7.4.1 PCLK, PSTCLK, and BCLKO

Figure 7-3 shows the frequency relationships between PCLK, PSTCLK,CLKIN, and the three possible versions of BCLKO. This figure does not show the skew between CLKIN and PCLK, PSTCLK, and BCLKO. PSTCLK is equal to frequency of PCLK. Similarly, the skew between PCLK and BCLKO is unspecified.



NOTE: The clock signals are shown with edges aligned to show frequency relationships only. Actual signal edges have some skew between them.

Figure 7-3. CLKIN, PCLK, PSTCLK, and BCLKO Timing

7.4.2 $\overline{\text{RSTI}}$ Timing

Figure 7-4 shows PLL timing during reset. As shown, $\overline{\text{RSTI}}$ must be asserted for at least 80 CLKIN cycles to give the MCF5307 time to begin its initialization sequence. At this time, the configuration pins should be asserted (D[3:2] for FREQ[1:0] and D[1:0] for DIVIDE[1:0]), meeting the minimum setup and hold times to $\overline{\text{RSTI}}$ given in Chapter 20, “Electrical Specifications.”

On the rising edge of BCLKO before the rising edge of $\overline{\text{RSTI}}$, the data on D[7:0] is latched and the PLL begins ramping to its final operating frequency. During this ramp and lock time, BCLKO and PSTCLK are held low. The PLL locks in about 2.2 mS with a 45-MHz CLKIN, at which time BCLKOand PSTCLK begin normal operation in the specified mode. The PLL requires 100,000 CLKIN cycles to guarantee PLL lock. To allow for reset of external peripherals requiring a clock source, $\overline{\text{RSTO}}$ remains asserted for a number of BCLKO cycles, as shown in Figure 7-4.

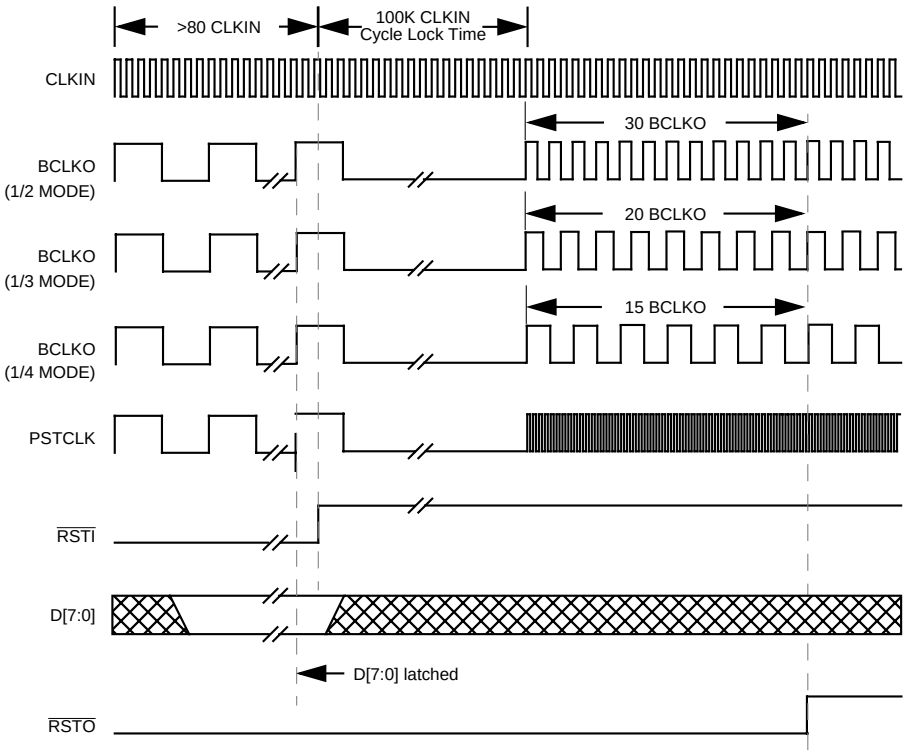


Figure 7-4. Reset and Initialization Timing

7.5 PLL Power Supply Filter Circuit

To ensure PLL stability, the power supply to the PLL power pin should be filtered using a circuit similar to the one in Figure 7-5. The circuit should be placed as close as possible to the PLL power pin to ensure maximum noise filtering.

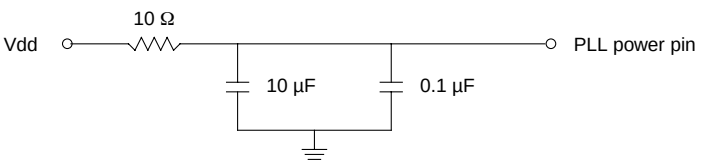


Figure 7-5. PLL Power Supply Filter Circuit

Chapter 8
I²C Module

This chapter describes the MCF5307 I²C module, including I²C protocol, clock synchronization, and the registers in the I²C programing model. It also provides extensive programming examples.

8.1 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 requiring occasional communications over a short distance between many devices. The flexible I²C allows additional devices to be connected to the bus for expansion and system development.

The I²C system is a true multiple-master bus including arbitration and collision detection that prevents data corruption if 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.

8.2 Interface Features

The I²C module has the following key features:

- Compatibility with I²C bus standard
- Support for 3.3-V tolerant devices
- Multiple-master operation
- Software-programmable for one of 64 different serial clock frequencies
- Software-selectable acknowledge bit
- Interrupt-driven, byte-by-byte data transfer
- Arbitration-lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated START signal generation

- Acknowledge bit generation/detection
- Bus-busy detection

Figure 8-1 is a block diagram of the I^{2}C module.

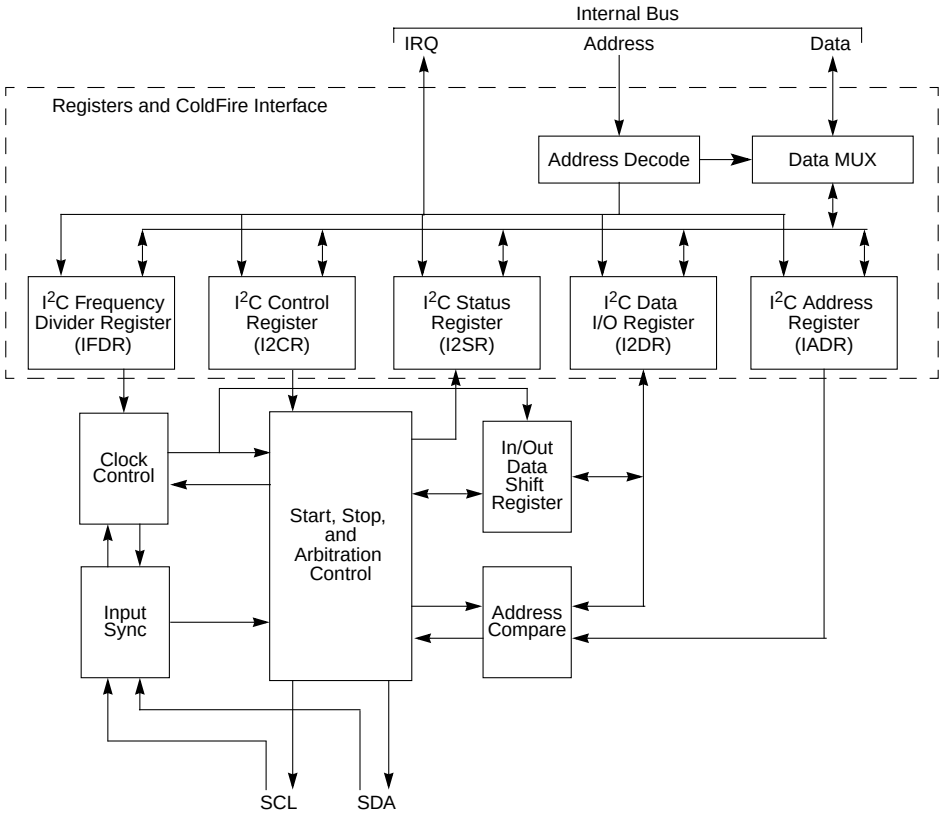


Figure 8-1. I^{2}C Module Block Diagram

Figure 8-1 shows the relationships of the I^{2}C registers, listed below:

- I^{2}C address register (IADR)
- I^{2}C frequency divider register (IFDR)
- I^{2}C control register (I2CR)
- I^{2}C status register (I2SR)
- I^{2}C data I/O register (I2DR)

These registers are described in Section 8.5, “Programming Model.”

8.3 I2C System Configuration

The I2C module uses a serial data line (SDA) and a serial clock line (SCL) for data transfer. For I2C compliance, all devices connected to these two signals must have open drain or open collector outputs. (There is no such requirement for inputs.) The logic AND function is exercised on both lines with external pull-up resistors.

Out of reset, the I2C default is as slave receiver. Thus, when not programmed to be a master or responding to a slave transmit address, the I2C module should return to the default slave receiver state. See Section 8.6.1, “Initialization Sequence,” for exceptions.

NOTE:

The I2C module is designed to be compatible with the Philips I2C bus protocol. For information on system configuration, protocol, and restrictions, see *The I2C Bus Specification, Version 2.1*.

8.4 I2C Protocol

Normally, a standard communication is composed of the following parts:

1. START signal—When no other device is bus master (both SCL and SDA lines are at logic high), a device can initiate communication by sending a START signal (see A in Figure 8-2). A START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a data transfer (each data transfer can be several bytes long) and awakens all slaves.

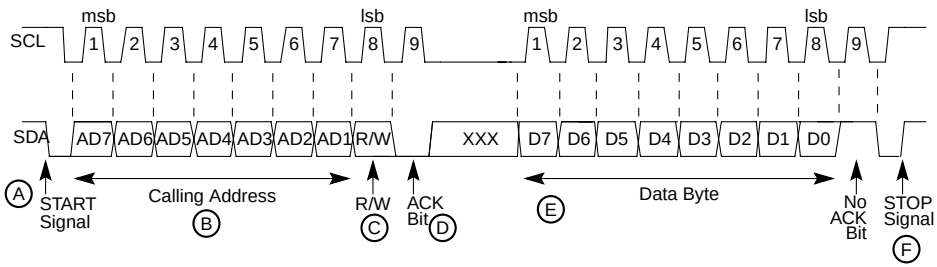


Figure 8-2. I2C Standard Communication Protocol

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.

   Each slave must have a unique address. An I2C master must not transmit an address that is the same as its slave address; it cannot be master and slave at the same time.

The slave whose address matches that sent by the master pulls SDA low at the ninth clock (D) to return an acknowledge bit.

3. Data transfer—When successful slave addressing is achieved, the data transfer can proceed (E) 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 SCL is low and must be held stable while SCL is high, as Figure 8-2 shows. SCL is pulsed once for each data bit, with the msb being sent first. The receiving device must acknowledge each byte by pulling SDA low at the ninth clock; therefore, a data byte transfer takes nine clock pulses.

   If it does not acknowledge the master, the slave receiver must leave SDA high. The master can then generate a STOP signal to abort the data transfer or generate a START signal (repeated start, shown in Figure 8-3) to start a new calling sequence.

   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 SDA for the master to generate a STOP or START signal.

4. 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 SDA while SCL is at logical high (F). Note that a master can generate a STOP even if the slave has made an acknowledgment, at which point the slave must release the bus.

Instead of signalling a STOP, the master can repeat the START signal, followed by a calling command, (A in Figure 8-3). A repeated START occurs when a START signal is generated without first generating a STOP signal to end the communication.

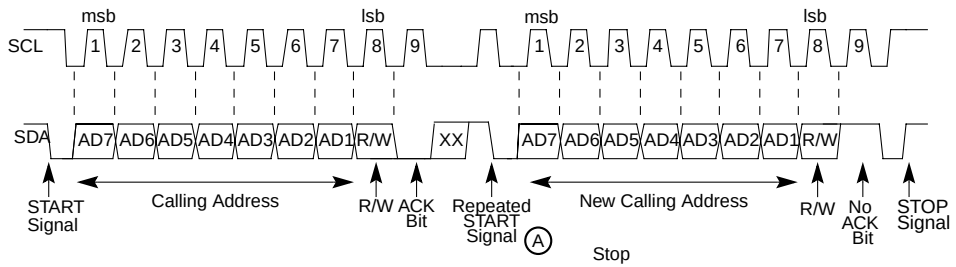


Figure 8-3. Repeated START

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.

8.4.1 Arbitration Procedure

If multiple devices simultaneously request the bus, the bus clock is determined by a synchronization procedure in which the low period equals the longest clock-low period among the devices and the high period equals the shortest. A data arbitration procedure

determines the relative priority of competing devices. A device loses arbitration if it sends logic high while another sends logic low; it immediately switches to slave-receive mode and stops driving SDA. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, hardware sets I2SR[IAL] to indicate loss of arbitration.

8.4.2 Clock Synchronization

Because wire-AND logic is used, a high-to-low transition on SCL affects devices connected to the bus. Devices start counting their low period when the master drives SCL low. When a device clock goes low, it holds SCL low until the clock high state is reached. However, the low-to-high change in this device clock may not change the state of SCL if another device clock is still in its low period. Therefore, the device with the longest low period holds the synchronized clock SCL low. Devices with shorter low periods enter a high wait state during this time (See Figure 8-4). When all devices involved have counted off their low period, the synchronized clock SCL is released and pulled high. There is then no difference between device clocks and the state of SCL, so all of the devices start counting their high periods. The first device to complete its high period pulls SCL low again.

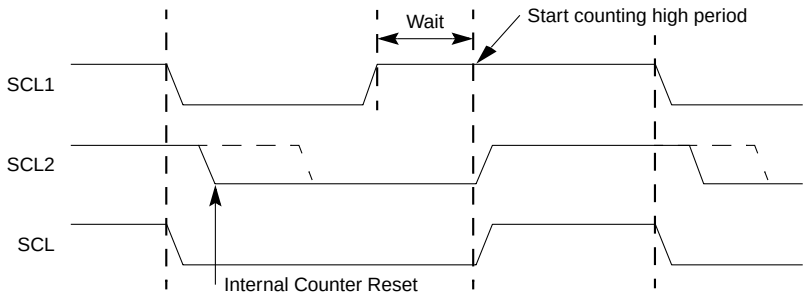


Figure 8-4. Synchronized Clock SCL

8.4.3 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfers. Slave devices can hold SCL low after completing one byte transfer (9 bits). In such a case, the clock mechanism halts the bus clock and forces the master clock into wait states until the slave releases SCL.

8.4.4 Clock Stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is longer than the master SCL low period, the resulting SCL bus signal low period is stretched.

8.5 Programming Model

Table 8-1 lists the configuration registers used in the I^{2}C interface.

Table 8-1. I^{2}C Interface Memory Map

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x280	I^{2}C address register (IADR) [p. 8-6]	Reserved		
0x284	I^{2}C frequency divider register (IFDR) [p. 8-7]	Reserved		
0x288	I^{2}C control register (I2CR) [p. 8-8]	Reserved		
0x28C	I^{2}C status register (I2SR) [p. 8-9]	Reserved		
0x290	I^{2}C data I/O register (I2DR) [p. 8-10]	Reserved		

NOTE:

External masters cannot access the MCF5307's on-chip memories or MBAR, but can access any I^{2}C module register.

8.5.1 I^{2}C Address Register (IADR)

The IADR holds the address the I^{2}C responds to when addressed as a slave. Note that it is not the address sent on the bus during the address transfer.

	7	6	5	4	3	2	1	0
Field	ADR							—
Reset	0000_0000							
R/W	Read/Write							
Address	MBAR + 0x280							

Figure 8-5. I^{2}C Address Register (IADR)

Table 8-2 describes IADR fields.

Table 8-2. I^{2}C Address Register Field Descriptions

Bits	Name	Description
7–1	ADR	Slave address. Contains the specific slave address to be used by the I^{2}C module. Slave mode is the default I^{2}C mode for an address match on the bus.
0	—	Reserved, should be cleared.

8.5.2 I²C Frequency Divider Register (IFDR)

The IFDR, Figure 8-6, provides a programmable prescaler to configure the clock for bit-rate selection.

	7	6	5	4	3	2	1	0
Field	—	—	IC	IC	IC	IC	IC	IC
Reset	0	0	0	0	0	0	0	0
R/W	Read/Write							
Address	MBAR + 0x284							

Figure 8-6. I²C Frequency Divider Register (IFDR)

Table 8-3 describes IFDR[IC].

Table 8-3. IFDR Field Descriptions

Bits	Name	Description
7–6	—	Reserved, should be cleared.
5–0	IC	I²C clock rate. Prescales the clock for bit-rate selection. Due to potentially slow SCL and SDA rise and fall times, bus signals are sampled at the prescaler frequency. The serial bit clock frequency is equal to BCLK0 divided by the divider shown below. Note that IC can be changed anywhere in a program.

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

8.5.3 I²C Control Register (I2CR)

The I2CR is used to enable the I²C module and the I²C interrupt. It also contains bits that govern operation as a slave or a master.

	7	6	5	4	3	2	1	0
Field	IEN	IIEN	MSTA	MTX	TXAK	RSTA	—	
Reset	0000_0000							
R/W	Read/Write							
Address	MBAR + 0x288							

Figure 8-7. I²C Control Register (I2CR)

Table 8-4 describes I2CR fields.

Table 8-4. I2CR Field Descriptions

Bits	Name	Description
7	IEN	I²C enable. Controls the software reset of the entire I²C module. If the module is enabled in the middle of a byte transfer, slave mode ignores the current bus transfer and starts operating when the next start condition is detected. Master mode is not aware that the bus is busy; so initiating a start cycle may corrupt the current bus cycle, ultimately causing either the current master or the I²C module to lose arbitration, after which bus operation returns to normal. 0 The module is disabled, but registers can still be accessed. 1 The I²C module is enabled. This bit must be set before any other I2CR bits have any effect.
6	IIEN	I²C interrupt enable. 0 I²C module interrupts are disabled, but currently pending interrupt condition are 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 a slave is addressed, software should set MTX according to I2SR[SRW]. In master mode, MTX should be set according to the type of transfer required. Therefore, for address cycles, MTX is always 1.
3	TXAK	Transmit acknowledge enable. Specifies the value driven onto SDA during acknowledge cycles for both master and slave receivers. Note that writing TXAK applies only when the I²C bus is a receiver. 0 An acknowledge signal is sent to the bus at the ninth clock bit after receiving one byte of data. 1 No acknowledge signal response is sent (that is, acknowledge bit = 1).
2	RSTA	Repeat start. Always read as 0. Attempting a repeat start without bus mastership causes loss of arbitration. 0 No repeat start 1 Generates a repeated START condition.
1–0	—	Reserved, should be cleared.

8.5.4 I²C Status Register (I2SR)

This I2SR contains bits that indicate transaction direction and status.

	7	6	5	4	3	2	1	0
Field	ICF	IAAS	IBB	IAL	—	SRW	IIF	RXAK
Reset	1000_0001							
R/W	R			R/W	R		R/W	R
Address	MBAR + 0x28C							

Figure 8-8. I²CR Status Register (I2SR)

Table 8-5 describes I2SR fields.

Table 8-5. I2SR Field Descriptions

Bits	Name	Description
7	ICF	Data transferring bit. While one byte of data is transferred, ICF is cleared. 0 Transfer in progress 1 Transfer complete. Set by the falling edge of the ninth clock of a byte transfer.
6	IAAS	I²C addressed as a slave bit. The CPU is interrupted if I2CR[IIEN] is set. Next, the CPU must check SRW and set its TX/RX mode accordingly. Writing to I2CR clears this bit. 0 Not addressed. 1 Addressed as a slave. Set when its own address (IADR) matches the calling address.
5	IBB	I²C bus busy bit. Indicates the status of the bus. 0 Bus is idle. If a STOP signal is detected, IBB is cleared. 1 Bus is busy. When START is detected, IBB is set.
4	IAL	Arbitration lost. Set by hardware in the following circumstances. (IAL must be cleared by software by writing zero to it.) • SDA sampled low when the master drives high during an address or data-transmit cycle. • SDA sampled low when the master drives high during the acknowledge bit of a data-receive cycle. • A start cycle is attempted when the bus is busy. • A repeated start cycle is requested in slave mode. • A stop condition is detected when the master did not request it.
3	—	Reserved, should be cleared.
2	SRW	Slave read/write. When IAAS is set, SRW indicates the value of the R/W command bit of the calling address sent from the master. SRW is valid only when a complete transfer has occurred, no other transfers have been initiated, and the I²C module is a slave and has an address match. 0 Slave receive, master writing to slave. 1 Slave transmit, master reading from slave.
1	IIF	I²C interrupt. Must be cleared by software by writing a zero to it 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 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.

8.5.5 I²C Data I/O Register (I2DR)

In master-receive mode, reading the I2DR, Figure 8-9, allows a read to occur and initiates next byte data receiving. In slave mode, the same function is available after it is addressed.

	7	6	5	4	3	2	1	0
Field	D							
Reset	0000_0000							
R/W	Read/Write							
Address	MBAR + 0x290							

Figure 8-9. I²C Data I/O Register (I2DR)

8.6 I²C Programming Examples

The following examples show programming for initialization, signalling START, post-transfer software response, signalling STOP, and generating a repeated START.

8.6.1 Initialization Sequence

Before the interface can transfer serial data, registers must be initialized, as follows:

1. Set IFDR[IC] to obtain SCL frequency from the system bus clock. See Section 8.5.2, “I2C Frequency Divider Register (IFDR).”
2. Update the IADR to define its slave address.
3. Set I2CR[IEN] to enable the I²C bus interface system.
4. Modify the I2CR to select master/slave mode, transmit/receive mode, and interrupt-enable or not.

NOTE:

If IBSR[IBB] when the I²C bus module is enabled, execute the following code sequence before proceeding with normal initialization code. This issues a STOP command to the slave device, placing it in idle state as if it were just power-cycled on.

```
I2CR = 0x0
I2CR = 0xA
dummy read of I2DR
IBSR = 0x0
I2CR = 0x0
```

8.6.2 Generation of START

After completion of the initialization procedure, serial data can be transmitted by selecting the master transmitter mode. On a multiple-master bus system, IBSR[IBB] must be tested to determine whether the serial bus is free. If the bus is free (IBB = 0), the START signal

and the first byte (the slave address) can be sent. The data written to the data register comprises the address of the desired slave and the lsb indicates the transfer direction.

The free time between a STOP and the next START condition is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be necessary to wait until the I2C is busy after writing the calling address to the I2DR before proceeding with the following instructions.

The following example signals START and transmits the first byte of data (slave address):

```
CHFLAG  MOVE.B I2SR,-(A0);Check I2SR[MBB]
        BTST.B #5, (A0)+
        BNE.S CHFLAG;If I2SR[MBB] = 1, wait until it is clear
TXSTART MOVE.B I2CR,-(A0);Set transmit mode
        BSET.B #4,(A0)
        MOVE.B (A0)+, I2CR
        MOVE.B I2CR, -(A0);Set master mode
        BSET.B #5, (A0);Generate START condition
        MOVE.B (A0)+, I2CR
        MOVE.B CALLING,-(A0);Transmit the calling address, D0=R/W
        MOVE.B (A0)+, I2DR
IFREE   MOVE.B I2SR,-(A0);Check I2SR[MBB]
            ;If it is clear, wait until it is set.
        BTST.B #5, (A0)+;
        BEQ.S IFREE;
```

8.6.3 Post-Transfer Software Response

Sending or receiving a byte sets the I2SR[ICF], which indicates one byte communication is finished. I2SR[IIF] is also set. An interrupt is generated if the interrupt function is enabled during initialization by setting I2CR[IIEN]. Software must first clear IIF in the interrupt routine. ICF is cleared either by reading from I2DR in receive mode or by writing to I2DR in transmit mode.

Software can service the I2C I/O in the main program by monitoring IIF if the interrupt function is disabled. Polling should monitor IIF rather than ICF because that operation is different when arbitration is lost.

When an interrupt occurs at the end of the address cycle, the master is always in transmit mode; that is, the address is sent. If master receive mode is required (I2DR[R/W], 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 8-10).

```
I2SR    LEA.L I2SR,-(A7);Load effective address
        BCLR.B #1,(A7)+;Clear the IIF flag
        MOVE.B I2CR,-(A7);Push the address on stack,
```

```
        BTST.B #5,(A7)+;check the MSTA flag
        BEQ.S SLAVE;Branch if slave mode
        MOVE.B I2CR,-(A7);Push the address on stack
        BTST.B #4,(A7)+;check the mode flag
        BEQ.S RECEIVE;Branch if in receive mode
        MOVE.B I2SR,-(A7);Push the address on stack,
        BTST.B #0,(A7)+;check ACK from receiver
        BNE.B END;If no ACK, end of transmission
TRANSMITMOVE.B DATABUF,-(A7);Stack data byte
        MOVE.B (A7)+, I2DR;Transmit next byte of data
```

8.6.4 Generation of STOP

A data transfer ends when the master signals a STOP, which can occur after all data is sent, as in the following example.

```
MASTX   MOVE.B I2SR, -(A7);If no ACK, branch to end
        BTST.B #0,(A7)+
        BNE.B END
        MOVE.B TXCNT,D0;Get value from the transmitting counter
        BEQ.S END;If no more data, branch to end
        MOVE.B DATABUF,-(A7);Transmit next byte of data
        MOVE.B (A7)+,I2DR
        MOVE.B TXCNT,D0;Decrease the TXCNT
        SUBQ.L #1,D0
        MOVE.B D0,TXCNT
        BRA.S EMASTX;Exit
END     LEA.L I2CR,-(A7);Generate a STOP condition
        BCLR.B #5,(A7)+
        EMASTX RTE;Return from interrupt
```

For a master receiver to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last data byte. This is done by setting I2CR[TXAK] before reading the next-to-last byte. Before the last byte is read, a STOP signal must be generated, as in the following example.

```
MASR    MOVE.B RXCNT,D0;Decrease RXCNT
        SUBQ.L #1,D0
        MOVE.B D0,RXCNT
        BEQ.S ENMASR;Last byte to be read
        MOVE.B RXCNT,D1;Check second-to-last byte to be read
        EXTB.L D1
        SUBI.L #1,D1;
        BNE.S NXMAR;Not last one or second last
        LAMAR BSET.B #3,I2CR;Disable ACK
        BRA NXMAR

ENMASR  BCLR.B #5,I2CR;Last one, generate STOP signal
NXMAR   MOVE.B I2DR,RXBUF;Read data and store RTE
```

8.6.5 Generation of Repeated START

After the data transfer, if the master still wants the bus, it can signal another START followed by another slave address without signalling a STOP, as in the following example.

```
RESTART MOVE.B I2CR,-(A7);Repeat START (RESTART)
        BSET.B #2, (A7)
        MOVE.B (A7)+, I2CR
        MOVE.B CALLING,-(A7);Transmit the calling address, D0=R/W-
        MOVE.B CALLING,-(A7);
        MOVE.B (A7)+, I2DR
```

8.6.6 Slave Mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (I2CR[MTX]) according to the I2SR[SRW]. Writing to the I2CR clears the IAAS automatically. The only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred; interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer can now be initiated by writing information to I2DR for slave transmits, or read from I2DR in slave-receive mode. A dummy read of I2DR in slave/receive mode releases SCL, allowing the master to send data.

In the slave transmitter routine, I2SR[RXAK] must be tested before sending the next byte of data. Setting RXAK means an end-of-data signal from the master receiver, after which software must switch it from transmitter to receiver mode. Reading I2DR then releases SCL so that the master can generate a STOP signal.

8.6.7 Arbitration Lost

If several devices try to engage the bus at the same time, one becomes master. Hardware immediately switches devices that lose arbitration to slave receive mode. Data output to SDA stops, but SCL is still generated until the end of the byte during which arbitration is lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with I2SR[IAL] = 1 and I2CR[MSTA] = 0.

If a device that is not a master tries to transmit or do a START, hardware inhibits the transmission, clears MSTA without signalling a STOP, generates an interrupt to the CPU, and sets IAL to indicate a failed attempt to engage the bus. When considering these cases, the slave service routine should first test IAL and software should clear it if it is set.

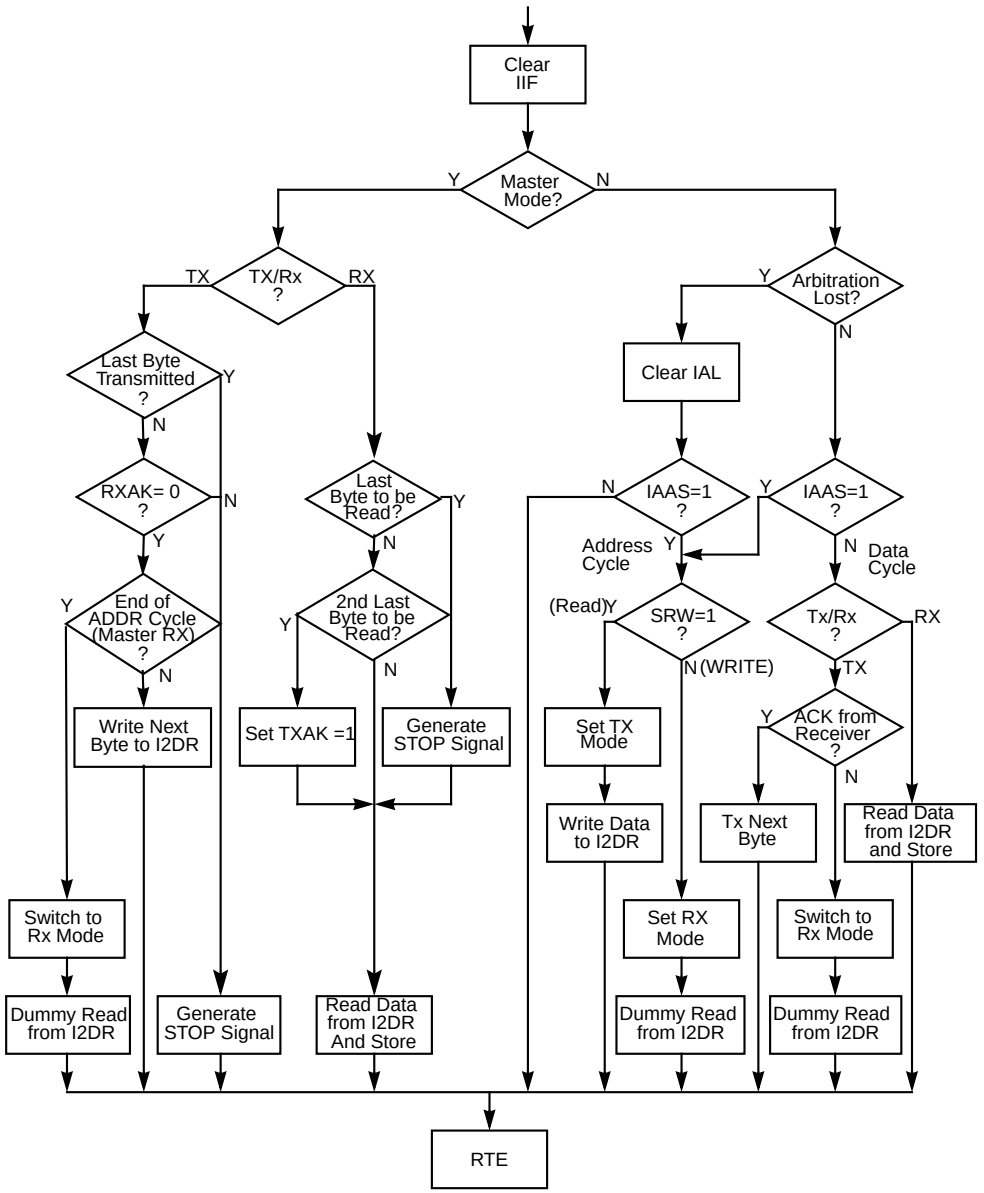


Figure 8-10. Flow-Chart of Typical I^{2}C Interrupt Routine

Chapter 9
Interrupt Controller

This chapter describes the operation of the interrupt controller portion of the system integration module (SIM). It includes descriptions of the registers in the interrupt controller memory map and the interrupt priority scheme.

9.1 Overview

The SIM provides a centralized interrupt controller for all MCF5307 interrupt sources, which consist of the following:

- External interrupts
- Software watchdog timer
- Timer modules
- I^2C module
- UART modules
- DMA module

Figure 9-1 is a block diagram of the interrupt controller.

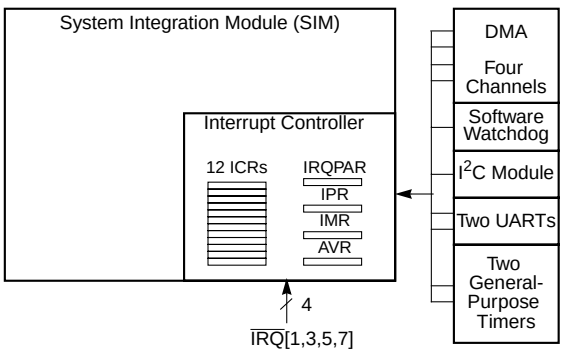


Figure 9-1. Interrupt Controller Block Diagram

The SIM provides the following registers for managing interrupts:

- Each potential interrupt source is assigned one of the 10 interrupt control registers (ICR0–ICR9), which are used to prioritize the interrupt sources.
- The interrupt mask register (IMR) provides bits for masking individual interrupt sources.
- The interrupt pending register (IPR) provides bits for indicating when an interrupt request is being made (regardless of whether it is masked in the IMR).
- The autovector register (AVEC) controls whether the SIM supplies an autovector or executes an external interrupt acknowledge cycle for each IRQ.
- The interrupt port assignment register (IRQPAR) provides the level assignment of the primary external interrupt pins—IRQ5, IRQ3, and IRQ1.

9.2 Interrupt Controller Registers

The interrupt controller register portion of the SIM memory map is shown in Table 9-2.

Table 9-1. Interrupt Controller Registers

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x040	Interrupt pending register (IPR) [p. 9-6]			
0x044	Interrupt mask register (IMR) [p. 9-6]			
0x048	Reserved			Autovector register (AVR) [p. 9-5]
Interrupt Control Registers (ICRs) [p. 9-3]				
0x04C	Software watchdog timer (ICR0) [p. 9-3]	Timer0 (ICR1) [p. 9-3]	Timer1 (ICR2) [p. 9-3]	I^2C (ICR3) [p. 9-3]
0x050	UART0 (ICR4) [p. 9-3]	UART1 (ICR5) [p. 9-3]	DMA0 (ICR6) [p. 9-3]	DMA1 (ICR7) [p. 9-3]
0x054	DMA2 (ICR8) [p. 9-3]	DMA3 (ICR9) [p. 9-3]	Reserved	

Each internal interrupt source has its own interrupt control register (ICR0–ICR9), shown in Table 9-2 and described in Section 9.2.1, “Interrupt Control Registers (ICR0–ICR9).”

Table 9-2. Interrupt Control Registers

MBAR Offset	Register	Name
0x04C	ICR0	Software watchdog timer
0x04D	ICR1	Timer0
0x04E	ICR2	Timer1
0x04F	ICR3	I^2C
0x050	ICR4	UART0
0x051	ICR5	UART1
0x052	ICR6	DMA0

Table 9-2. Interrupt Control Registers (Continued)

MBAR Offset	Register	Name
0x053	ICR7	DMA1
0x054	ICR8	DMA2
0x055	ICR9	DMA3

Internal interrupts are programmed to a level and priority. Each internal interrupt has a unique ICR. Each of the 7 interrupt levels has 5 priorities, for a total of 35 possible priority levels, encompassing internal and external interrupts. The four external interrupt pins offer seven possible settings at a fixed interrupt level and priority.

The IRQPAR determines these settings for external interrupt request levels. External interrupts can be programmed to supply an autovector or execute an external interrupt acknowledge cycle. This is described in Section 9.2.2, “Autovector Register (AVR).”

9.2.1 Interrupt Control Registers (ICR0–ICR9)

The interrupt control registers (ICR0–ICR9) provide bits for defining the interrupt level and priority for the interrupt source assigned to the ICR, shown in Table 9-2.

	7	6	5	4	3	2	1	0
Field	AVEC	—		IL			IP	
Reset	0	—		0_00			00	
R/W	R/W							
Address	MBAR + 0x04C (ICR0); 0x04D (ICR1); 0x04E (ICR2); 0x04F (ICR3); 0x050 (ICR4); 0x051 (ICR5); 0x052 (ICR6); 0x053 (ICR7); 0x054 (ICR8); 0x055 (ICR9)							

Figure 9-2. Interrupt Control Registers (ICR0–ICR9)

Table 9-3 describes ICR fields.

Table 9-3. ICR*n* Field Descriptions

Bits	Field	Description
7	AVEC	Autovector enable. Determines whether the interrupt-acknowledge cycle input (for the internal interrupt level indicated in IL for each interrupt) requires an autovector response. 0 Interrupting source returns vector during interrupt-acknowledge cycle. 1 SIM generates autovector during interrupt acknowledge cycle.
6–5	—	Reserved, should be cleared.
4–2	IL	Interrupt level. Indicates the interrupt level assigned to each interrupt input. See Table 9-4.
1–0	IP	Interrupt priority. Indicates the interrupt priority for internal modules within the interrupt-level assignment. See Table 9-4. 00 Lowest 01 Low 10 High 11 Highest

NOTE:

Assigning the same interrupt level and priority to multiple ICRs causes unpredictable system behavior.

Table 9-4 shows possible priority schemes for internal and external sources of the MCF5307. The internal module interrupt source in this table can be any internal interrupt source programmed to the given level and priority.

This table shows how external interrupts are prioritized with respect to internal interrupt sources within the same level. For example, UART0 and UART1 sources are programmed to IL = 110; in this case, UART0 is given lower priority than UART1, so ICR4[IP] = 01 and the ICR5[IP] = 10. $\overline{\text{IRQ3}}$ is programmed to level 6. If all three assert an interrupt request at the same time, they are serviced in the following order:

1. ICR5[IL] = 110 and ICR5[IP] = 10, so UART1 is serviced first (priority 7 in Table 9-4).
2. External interrupt IRQ3, set to level 6, is serviced next (priority 8).
3. ICR4[IL] = 110 and ICR5[IP] = 01, so UART0 is serviced last (priority 9).

Table 9-4. Interrupt Priority Scheme

Priority	Interrupt Level	ICR		Interrupt Source	IRQPAR[IRQPAR]
		IL	IP		
1	7	111	11	Internal module	xxx
2		111	10		xxx
3		xxx	xx	External interrupt pin IRQ7	xxx
4		111	01	Internal module	xxx
5		111	00		xxx
6	6	110	11	Internal module	xxx
7		110	10		xxx
8		xxx	xx	External interrupt pin IRQ3 (programmed as IRQ6)	x1x
9		110	01	Internal module	xxx
10		110	00		xxx
11	5	101	11	Internal module	xxx
12		101	10		xxx
13		xxx	xx	External interrupt pin IRQ5	0xx
14		101	01	Internal module	xxx
15		101	00		xxx

Table 9-4. Interrupt Priority Scheme (Continued)

Priority	Interrupt Level	ICR		Interrupt Source	IRQPAR[IRQPAR]
		IL	IP		
16	4	100	11	Internal module	xxx
17		100	10		xxx
18		xxx	xx	External interrupt pin IRQ5 (programmed as IRQ4)	1xx
19		100	01	Internal module	xxx
20		100	00		xxx
21	3	011	11	Internal module	xxx
22		011	10		xxx
23		xxx	xx	External interrupt pin IRQ3	x0x
24		011	01	Internal module	xxx
25		011	00		xxx
26	2	010	11	Internal module	xxx
27		010	10		xxx
28		xxx	xx	External interrupt pin IRQ1 (programmed as IRQ2)	xx1
29		010	01	Internal module	xxx
30		010	00		xxx
31	1	001	11	Internal module	xxx
32		001	10		xxx
33		xxx	xx	External interrupt pin IRQ1	xx0
34		001	01	Internal module	xxx
35		001	00		xxx

9.2.2 Autovector Register (AVR)

The autovector register (AVR), shown in Figure 9-3, enables external interrupt sources to be autovectored, using the vector offset defined in Table 2-19 in Section 2.8, “Exception Processing Overview.” Note that the autovector enable for internal interrupt sources applies for respective ICRs.

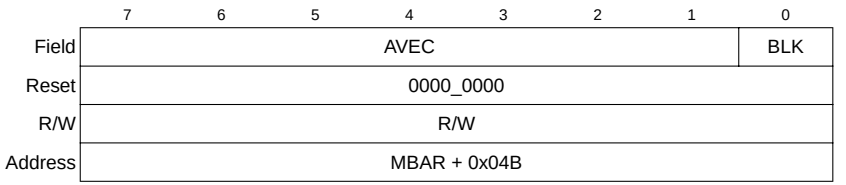


Figure 9-3. Autovector Register (AVR)

Table 9-5 describes AVR fields.

Table 9-5. AVR Field Descriptions

Bit	Name	Description
7–1	AVEC	Autovector control. Determines whether the external interrupt at that level is autovectored. 0 Interrupting source returns vector during interrupt-acknowledge cycle. 1 SIM generates autovector during interrupt-acknowledge cycle.
0	BLK	Block address strobe ($\overline{AS}$) for external AVEC access. Available for users who use $\overline{AS}$ as a global chip select for peripherals and do not want to enable them during an AVEC cycle. 0 Do not block address strobe. 1 Block address strobe from asserting.

Table 9-6 shows the correlation between AVR[AVEC] and the external interrupts. Note that an AVEC*n* bit is valid only when the corresponding external interrupt request level is enabled in the IRQPAR.

Table 9-6. Autovector Register Bit Assignments

Autovector Interrupt Source	Autovector Register Bit Location	Vector Offset
External interrupt request 1	AVEC1	0x64
External interrupt request 2	AVEC2	0x68
External interrupt request 3	AVEC3	0x6C
External interrupt request 4	AVEC4	0x70
External interrupt request 5	AVEC5	0x74
External interrupt request 6	AVEC6	0x78
External interrupt request 7	AVEC7	0x7C

9.2.3 Interrupt Pending and Mask Registers (IPR and IMR)

The interrupt pending register (IPR), Figure 9-4, makes visible the interrupt sources that have an interrupt pending. The interrupt mask register (IMR), also shown in Figure 9-4, is used to mask the internal and external interrupt sources.

NOTE:

To mask interrupt sources, first set the core's status register interrupt mask level to that of the source being masked in the IMR. Then, the IMR bit can be masked.

An interrupt is masked by setting, and enabled by clearing, the corresponding IMR bit. When a masked interrupt occurs, the corresponding IPR bit is still set, but no interrupt request is passed to the core.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—														DMA3	DMA2
Reset	—														1	1
R/W	Read-only (IPR); R/W (IMR)															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	DMA1	DMA0	UART1	UART0	I2C	TIMER2	TIMER1	SWT	EINT7	EINT6	EINT5	EINT4	EINT3	EINT2	EINT1	—
Reset	1111				1111				1111				1	1	1	—
R/W	Read-only (IPR); R/W (IMR)															
Addr	MBAR + 0x040 (IPR); + 0x044 (IMR)															

Figure 9-4. Interrupt Pending Register (IPR) and Interrupt Mask Register (IMR)

Table 9-7 describes IPR and IMR fields.

Table 9-7. IPR and IMR Field Descriptions

Bits	Name	Description
31–18	—	Reserved, should be cleared.
17–1	See Figure 9-4	Interrupt pending/mask. Each bit corresponds to an interrupt source defined by the ICR. The corresponding IMR bit determines whether an interrupt condition can generate an interrupt. At every clock, the IPR samples the signal generated by the interrupting source. The corresponding IPR bit reflects the state of the interrupt signal even if the corresponding IMR bit is set. 0 The corresponding interrupt source is not masked (IMR) and has no interrupt pending (IPR). 1 The corresponding interrupt source is masked (IMR) and has an interrupt pending (IPR)

9.2.4 Interrupt Port Assignment Register (IRQPAR)

The interrupt port assignment register (IRQPAR), shown in Figure 9-5, provides the level assignment of the primary external interrupt pins—IRQ5, IRQ3, and IRQ1. The setting of IRQPAR2–IRQPAR0 determines the interrupt level of these external interrupt pins.

	7	6	5	4	3	2	1	0
Field	IRQPAR2	IRQPAR1	IRQPAR0	—				
Reset	0000_0000							
R/W	R/W							
Address	MBAR + 0x06							

Figure 9-5. Interrupt Port Assignment Register (IRQPAR)

Table 9-8 describes IRQPAR fields.

Table 9-8. IRQPAR Field Descriptions

Bits	Name	Description
7–5	IRQPARn	Configures the IRQ pin assignments and priorities IRQPARn — External Pin — IRQPARn = 0 — IRQPARn = 1 IRQPAR2 — IRQ5 — Level 5 — Level 4 IRQPAR1 — IRQ3 — Level 3 — Level 6 IRQPAR0 — IRQ1 — Level 1 — Level 2
4–0	—	Reserved, should be cleared.

Chapter 10
Chip-Select Module

This chapter describes the MCF5307 chip-select module, including the operation and programming model of the chip-select registers, which include the chip-select address, mask, and control registers.

10.1 Overview

The following list summarizes the key chip-select features:

- Eight independent, user-programmable chip-select signals ($\overline{CS}$[7:0]) that can interface with SRAM, PROM, EPROM, EEPROM, Flash, and peripherals
- Address masking for 64-Kbyte to 4-Gbyte memory block sizes
- Programmable wait states and port sizes
- External master access to chip selects

10.2 Chip-Select Module Signals

Table 10-1 lists signals used by the chip-select module.

Table 10-1. Chip-Select Module Signals

Signal	Description
Chip Selects ($\overline{CS}$[7:0])	Each $\overline{CSn}$ 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 when it acts as a global chip select that allows boot ROM to be at any defined address space. Port size and termination (internal versus external) and byte enables for $\overline{CS0}$ are configured by the logic levels of D[7:5] when $\overline{RSTI}$ negates.
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 Enables/ Byte Write Enables ($\overline{BE}$[3:0]/ $\overline{BWE}$[3:0])	These multiplexed signals are individually programmed through the byte enable mode bit, CSCRn[BEM], described in Section 10.4.1.3, “Chip-Select Control Registers (CSCR0–CSCR7).” 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 10-2 shows.

Table 10-2 shows the interaction of the byte enable/byte-write enables with related signals.

Table 10-2. Byte Enables/Byte Write Enable Signal Settings

Transfer Size	Port Size	A1	A0	$\overline{\text{BE0/BWE0}}$	$\overline{\text{BE1/BWE1}}$	$\overline{\text{BE2/BWE2}}$	$\overline{\text{BE3/BWE3}}$
				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

10.3 Chip-Select Operation

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

- Chip-select address registers (CSAR*n*) control the base address space of the chip select. See Section 10.4.1.1, “Chip-Select Address Registers (CSAR0–CSAR7).”

- Chip-select mask registers (CSMR*n*) provide 16-bit address masking and access control. See Section 10.4.1.2, “Chip-Select Mask Registers (CSMR0–CSMR7).”
- Chip-select control registers (CSCR*n*) provide port size and burst capability indication, wait-state generation, and automatic acknowledge generation features. See Section 10.4.1.3, “Chip-Select Control Registers (CSCR0–CSCR7).”

Each $\overline{\text{CS}n}$ can assert during specific CPU space accesses such as interrupt-acknowledge cycles and each can be accessed by an external master. $\overline{\text{CS0}}$ is a global chip select after reset and provides relocatable boot ROM capability.

10.3.1 General Chip-Select Operation

When a bus cycle is initiated, the MCF5307 first compares its address 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 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 CSCR or DACR, the MCF5307 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 MCF5307 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 10-3 shows the type of access as a function of match in the CSCRs and DACRs.

Table 10-3. 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

10.3.1.1 8-, 16-, and 32-Bit Port Sizing

Static bus sizing is programmable through the port size bits, CSCR[PS]. See Section 10.4.1.3, “Chip-Select Control Registers (CSCR0–CSCR7).” Figure 10-1 shows the correspondence between data byte lanes and the external chip-select memory. Note that all lanes are driven, although unused lines are undefined.

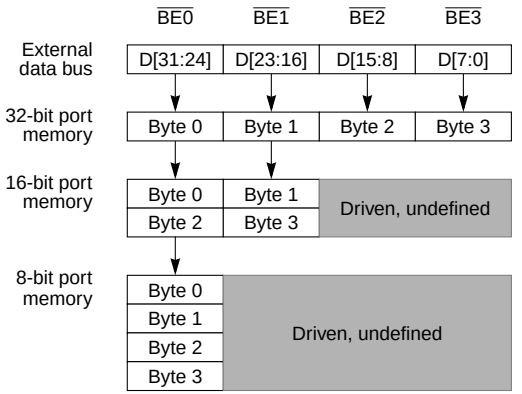


Figure 10-1. Connections for External Memory Port Sizes

10.3.1.2 Global Chip-Select Operation

$\overline{CS0}$, the global (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 $\overline{CS}$[7:1] can be used. At reset, the port size and automatic acknowledge functions of the global chip-select are determined by the logic levels of the inputs on D[7:5]. Table 10-4 and Table 10-5list the various reset encodings for the configuration signals multiplexed with D[7:5].

Table 10-4. D7/AA, Automatic Acknowledge of Boot $\overline{CS0}$

D7/AA	Boot $\overline{CS0}$ AA Configuration at Reset
0	Disabled
1	Enable with 15 wait states

Provided the required address range is in the chip-select address register (CSAR0), $\overline{CS0}$ can

Table 10-5. D[6:5]/PS[1:0], Port Size of Boot $\overline{CS0}$

D[6:5]/PS[1:0]	Boot $\overline{CS0}$ Port Size at Reset
00	32-bit port
01	8-bit port
1x	16-bit port

be programmed to continue decoding for a range of addresses after the CSMR0[V] is set, after which the global chip-select can be restored only by a system reset.

10.4 Chip-Select Registers

Table 10-6Table 10-6 is the chip-select register memory map. Reading reserved locations returns zeros.

Table 10-6. Chip-Select Registers

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x080	Chip-select address register—bank 0 (CSAR0) [p. 10-6]		Reserved[1]	
0x084	Chip-select mask register—bank 0 (CSMR0) [p. 10-6]			
0x088	Reserved[1]		Chip-select control register—bank 0 (CSCR0) [p. 10-8]	
0x08C	Chip-select address register—bank 1 (CSAR1) [p. 10-6]		Reserved[1]	
0x090	Chip-select mask register—bank 1 (CSMR1) [p. 10-6]			
0x094	Reserved[1]		Chip-select control register—bank 1 (CSCR1) [p. 10-8]	
0x098	Chip-select address register—bank 2 (CSAR2) [p. 10-6]		Reserved[1]	
0x09C	Chip-select mask register—bank 2 (CSMR2) [p. 10-6]			
0x0A0	Reserved[1]		Chip-select control register—bank 2 (CSCR2) [p. 10-8]	
0x0A4	Chip-select address register—bank 3 (CSAR3) [p. 10-6]		Reserved[1]	
0x0A8	Chip-select mask register—bank 3 (CSMR3) [p. 10-6]			
0x0AC	Reserved[1]		Chip-select control register—bank 3 (CSCR3) [p. 10-8]	
0x0B0	Chip-select address register—bank 4 (CSAR4) [p. 10-6]		Reserved[1]	
0x0B4	Chip-select mask register—bank 4 (CSMR4) [p. 10-6]			
0x0B8	Reserved[1]		Chip-select control register—bank 4 (CSCR4) [p. 10-8]	
0x0BC	Chip-select address register—bank 5 (CSAR5) [p. 10-6]		Reserved[1]	
0x0C0	Chip-select mask register—bank 5 (CSMR5) [p. 10-6]			
0x0C4	Reserved		Chip-select control register—bank 5 (CSCR5) [p. 10-8]	
0x0C8	Chip-select address register—bank 6 (CSAR6) [p. 10-6]		Reserved[1]	
0x0CC	Chip-select mask register—bank 6 (CSMR6) [p. 10-6]			
0x0D0	Reserved[1]		Chip-select control register—bank 6 (CSCR6) [p. 10-8]	
0x0D4	Chip-select address register—bank 7 (CSAR7) [p. 10-6]		Reserved[1]	

Table 10-6. Chip-Select Registers (Continued)

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x0D8	Chip-select mask register—bank 7 (CSMR7) [p. 10-6]			
0x0DC	Reserved[1]		Chip-select control register—bank 7 (CSCR7) [p. 10-8]	

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.

NOTE:

External masters cannot access MCF5307 on-chip memories or MBAR, but can access any of the chip-select module registers.

10.4.1 Chip-Select Module Registers

The chip-select module is programmed through the chip select address registers (CSAR0–CSAR7), chip select mask registers (CSMR0–CSMR7), and the chip select control registers (CSCR0–CSCR7).

10.4.1.1 Chip-Select Address Registers (CSAR0–CSAR7)

Chip select address registers, Figure 10-2, specify the chip select base addresses.

	15 … 0
Field	BA
Reset	Uninitialized
R/W	R/W
Addr	0x080 (CSAR0); 0x08C (CSAR1); 0x098 (CSAR2); 0x0A4 (CSAR3); 0x0B0 (CSAR4); 0x0BC (CSAR5); 0x0C8 (CSAR6); 0x0D4 (CSAR7)

Figure 10-2. Chip Select Address Registers (CSAR0–CSAR7)

Table 10-7 describes CSAR[BA].

Table 10-7. CSAR*n* Field Description

Bits	Name	Description
15–0	BA	Base address. Defines the base address for memory dedicated to chip select $\overline{CS}$[7:0]. BA is compared to bits 31–16 on the internal address bus to determine if chip-select memory is being accessed.

10.4.1.2 Chip-Select Mask Registers (CSMR0–CSMR7)

The chip select mask registers, Figure 10-3, 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); 0x0D8 (CSMR7)										

Figure 10-3. Chip Select Mask Registers (CSMR*n*)

Table 10-8 describes CSMR fields.

Table 10-8. 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 $\overline{CS}$[7:0] is 2^n; n = (number of bits set in respective CSMR[BAM]) + 16. So, if CSAR0 = 0x0000 and CSMR0[BAM] = 0x0008, $\overline{CS0}$ would address two discontinuous 64-Kbyte memory blocks: one from 0x0000–0xFFFF and one from 0x8_0000–0x8_FFFF. Likewise, for $\overline{CS0}$ to access 32 Mbytes of address space starting at location 0x0, $\overline{CS1}$ must begin at the next byte after $\overline{CS0}$ for a 16-Mbyte address 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 CSARn[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 an external master or 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 if AM = 0, SC, SD, UC, and UD are ignored in the chip select decode on external master or 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 CSMRn[V]. 0 Chip select invalid 1 Chip select valid

10.4.1.3 Chip-Select Control Registers (CSCR0–CSCR7)

Each chip-select control register, Figure 10-4, controls the auto acknowledge, external master support, port size, burst capability, and activation of each chip select. Note that to support the global 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	—	D7	D6	D5	—	—		
Reset: Other CSCRs	Unitialized									
R/W	R/W									
Address	0x08A (CSCR0); 0x096 (CSCR1); 0x0A2 (CSCR2); 0x0AE (CSCR3); 0x0BA (CSCR4); 0x0C6 (CSCR5); 0x0D2 (CSCR6); 0x0DE (CSCR7)									

Figure 10-4. Chip-Select Control Registers (CSCR0–CSCR7)

Table 10-9 describes CSCR*n* fields.

Table 10-9. 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 $\overline{TA}$ supersedes 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 10.3.1.1, “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 Neither $\overline{BE}$ nor $\overline{BWE}$ is asserted for read. $\overline{BWE}$ is generated for data write only. 1 $\overline{BE}$ is asserted for read; $\overline{BWE}$ is asserted for write.
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 10-9. CSCR*n* Field Descriptions

Bits	Name	Description
3	BSTW	Burst write enable. Specifies whether burst writes are used for memory associated with each $\overline{\text{CS}}$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.

10.4.1.4 Code Example

The code below provides an example of how to initialize the chip-selects. Only chip selects 0, 1, 2, and 3 are programmed here; chip selects 4, 5, 6, and 7 are left invalid. MBARx defines the base of the module address space.

```
CSAR0 EQU MBARx+0x080   ;Chip select 0 address register
CSMR0 EQU MBARx+0x084   ;Chip select 0 mask register
CSCR0 EQU MBARx+0x08A   ;Chip select 0 control register

CSAR1 EQU MBARx+0x08C   ;Chip select 1 address register
CSMR1 EQU MBARx+0x090   ;Chip select 1 mask register
CSCR1 EQU MBARx+0x096   ;Chip select 1 control register

CSAR2 EQU MBARx+0x098   ;Chip select 2 address register
CSMR2 EQU MBARx+0x09C   ;Chip select 2 mask register
CSCR2 EQU MBARx+0x0A2   ;Chip select 2 control register

CSAR3 EQU MBARx+0x0A4   ;Chip select 3 address register
CSMR3 EQU MBARx+0x0A8   ;Chip select 3 mask register
CSCR3 EQU MBARx+0x0AE   ;Chip select 3 control register

CSAR4 EQU MBARx+0x0B0   ;Chip select 4 address register
CSAR4 EQU MBARx+0x0B4   ;Chip select 4 mask register
CSMR4 EQU MBARx+0x0BA   ;Chip select 4 control register

CSAR5 EQU MBARx+0x0BC   ;Chip select 5 address register
CSMR5 EQU MBARx+0x0C0   ;Chip select 5 mask register
CSCR5 EQU MBARx+0x0C6   ;Chip select 5 control register

CSAR6 EQU MBARx+0x0C8   ;Chip select 6 address register
CSMR6 EQU MBARx+0x0CC   ;Chip select 6 mask register
CSCR6 EQU MBARx+0x0D2   ;Chip select 6 control register

CSAR7 EQU MBARx+0x0D4   ;Chip select 7 address register
CSMR7 EQU MBARx+0x0D8   ;Chip select 7 mask register
CSCR7 EQU MBARx+0x0DE   ;Chip select 7 control register

; All other chip selects should be programmed and made valid before global
; chip select is de-activated by validating CS0

; Program Chip Select 3 Registers
move.w  #0x0040,D0      ;CSAR3 base address 0x00400000
move.w  D0,CSAR3

move.w  #0x00A0,D0      ;CSCR3 = no wait states, AA=0, PS=16-bit, BEM=1,
move.w  D0,CSCR3        ;BSTR=0, BSTW=0

move.l #0x001F016B,D0   ;Address range from 0x00400000 to 0x005FFFFF
move.l D0,CSMR3         ;WP,EM,C/I,SD,UD,V=1; SC,UC=0

; Program Chip Select 2 Registers

move.w  #0x0020,D0      ;CSAR2 base address 0x00200000 (to 0x003FFFFF)
```

```
move.w  D0,CSAR2

move.w  #0x0538,D0      ;CSCR2 = 1 wait state, AA=1, PS=32-bit, BEM=1,
move.w  D0,CSCR2        ;BSTR=1, BSTW=1

move.l  #0x001F0001,D0  ;Address range from 0x00200000 to 0x003FFFFF
move.l  D0,CSMR2        ;WP,EM,C/I,SC,SD,UC,UD=0; V=1

; Program Chip Select 1 Registers

move.w  #0x0000,D0      ;CSAR1 base addresses 0x00000000 (to 0x001FFFFF)
move.w  D0,CSAR1        ;and 0x80000000 (to 0x801FFFFF)

move.w  #0x09B0,D0      ;CSCR1 = 2 wait states, AA=1, PS=16-bit, BEM=1,
move.w  D0,CSCR1        ;BSTR=1, BSTW=0

move.l  #0x801F0001,D0  ;Address range from 0x00000000 to 0x001FFFFF and
move.l  D0,CSMR1        ;0x80000000 to 0x801FFFFF
                        ;WP, EM, C/I, SC, SD, UC, UD=0, V=1

; Program Chip Select 0 Registers

move.w  #0x0080,D0      ;CSAR0 base address 0x00800000 (to 0x009FFFFF)
move.w  D0,CSAR0

move.w  #0x0D80,D0      ;CSCR0 = three wait states, AA=1, PS=16-bit, BEM=0,
move.w  D0,CSCR0        ;BSTR=0, BSTW=0

; Program Chip Select 0 Mask Register (validate chip selects)

move.l  #0x001F0001,D0  ;Address range from 0x00800000 to 0x009FFFFF
move.l  D0,CSMR0        ;WP,EM,C/I,SC,SD,UC,UD=0; V=1
```

Chapter 11
Synchronous/Asynchronous DRAM Controller Module

This chapter describes configuration and operation of the synchronous/asynchronous DRAM controller component of the system integration module (SIM). It begins with a general description and brief glossary, and includes a description of signals involved in DRAM operations. The remainder of the chapter consists of the two following parts:

- Section 11.3, “Asynchronous Operation,” describes the programming model and signal timing for the four basic asynchronous modes.
  - — Non-page mode
  - — Burst page mode
  - — Continuous page mode
  - — Extended data-out mode
- Section 11.4, “Synchronous Operation,” describes the programming model and signal timing, as well as the command set required for synchronous operations. This section also includes extensive examples the designer can follow to better understand how to configure the DRAM controller for synchronous operations.

11.1 Overview

The DRAM controller module provides glueless integration of DRAM with the ColdFire product. The key features of the DRAM controller include the following:

- Support for two independent blocks of DRAM
- Interface to standard synchronous/asynchronous dynamic random access memory (ADRAM/SDRAM) components
- Programmable $\overline{\text{SRAS}}$, $\overline{\text{SCAS}}$, and refresh timing
- Support for page mode
- Support for 8-, 16-, and 32-bit wide DRAM blocks
- Support for synchronous and asynchronous DRAMs, including EDO DRAM, SDRAM, and fast page mode

11.1.1 Definitions

The following terminology is used in this chapter:

- A/SDRAM block—Any group of DRAM memories selected by one of the MCF5307 $\overline{\text{RAS}}$[1:0] signals. Thus, the MCF5307 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 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.

11.1.2 Block Diagram and Major Components

The basic components of the DRAM controller are shown in Figure 11-1.

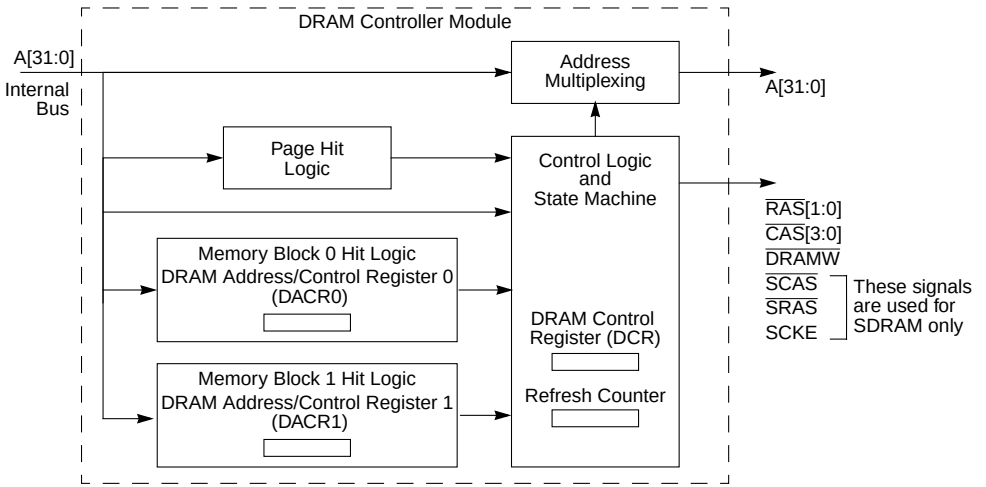


Figure 11-1. Asynchronous/Synchronous DRAM Controller Block Diagram

The DRAM controller's major components, shown in Figure 11-1, are described 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 MBAR + 0x0108; DACR1 is accessed at 0x010. The register information is passed on to the hit logic.

- Control logic and state machine—Generates all DRAM signals, taking bus cycle characteristic data from the block logic, along with hit information to generate DRAM 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, determined 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 DRAM bus cycle to both DACRs to determine if a DRAM block is being accessed. Hits are passed to the control logic along with characteristics of the bus cycle to be generated.
- Page hit logic—Determines if the next DRAM access is in the same DRAM page as the previous one. This information is passed on to the control logic.
- Address multiplexing—Multiplexes addresses to allow column and row addresses to share pins. This allows glueless interface to DRAMs.

11.2 DRAM Controller Operation

The DRAM controller mode is programmed through DCR[SO]. Asynchronous mode (SO = 0) includes support for page mode and EDO DRAMs. Synchronous mode is designed to work with industry-standard SDRAMs. These modes act very differently from one another, especially regarding the use of DRAM registers and pins. Memory blocks cannot operate in different modes; both are either synchronous or asynchronous.

11.2.1 DRAM Controller Registers

The DRAM controller registers memory map, Table 11-1, is the same regardless of whether asynchronous or synchronous DRAM is used, although bit configurations may vary.

Table 11-1. DRAM Controller Registers

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x100	DRAM control register (DCR) [p. 11-4]		Reserved	
0x104	Reserved			
0x108	DRAM address and control register 0 (DACR0) [p. 11-5]			
0x10C	DRAM mask register block 0 (DMR0) [p. 11-7]			
0x110	DRAM address and control register 1 (DACR1) [p. 11-5]			
0x114	DRAM mask register block 1 (DMR1) [p. 11-7]			

NOTE:

External masters cannot access MCF5307 on-chip memories or MBAR, but they can access DRAM controller registers.

11.3 Asynchronous Operation

The DRAM controller supports asynchronous DRAMs for cost-effective systems. Typical access times for the DRAM controller interfacing to ADRAM are 4-3-3-3. The DRAM controller supports the following four asynchronous modes:

- Non-page mode
- Burst page mode
- Continuous page mode
- Extended data-out mode

In asynchronous mode, $\overline{RAS}$ and $\overline{CAS}$ always transition at the falling clock edge. As summarized previously, operation and timing of each ADRAM block is controlled by separate registers, but refresh is the same for both. All ADRAM accesses should be terminated by the DRAM controller. There is no priority encoding between memory blocks, so programming blocks to overlap with other blocks or with other internal resources causes undefined behavior.

11.3.1 DRAM Controller Signals in Asynchronous Mode

Table 11-2 summarizes DRAM signals used in asynchronous mode.

Table 11-2. SDRAM Signal Summary

Signal	Description
$\overline{RAS}$[1:0]	Row address strobes. Interface to $\overline{RAS}$ inputs on industry-standard ADRAMs. When SDRAMs are used, these signals interface to the chip-select lines within an SDRAM's memory block. Thus, there is one $\overline{RAS}$ line for each of the two blocks.
$\overline{CAS}$[3:0]	Column address strobes. Interface to $\overline{CAS}$ inputs on industry-standard DRAMs. These provide $\overline{CAS}$ for a given ADRAM block. When SDRAMs are used, $\overline{CAS}$[3:0] control the byte enables (DQMx) for standard SDRAMs. $\overline{CAS}$[3:0] strobes data in least-to-most significant byte order ($\overline{CAS0}$ is MSB).
$\overline{DRAMW}$	DRAM read/write. Asserted when a DRAM write cycle is underway. Negated for read bus cycles.

11.3.2 Asynchronous Register Set

The following register configurations apply when DCR[SO] is 0, indicating the DRAM controller is interfacing to asynchronous DRAMs.

11.3.2.1 DRAM Control Register (DCR) in Asynchronous Mode

The DCR provides programmable options for the refresh logic as well as the control bit to determine if the module is operating with synchronous or asynchronous DRAMs. The DCR is shown in Figure 11-2.

	15	14	13	12	11	10	9	8	0
Field	SO	—	NAM	RRA		RRP		RC	
Reset	0	Uninitialized							
R/W	R/W								
Address	MBAR + 0x100								

Figure 11-2. DRAM Control Register (DCR) (Asynchronous Mode)

Table 11-3 describes DCR fields.

Table 11-3. DCR Field Descriptions (Asynchronous Mode)

Bits	Name	Description
15	SO	Synchronous operation. Selects synchronous or asynchronous mode. A DRAM controller in synchronous mode can be switched to ADRAM mode only by resetting the MCF5307. 0 Asynchronous DRAMs. Default at reset. 1 Synchronous DRAMs
14	—	Reserved, should be cleared.
13	NAM	No address multiplexing. Some implementations require external multiplexing. For example, when linear addressing is required, the DRAM should not multiplex addresses on DRAM accesses. 0 The DRAM controller multiplexes the external address bus to provide column addresses. 1 The DRAM controller does not multiplex the external address bus to provide column addresses.
12–11	RRA	Refresh $\overline{\text{RAS}}$ asserted. Determines how long $\overline{\text{RAS}}$ is asserted during a refresh operation. 00 2 clocks 01 3 clocks 10 4 clocks 11 5 clocks
10–9	RRP	Refresh $\overline{\text{RAS}}$ precharge. Controls how many clocks $\overline{\text{RAS}}$ is precharged after a refresh operation before accesses are allowed to DRAM. 00 1 clock 01 2 clocks 10 3 clocks 11 4 clocks
8–0	RC	Refresh count. Controls refresh frequency. The number of bus clocks between refresh cycles is (RC + 1) * 16. Refresh can range from 16–8192 bus clocks to accommodate both standard and low-power DRAMs 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 (625 bus clocks at 40 MHz). # of bus clocks = 625 = (RC field + 1) * 16 RC = (625 bus clocks/16) -1 = 38.06, which rounds to 38; therefore, RC = 0x26.

11.3.2.2 DRAM Address and Control Registers (DACR0/DACR1)

DACR0 and DACR1, Figure 11-3, contain the base address compare value and the control bits for memory blocks 0 and 1. Address and timing are also controlled by these registers. Memory areas defined for each block should not overlap; operation is undefined for accesses in overlapping regions.

	31 18	17 16	15	14	13 12	11 10	9	8	7	6	5 4	3 2	1 0
Field	BA	—	RE	—	CAS	RP	RNCN	RCD	—	EDO	PS	PM	—
Reset	Unitialized		0	Unitialized									
R/W	R/W												
Addr	MBAR + 0x10C (DACR0); 0x110 (DACR1)												

Figure 11-3. DRAM Address and Control Registers (DACR0/DACR1)

Table 11-4 describes DACR*n* fields.

Table 11-4. DACR0/DACR1 Field Description

Bits	Name	Description
31–18	BA	Base address. Used with DMR[BAM] to determine the address range in which the associated DRAM block is located. Each BA bit is compared with the corresponding address of the bus cycle in progress. If each bit matches, or if bits that do not match are masked in the BAM, the address selects the associated DRAM block.
17–16	—	Reserved, should be cleared.
15	RE	Refresh enable. Determines whether the DRAM controller generates a refresh to the associated DRAM block. DRAM contents are not preserved during hard reset or software watchdog reset. 0 Do not refresh associated DRAM block. (Default at reset) 1 Refresh associated DRAM block.
14	—	Reserved, should be cleared.
13–12	CAS	$\overline{CAS}$ timing. Determines how long $\overline{CAS}$ is asserted during a DRAM access. 00 1 clock cycle 01 2 clock cycles 10 3 clock cycles 11 4 clock cycles
11–10	RP	$\overline{RAS}$ precharge timing. Determines how long $\overline{RAS}$ is precharged between accesses. Note that RP is different from DCR[RRP]. 00 1 clock cycle 01 2 clock cycles 10 3 clock cycles 11 4 clock cycles
9	RNCN	$\overline{RAS}$-negate-to-$\overline{CAS}$-negate. Controls whether $\overline{RAS}$ and $\overline{CAS}$ negate concurrently or one clock apart. RNCN is ignored if $\overline{CAS}$ is asserted for only one clock and both $\overline{RAS}$ and $\overline{CAS}$ are negated. RNCN is used only for non-page-mode accesses and single accesses in page mode. 0 $\overline{RAS}$ negates concurrently with $\overline{CAS}$. 1 $\overline{RAS}$ negates one clock before $\overline{CAS}$.
8	RCD	$\overline{RAS}$-to-$\overline{CAS}$ delay. Determines the number of system clocks between assertions of $\overline{RAS}$ and $\overline{CAS}$. 0 1 clock cycle 1 2 clock cycles
7	—	Reserved, should be cleared.
6	EDO	Extended data out. Determines whether the DRAM block operates in a mode to take advantage of industry-standard EDO DRAMs. Do not use EDO mode with non-EDO DRAM. 0 EDO operation disabled. 1 EDO operation enabled.

Table 11-4. DACR0/DACR1 Field Description (Continued)

Bits	Name	Description
5–4	PS	Port size. Determines the port size of the associated DRAM block. For example, if two 16-bit wide DRAM components form one DRAM block, the port size is 32 bits. Programming PS allows the DRAM controller to execute dynamic bus sizing for associated accesses. 00 32-bit port 01 8-bit port 1x 16-bit port
3–2	PM	Page mode. Configures page-mode operation for the memory block. 00 No page mode 01 Burst page mode (page mode for bursts only) 10 Reserved 11 Continuous page mode
1–0	—	Reserved, should be cleared.

11.3.2.3 DRAM Controller Mask Registers (DMR0/DMR1)

The DRAM controller mask registers (DMR0 and DMR1), shown in Figure 11-4, include mask bits for the base address and for address attributes.

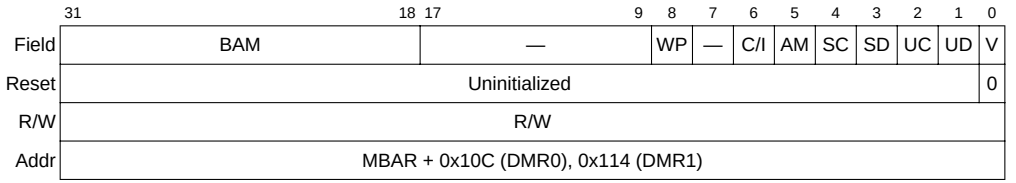


Figure 11-4. DRAM Controller Mask Registers (DMR0 and DMR1)

Table 11-5 describes DMR*n* fields.

Table 11-5. DMR0/DMR1 Field Descriptions

Bits	Name	Description
31–18	BAM	Base address mask. Masks the associated DACRn[BA]. Lets the DRAM controller connect to various DRAM sizes. Mask bits need not be contiguous (see Section 11.5, “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.

Table 11-5. DMR0/DMR1 Field Descriptions (Continued)

Bits	Name	Description
6–1	AMx	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	External or 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

11.3.3 General Asynchronous Operation Guidelines

The DRAM controller provides control for $\overline{\text{RAS}}$, $\overline{\text{CAS}}$, and $\overline{\text{DRAMW}}$ signals, as well as address multiplexing and bus cycle termination. Whether the mode is synchronous or asynchronous determines signal control and termination. To reduce complexity, multiplexing is the same for both modes. Table 11-6 shows the scheme for DRAM configurations. This scheme works for symmetric configurations (in which the number of rows equals the number of columns) as well as asymmetric configurations (in which the number of rows and columns are different).

Table 11-6. 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

Table 11-6. Generic Address Multiplexing Scheme (Continued)

Address Pin	Row Address	Column Address	Notes Relating to Port Sizes
20	20	19	
21	21	20	
22	22	21	
23	23	22	
24	24	23	
25	25	24	

Note the following:

- Each MCF5307 address bit drives both a row address and a column address bit.
- As the user upgrades ADRAM, corresponding MCF5307 address bits must be connected. This multiplexing scheme allows various memory widths to be connected to the address bus.
- Some differences exist for each of the three possible port sizes. Note that only 8-bit ports use an A0 address from the MCF5307. Because 16- and 32-bit ports issue either words or longwords when accessed, they do not use the MCF5307 A0 signal.

  Likewise, the configuration for 32-bit ports uses neither A0 or A1. This presents a slight problem because DRAM address signal A0 is issued on physical pin A17 of the MCF5307 along with the ADRAM address signal A17. Although A0 is not used for larger ports, A17 is still needed. The MCF5307 DRAM controller provides for this by changing the column address that appears on physical pin A17 of the processor whenever an 8-bit port is not selected. This is determined by the DACR*n*[PS] settings. For 8-bit ports, MCF5307 physical pin A17 drives logical address A0 during the CAS cycle. When 16- or 32-bit port sizes are programmed, the CAS cycle pin A17 drives logical address A16, as indicated in the generic connection scheme.
- If a 32-bit port is used with only eight column address lines, A18 must drive DRAM address bit A18. Otherwise, in 32-bit port configurations, the MCF5307 physical address line is not connected with more than eight column address lines.
- All ADRAM blocks have a fixed page size of 512 bytes for page-mode operation. The addresses are connected differently for various width combinations.

Table 11-7, Table 11-8, and Table 11-9 show how 8-, 16-, and 32-bit symmetrical ADRAM memories are connected to the address bus. The memory sizes show what DRAM size is accessed if the corresponding bits are connected to the memory. In each case, there is a base memory size. This limitation exists to allow simple page-mode multiplexing. Notice also that MCF5307 pin 17 is treated differently in byte-wide operations. In byte-wide operations, address bits 16 and 17 are driven on MCF5307 physical address pins 16 and 17, rather than the two bits being driven solely on A17, as they are for 32-wide memories.

Table 11-7. DRAM Addressing for Byte-Wide Memories

MCF5307 Address Pin	MCF5307 Address Bit Driven for RAS	MCF5307 Address Bit Driven when CAS is Asserted	Memory Size
17	17	0	Base memory size of 256 Kbytes
16	16	1	
15	15	2	
14	14	3	
13	13	4	
12	12	5	
11	11	6	
10	10	7	
9	9	8	
19	19	18	1 Mbyte
21	21	20	4 Mbytes
23	23	22	16 Mbytes
25	25	24	64 Mbytes

Note that in Table 11-8, MCF5307 pin A19 is not connected because DRAM address bit 18 is already provided on MCF5307 pin A18; thus, the next MCF5307 pin used should be A20.

Table 11-8. DRAM Addressing for 16-Bit Wide Memories

MCF5307 Address Pin	MCF5307 Address Bit Driven for RAS	MCF5307 Address Bit Driven when CAS is Asserted	Memory Size
16	16	1	Base memory size of 128 Kbytes
15	15	2	
14	14	3	
13	13	4	
12	12	5	
11	11	6	
10	10	7	
9	9	8	
18	18	17	512 Kbytes
20	20	19	2 Mbytes
22	22	21	8 Mbytes
24	24	23	32 Mbytes

Table 11-9. DRAM Addressing for 32-Bit Wide Memories

MCF5307 Address Pin	MCF5307 Address Bit Driven for $\overline{RAS}$	MCF5307 Address Bit Driven when CAS is Asserted	Memory Size
15	15	2	Base Memory Size of 64 Kbytes
14	14	3	
13	13	4	
12	12	5	
11	11	6	
10	10	7	
9	9	8	
17	17	16	256 Kbytes
19	19	18	1 Mbyte
21	21	20	4 Mbytes
23	23	22	16 Mbytes
25	25	24	64 Mbytes

11.3.3.1 Non-Page-Mode Operation

In non-page mode, the simplest mode, the DRAM controller provides termination and runs a separate bus cycle for each data transfer. Figure 11-5 shows a non-page-mode access in which a DRAM read is followed by a write. Addresses for a new bus cycle are driven at the rising clock edge.

For a DRAM block hit, the associated $\overline{RAS}$ is driven at the next falling edge. Here DACR*n*[RCD] = 0, so the address is multiplexed at the next rising edge to provide the column address. The required $\overline{CAS}$ signals are then driven at the next falling edge and remain asserted for the period programmed in DACR*n*[CAS]. Here, DACR*n*[RNCN] = 1, so it is precharged one clock before $\overline{CAS}$ is negated. On a read, data is sampled on the last rising edge of the clock that $\overline{CAS}$ is valid.

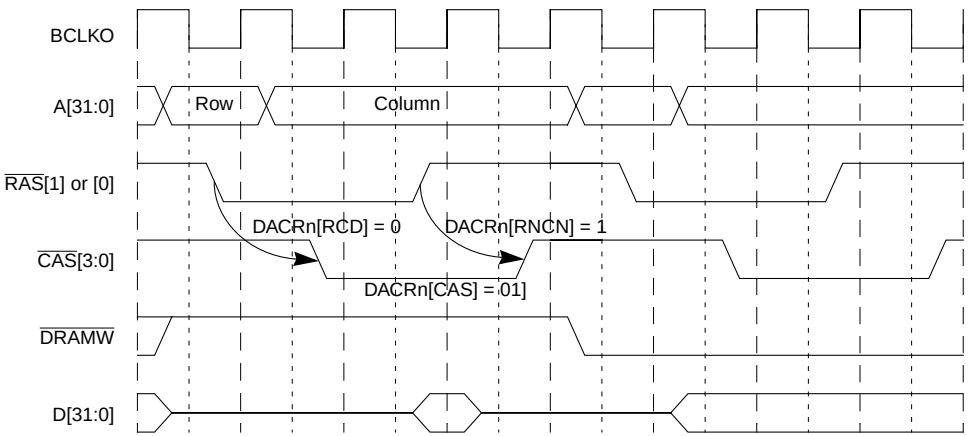


Figure 11-5. Basic Non-Page-Mode Operation RCD = 0, RNCN = 1 (4-4-4-4)

Figure 11-6 shows a variation of the basic cycle. In this case, RCD is 1, so there are two clocks between $\overline{RAS}$ and $\overline{CAS}$. Note that the address is multiplexed on the rising clock immediately before $\overline{CAS}$ is asserted. Because RNCN = 0, $\overline{RAS}$ and $\overline{CAS}$ are negated together. The next bus cycle is initiated, but because DACR*n*[RP] requires $\overline{RAS}$ to be precharged for two clocks, $\overline{RAS}$ is delayed for a clock in the bus cycle. Note that this does not delay the address signals, only $\overline{RAS}$.

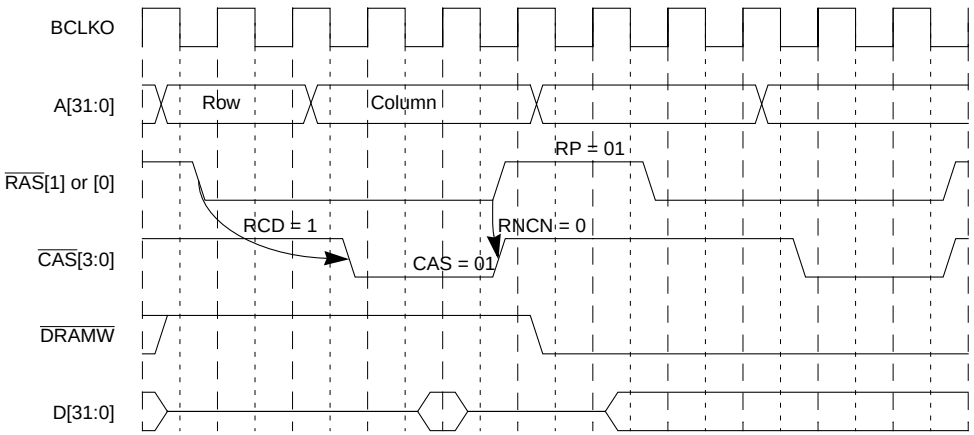


Figure 11-6. Basic Non-Page-Mode Operation RCD = 1, RNCN = 0 (5-5-5-5)

11.3.3.2 Burst Page-Mode Operation

Burst page-mode operation (DACRn[PM] = 01) optimizes memory accesses in page mode by allowing a row address to remain registered in the DRAM while accessing data in different columns. This eliminates the setup and hold times associated with the need to precharge and assert RAS. Therefore, only the first bus cycle in the page takes the full access time; subsequent accesses are streamlined. Single accesses look the same as non-page-mode accesses.

Burst page-mode accesses of any size—byte, word, longword, or line—are assumed to reside in the same page. In this mode, the DRAM controller generates a burst transfer only when the operand is larger than the DRAM block port size (such as, a line transfer to a 32-bit port or a longword transfer to an 8-bit port). The primary cycle asserts $\overline{RAS}$ and $\overline{CAS}$; subsequent cycles assert only $\overline{CAS}$. At the end of the access, $\overline{RAS}$ is precharged. The DRAM controller increments addresses between cycles.

Figure 11-7 shows a read access in burst page mode. Four accesses take place, which could be a 32-bit access to an 8-bit port or a line access to a 32-bit port. Other burst page-mode operations may be from 2 to 16 accesses long, depending on the access and port sizes. In those cases, timing is similar with more or fewer accesses.

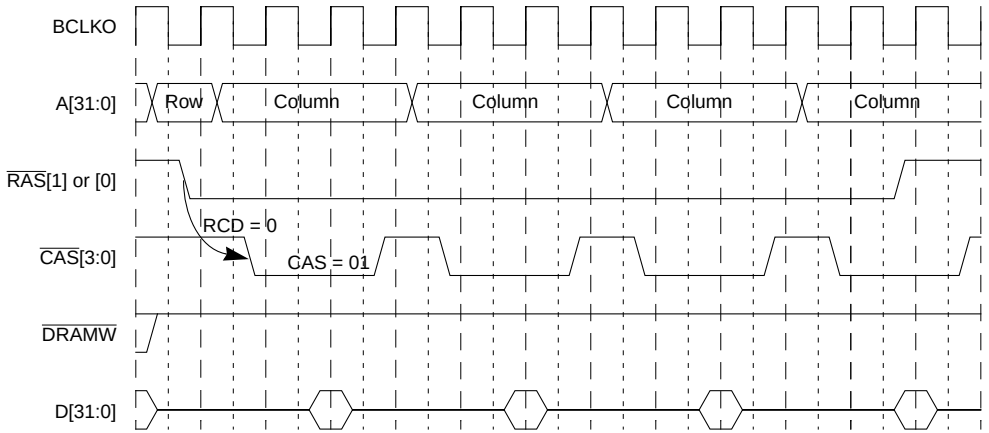


Figure 11-7. Burst Page-Mode Read Operation (4-3-3-3)

Figure 11-8 shows the write operation with the same configuration.

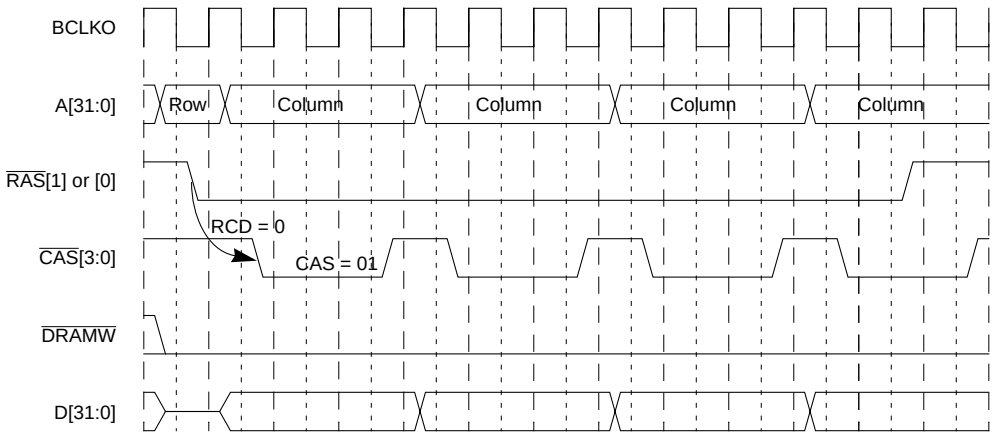


Figure 11-8. Burst Page-Mode Write Operation (4-3-3-3)

11.3.3.3 Continuous Page Mode

Continuous page mode (DACRn[PM] = 11) is a type of page mode that balances performance, complexity, and size. In typical page-mode implementations, sequential addresses are checked for multiple hits in a DRAM block. On a hit, $\overline{\text{RAS}}$ remains asserted and $\overline{\text{CAS}}$ is asserted with the new column address. On a miss, $\overline{\text{RAS}}$ must be precharged again before the bus cycle begins.

Continuous page mode supports page-mode operation without requiring an address holding register per memory block and eliminates the delay for a miss-to-precharge $\overline{\text{RAS}}$ for the upcoming bus cycle. Because the internal MCF5307 address bus is pipelined, addresses for

the next bus cycle are often available before the current cycle completes. The two addresses are compared at the end of the cycle to determine if the next address hits the same page. If so, $\overline{RAS}$ remains asserted. If not, or if no access is pending, $\overline{RAS}$ is precharged before the next bus cycle is active on the external bus. As a result, a page miss suffers no penalty. Single accesses not followed by a hit in the page look like non-page-mode accesses.

Figure 11-9 shows a write cycle followed by a read cycle in continuous page mode. The read hits in the same page as the write so $\overline{RAS}$ is not negated before the second cycle. Note that the row address does not appear on the pins for a bus cycle that hits in the page. Column addresses are immediately multiplexed onto the pins. The third bus cycle is a page miss, so $\overline{RAS}$ is precharged before the end of the bus cycle and no extra precharge delay is incurred.

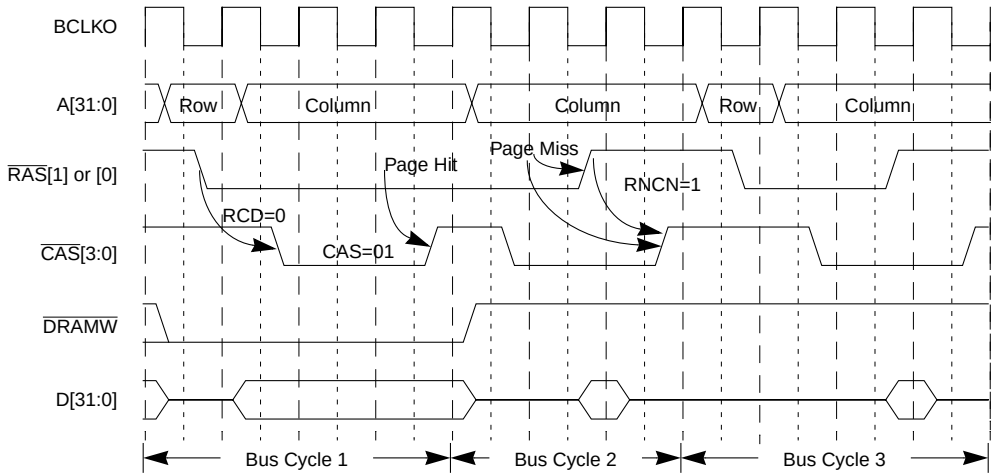


Figure 11-9. Continuous Page-Mode Operation

If a write cycle hits in the page, $\overline{CAS}$ must be delayed by one clock to allow data to become valid, as shown in Figure 11-10.

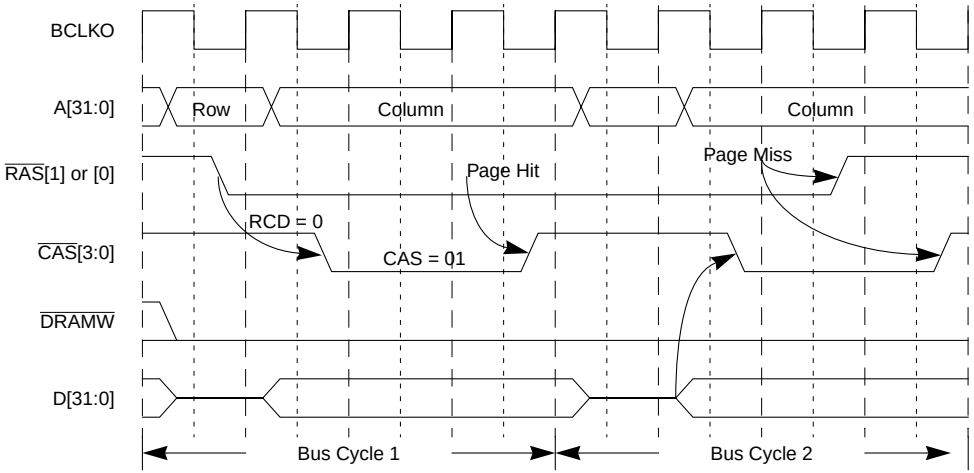


Figure 11-10. Write Hit in Continuous Page Mode

11.3.3.4 Extended Data Out (EDO) Operation

EDO is a variation of page mode that allows the DRAM to continue driving data out of the device while $\overline{CAS}$ is precharging. To support EDO DRAMs, the DRAM controller delays internal termination of the cycle by one clock so data can continue to be captured as $\overline{CAS}$ is being precharged. For data to be driven by the DRAMs, $\overline{RAS}$ is held after $\overline{CAS}$ is negated. EDO operation does not affect write operations. EDO DRAMs can be used in continuous page or burst page modes. Single accesses not followed by a hit in the page look like non-page-mode accesses.

Figure 11-11 shows four consecutive EDO accesses. Note that data is sampled after $\overline{CAS}$ is negated and that on the last page access, $\overline{CAS}$ is held until after data is sampled to assure that the data is driven. This allows $\overline{RAS}$ to be precharged before the end of the cycle.

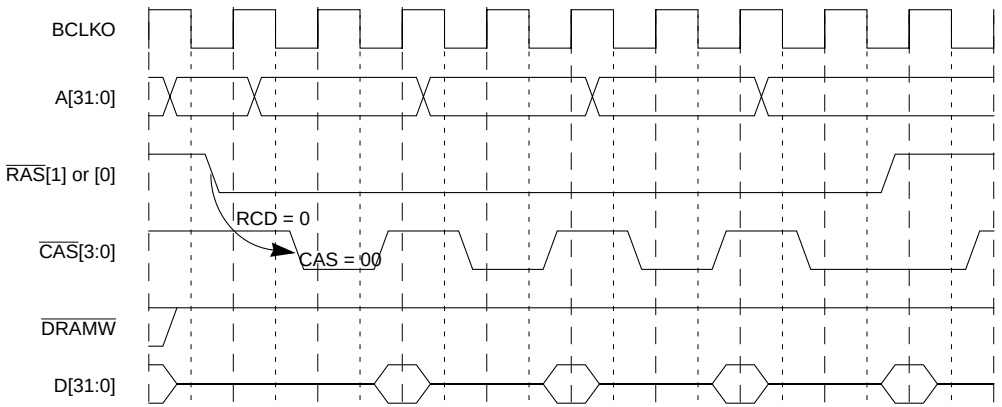


Figure 11-11. EDO Read Operation (3-2-2-2)

11.3.3.5 Refresh Operation

The DRAM controller supports $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ refresh operations that are not synchronized to bus activity. A special $\overline{\text{DRAMW}}$ pin is provided so refresh can occur regardless of the state of the processor bus.

When the refresh counter rolls over, it sets an internal flag to indicate that a refresh is pending. If that happens during a continuous page-mode access, the page is closed ($\overline{\text{RAS}}$ precharged) when the data transfer completes to allow the refresh to occur. The flag is cleared when the refresh cycle is run. Both memory blocks are simultaneously refreshed as determined by the DCR. DRAM accesses are delayed during refresh. Only an active bus access to a DRAM block can delay refresh.

Figure 11-12 shows a bus cycle delayed by a refresh operation. Notice that $\overline{\text{DRAMW}}$ is forced high during refresh. The row address is held until the pending DRAM access.

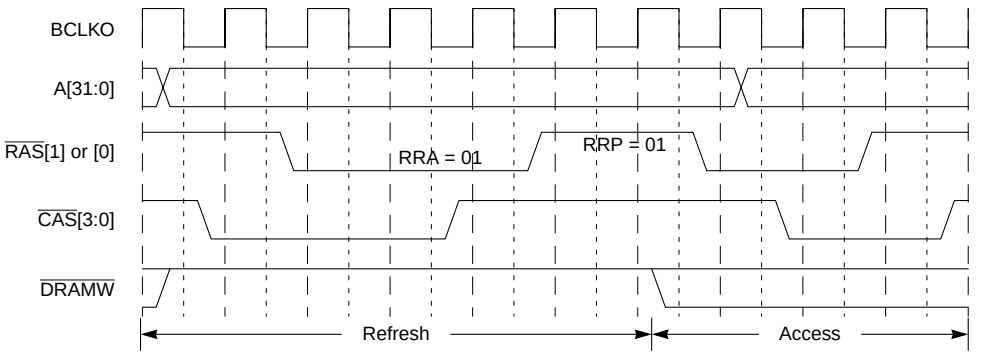


Figure 11-12. DRAM Access Delayed by Refresh

11.4 Synchronous Operation

By running synchronously with the system clock instead of responding to asynchronous control signals, SDRAM can (after an initial latency period) be accessed on every clock; 5-1-1-1 is a typical MCF5307 burst rate to SDRAM.

Note that because the MCF5307 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 11-10 lists common SDRAM commands.

Table 11-10. 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; $\overline{\text{RAS}}$ 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. Note that, after synchronous operation is selected by setting DCR[SO], DRAM controller registers reflect the synchronous operation and there is no way to return to asynchronous operation without resetting the processor.

11.4.1 DRAM Controller Signals in Synchronous Mode

Table 11-11 shows the behavior of DRAM signals in synchronous mode.

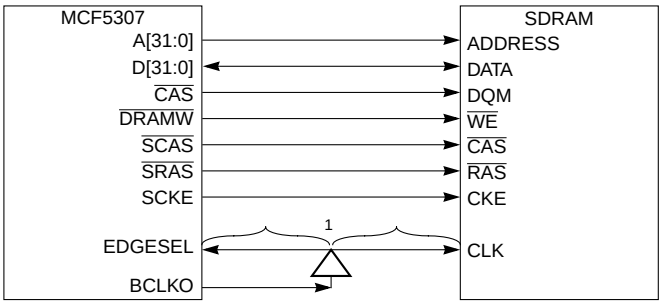
Table 11-11. Synchronous DRAM Signal Connections

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 $\overline{\text{SRAS}}$. Do not confuse $\overline{\text{SRAS}}$ with the DRAM controller's $\overline{\text{RAS}}$[1:0], which should not be interfaced to the SDRAM $\overline{\text{SRAS}}$ signals.
$\overline{\text{SCAS}}$	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 signal labeled $\overline{\text{SCAS}}$ on the SDRAM. Do not confuse $\overline{\text{SCAS}}$ with the DRAM controller's $\overline{\text{CAS}}$[3:0] signals.
$\overline{\text{DRAMW}}$	DRAM read/write. Asserted for write operations and negated for read operations.
$\overline{\text{RAS}}$[1:0]	Row address strobe. Select each memory block of SDRAMs connected to the MCF5307. One $\overline{\text{RAS}}$ 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 where operations are suspended or they can enter self-refresh mode. SCKE functionality is controlled by DCR[COC]. For designs using external multiplexing, setting COC allows SCKE to provide command-bit functionality.
$\overline{\text{CAS}}$[3:0]	Column address strobe. For synchronous operation, $\overline{\text{CAS}}$[3:0] function as byte enables to the SDRAMs. They connect to the DQM signals (or mask qualifiers) of the SDRAMs.

Table 11-11. Synchronous DRAM Signal Connections (Continued)

Signal	Description
BCLKO	Bus clock output. Connects to the CLK input of SDRAMs.
EDGESEL	Synchronous edge select. Provides additional output hold time for signals that interface to external SDRAMs. EDGESEL supports the three following modes for SDRAM interface signals: • Tied high. Signals change on the rising edge of BCLKO. • Tied low. Signals change on the falling edge of BCLKO. • Tied to buffered BCLKO. Signals change on the rising edge of the buffered clock. EDGESEL can provide additional output hold time for SDRAM interface signals, however the SDRAM clock and BCLKO frequencies must be the same. See Section 11.4.2, “Using Edge Select (EDGESEL).”

Figure 11-13 shows a typical signal configuration for synchronous mode.



[1] Trace length from buffer to CLK must equal length from buffer to EDGESEL.

Figure 11-13. MCF5307 SDRAM Interface

11.4.2 Using Edge Select (EDGESEL)

EDGESEL can ease system-level timings (note that the optional buffer in Figure 11-13 is for memories that need extra delay). The clock at the input to the SDRAM is monitored and data is held until the next edge of the bus clock, adding required output hold time to the address, data, and control signals.

To generate SDRAM interface timing, address, data, and control signals are clocked through a two-stage shift register. The first stage is clocked on the rising edge of BCLKO; the second is clocked on the falling edge. This makes the signal available for up to an additional half bus clock cycle, of which only a small amount is needed for proper timing.

Using the connection shown in Figure 11-13 ensures that data remains held for a longer time after the rising edge of the SDRAM clock input. This helps to match the MCF5307 output timing with the SDRAM clock.

Figure 11-14 shows the output wave forms for the interface signals changing on the rising edge (A) and falling edge (B) of BCLKO as determined by whether EDGESEL is tied high or low. It also shows timing (C) with EDGESEL tied to buffered BCLKO.

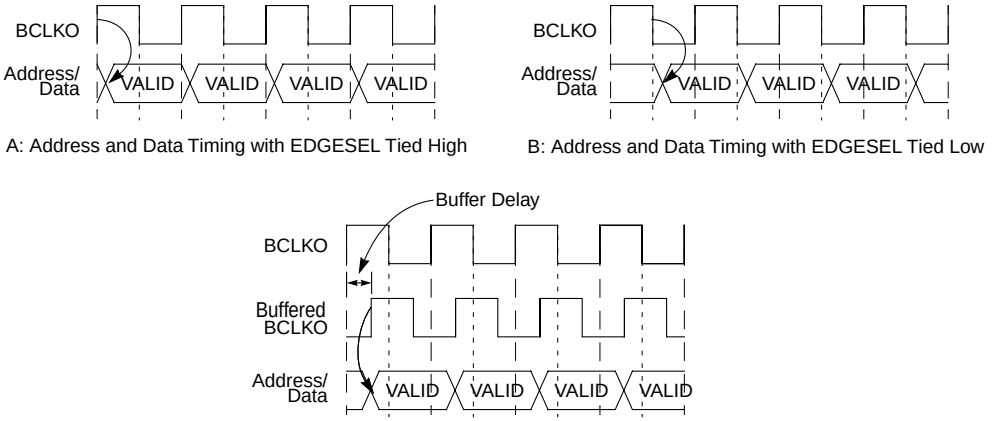


Figure 11-14. Using EDGESEL to Change Signal Timing

11.4.3 Synchronous Register Set

The memory map in Table 11-1 is the same for both synchronous and asynchronous operation. However, some bits are different, as noted in the following sections.

11.4.3.1 DRAM Control Register (DCR) in Synchronous Mode

The DRAM control register (DCR), Figure 11-15, controls refresh logic.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	SO	—	NAM	COC	IS	RTIM		RC								
Reset	0	Uninitialized														
R/W	R/W															
Addr	MBAR + 0x100															

Figure 11-15. DRAM Control Register (DCR) (Synchronous Mode)

Table 11-12 describes DCR fields.

Table 11-12. DCR Field Descriptions (Synchronous Mode)

Bits	Name	Description
15	SO	Synchronous operation. Selects synchronous or asynchronous mode. When in synchronous mode, the DRAM controller can be switched to ADRAM mode only by resetting the MCF5307. 0 Asynchronous DRAMs. Default at reset. 1 Synchronous DRAMs
14	—	Reserved, should be cleared.
13	NAM	No address multiplexing. Some implementations require external multiplexing. For example, when linear addressing is required, the DRAM should not multiplex addresses on DRAM accesses. 0 The DRAM controller multiplexes the external address bus to provide column addresses. 1 The DRAM controller does not multiplex the external address bus to provide column addresses.

Table 11-12. DCR Field Descriptions (Synchronous Mode) (Continued)

Bits	Name	Description
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 DRAM 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 DRAM 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, at which point 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 DRAM controller. Specifically, it determines the number of clocks inserted between a REF command and the next possible ACTV command. This same timing is used for both memory blocks controlled by the DRAM 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) * 16. Refresh can range from 16–8192 bus clocks to accommodate both standard and low-power DRAMs 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 (625 bus clocks at 40 MHz). This operation is the same as in asynchronous mode. # of bus clocks = 625 = (RC field + 1) * 16 RC = (625 bus clocks/16) -1 = 38.06, which rounds to 38; therefore, RC = 0x26.

11.4.3.2 DRAM Address and Control Registers (DACR0/DACR1) in Synchronous Mode

The DRAM address and control registers (DACR0 and DACR1), shown in Figure 11-16, contain the base address compare value and the control bits for both memory blocks 0 and 1 of the DRAM 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	1	0
Field	BA		—	RE	—	CASL		—	CBM			—	IMRS	PS		IP	PM	—	
Reset	Uninitialized			0	Uninitialized								0	Uninitialized					
R/W	R/W																		
Addr	MBAR+0x108 (DACR0); 0x110(DACR1)																		

Figure 11-16. DACR0 and DACR1 Registers (Synchronous Mode)

Table 11-13 describes DACR*n* fields.

Table 11-13. DACR0/DACR1 Field Descriptions (Synchronous Mode)

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>31–18</td><td>BA</td><td>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.</td></tr>
<tr><td>17–16</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>15</td><td>RE</td><td>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</td></tr>
<tr><td>14</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>13–12</td><td>CASL</td><td>$\overline{CAS}$ latency. Affects the following SDRAM timing specifications. Timing nomenclature varies with manufacturers. Refer to the SDRAM specification for the appropriate timing nomenclature:
<table>
<tr><th rowspan="2">Parameter</th><th colspan="4">Number of Bus Clocks</th></tr>
<tr><th>CASL= 00</th><th>CASL = 01</th><th>CASL= 10</th><th>CASL= 11</th></tr>
<tr><td>t_{RCD}—$\overline{SRAS}$ assertion to $\overline{SCAS}$ assertion</td><td>1</td><td>2</td><td>3</td><td>3</td></tr>
<tr><td>t_{CASL}—$\overline{SCAS}$ assertion to data out</td><td>1</td><td>2</td><td>3</td><td>3</td></tr>
<tr><td>t_{RAS}—ACTV command to precharge command</td><td>2</td><td>4</td><td>6</td><td>6</td></tr>
<tr><td>t_{RP}—Precharge command to ACTV command</td><td>1</td><td>2</td><td>3</td><td>3</td></tr>
<tr><td>t_{RWL},t_{RDL}—Last data input to precharge command</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>t_{EP}—Last data out to precharge command)</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
</table></td></tr>
<tr><td>11</td><td>—</td><td>Reserved, should be cleared.</td></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.
CBM Command Bit Bank Select Bits
000 17 18 and up
001 18 19 and up
010 19 20 and up
011 20 21 and up
100 21 22 and up
101 22 23 and up
110 23 24 and up
111 24 25 and up
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>
</table>

Table 11-13. DACR0/DACR1 Field Descriptions (Synchronous Mode) (Continued)

Bit	Name	Description
6	IMRS	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
5–4	PS	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
3	IP	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.
2	PM	Page mode. Indicates how the associated SDRAM block supports page-mode operation. 0 Page mode on bursts only. The DRAM controller dynamically bursts the transfer if it falls within a single page and the transfer size exceeds the port size of the SDRAM block. After the burst, the page closes and a precharge is issued. 1 Continuous page mode. The page stays open and only $\overline{\text{SCAS}}$ needs to be asserted for sequential SDRAM accesses that hit in the same page, regardless of whether the access is a burst.
1–0	—	Reserved, should be cleared.

11.4.3.3 DRAM Controller Mask Registers (DMR0/DMR1)

The DMR*n*, Figure 11-17, include mask bits for the base address and for address attributes. They are the same as in asynchronous operation.

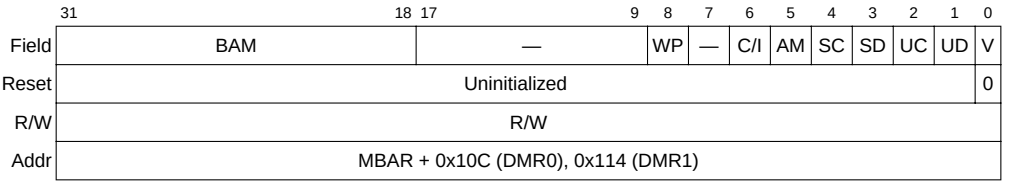


Figure 11-17. DRAM Controller Mask Registers (DMR0 and DMR1)

Table 11-14 describes DMR*n* fields.

Table 11-14. DMR0/DMR1 Field Descriptions

Bits	Name	Description
31–18	BAM	Base address mask. Masks the associated DACRn[BA]. Lets the DRAM controller connect to various DRAM sizes. Mask bits need not be contiguous (see Section 11.5, “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	AMx	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	External or 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

11.4.4 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.

When SDRAM blocks are accessed, the DRAM controller can operate in either burst or continuous page mode. The following sections describe the DRAM controller interface to SDRAM, the supported bus transfers, and initialization.

11.4.4.1 Address Multiplexing

Table 11-6 shows the generic address multiplexing scheme for SDRAM configurations. All possible address connection configurations can be derived from this table.

The following tables provide a more comprehensive, step-by-step way to determine the correct address line connections for interfacing the MCF5307 to 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 MCF5307, which is not necessarily the SDRAM port size. For example, if two 1M x 16-bit SDRAMs together form a 2M 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 11-15. MCF5307 to SDRAM Interface (8-Bit Port, 9-Column Address Lines)

MCF5307 Pins	A17	A16	A15	A14	A13	A12	A11	A10	A9	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
Row	17	16	15	14	13	12	11	10	9	18	19	20	21	22	23	24	25	26	27	28	29	30	31
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	A15	A16	A17	A18	A19	A20	A21	A22

Table 11-16. MCF5307 to SDRAM Interface (8-Bit Port,10-Column Address Lines)

MCF5307 Pins	A17	A16	A15	A14	A13	A12	A11	A10	A9	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
Row	17	16	15	14	13	12	11	10	9	19	20	21	22	23	24	25	26	27	28	29	30	31
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	A14	A15	A16	A17	A18	A19	A20	A21

Table 11-17. MCF5307 to SDRAM Interface (8-Bit Port,11-Column Address Lines)

MCF5307 Pins	A17	A16	A15	A14	A13	A12	A11	A10	A9	A19	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
Row	17	16	15	14	13	12	11	10	9	19	21	22	23	24	25	26	27	28	29	30	31
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	A13	A14	A15	A16	A17	A18	A19	A20

Table 11-18. MCF5307 to SDRAM Interface (8-Bit Port,12-Column Address Lines)

MCF5307 Pins	A17	A16	A15	A14	A13	A12	A11	A10	A9	A19	A21	A23	A24	A25	A26	A27	A28	A29	A30	A31
Row	17	16	15	14	13	12	11	10	9	19	21	23	24	25	26	27	28	29	30	31
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	A12	A13	A14	A15	A16	A17	A18	A19

Table 11-19. MCF5307 to SDRAM Interface (8-Bit Port,13-Column Address Lines)

MCF5307 Pins	A17	A16	A15	A14	A13	A12	A11	A10	A9	A19	A21	A23	A25	A26	A27	A28	A29	A30	A31
Row	17	16	15	14	13	12	11	10	9	19	21	23	25	26	27	28	29	30	31
Column	0	1	2	3	4	5	6	7	8	18	20	22	24						
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18

Table 11-20. MCF5307 to SDRAM Interface (16-Bit Port, 8-Column Address Lines)

MCF5307 Pins	A16	A15	A14	A13	A12	A11	A10	A9	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
Row	16	15	14	13	12	11	10	9	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
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	A15	A16	A17	A18	A19	A20	A21	A22

Table 11-21. MCF5307 to SDRAM Interface (16-Bit Port, 9-Column Address Lines)

MCF5307 Pins	A16	A15	A14	A13	A12	A11	A10	A9	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
Row	16	15	14	13	12	11	10	9	18	19	20	21	22	23	24	25	26	27	28	29	30	31
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	A14	A15	A16	A17	A18	A19	A20	A21

Table 11-22. MCF5307 to SDRAM Interface (16-Bit Port, 10-Column Address Lines)

MCF5307 Pins	A16	A15	A14	A13	A12	A11	A10	A9	A18	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
Row	16	15	14	13	12	11	10	9	18	20	21	22	23	24	25	26	27	28	29	30	31
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	A13	A14	A15	A16	A17	A18	A19	A20

Table 11-23. MCF5307 to SDRAM Interface (16-Bit Port, 11-Column Address Lines)

MCF5307 Pins	A16	A15	A14	A13	A12	A11	A10	A9	A18	A20	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
Row	16	15	14	13	12	11	10	9	18	20	22	23	24	25	26	27	28	29	30	31
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	A12	A13	A14	A15	A16	A17	A18	A19

Table 11-24. MCF5307 to SDRAM Interface (16-Bit Port, 12-Column Address Lines)

MCF5307 Pins	A16	A15	A14	A13	A12	A11	A10	A9	A18	A20	A22	A24	A25	A26	A27	A28	A29	A30	A31
Row	16	15	14	13	12	11	10	9	18	20	22	24	25	26	27	28	29	30	31
Column	1	2	3	4	5	6	7	8	17	19	21	23							
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18

Table 11-25. MCF5307to SDRAM Interface (16-Bit Port, 13-Column-Address Lines)

MCF5307 Pins	A16	A15	A14	A13	A12	A11	A10	A9	A18	A20	A22	A24	A26	A27	A28	A29	A30	A31
Row	16	15	14	13	12	11	10	9	18	20	22	24	26	27	28	29	30	31
Column	1	2	3	4	5	6	7	8	17	19	21	23	25					
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17

Table 11-26. MCF5307 to SDRAM Interface (32-Bit Port, 8-Column Address Lines)

MCF5307 Pins	A15	A14	A13	A12	A11	A10	A9	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
Row	15	14	13	12	11	10	9	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
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	A14	A15	A16	A17	A18	A19	A20	A21

Table 11-27. MCF5307 to SDRAM Interface (32-Bit Port, 9-Column Address Lines)

MCF5307 Pins	A15	A14	A13	A12	A11	A10	A9	A17	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
Row	15	14	13	12	11	10	9	17	19	20	21	22	23	24	25	26	27	28	29	30	31
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	A13	A14	A15	A16	A17	A18	A19	A20

Table 11-28. MCF5307 to SDRAM Interface (32-Bit Port, 10-Column Address Lines)

MCF5307 Pins	A15	A14	A13	A12	A11	A10	A9	A17	A19	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
Row	15	14	13	12	11	10	9	17	19	21	22	23	24	25	26	27	28	29	30	31
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	A12	A13	A14	A15	A16	A17	A18	A19

Table 11-29. MCF5307 to SDRAM Interface (32-Bit Port, 11-Column Address Lines)

MCF5307 Pins	A15	A14	A13	A12	A11	A10	A9	A17	A19	A21	A23	A24	A25	A26	A27	A28	A29	A30	A31
Row	15	14	13	12	11	10	9	17	19	21	23	24	25	26	27	28	29	30	31
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	A11	A12	A13	A14	A15	A16	A17	A18

Table 11-30. MCF5307 to SDRAM Interface (32-Bit Port, 12-Column Address Lines)

MCF5307 Pins	A15	A14	A13	A12	A11	A10	A9	A17	A19	A21	A23	A25	A26	A27	A28	A29	A30	A31
Row	15	14	13	12	11	10	9	17	19	21	23	25	26	27	28	29	30	31
Column	2	3	4	5	6	7	8	16	18	20	22	24						
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17

11.4.4.2 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) to the MCF5307, the connections would be as shown in Table 11-31.

Table 11-31. SDRAM Hardware Connections

SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10 = CMD	BA0	BA1
MCF5307 Pins	A15	A14	A13	A12	A11	A10	A9	A17	A18	A19	A20	A21	A22

11.4.4.3 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 clock for as long as accesses are 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 MCF5307 DRAM controller. Thus, instead of the SDRAM enabling its internal burst incrementing capability, the MCF5307 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 MCF5307.

Figure 11-18 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 BCLKO cycles. Because t_{RCD} is equal to the read CAS

latency ($\overline{SCAS}$ assertion to data out), this value is also 2 BCLKO cycles. Notice that NOPs are executed until the last data is read. A PALL command is executed one cycle after the last data transfer.

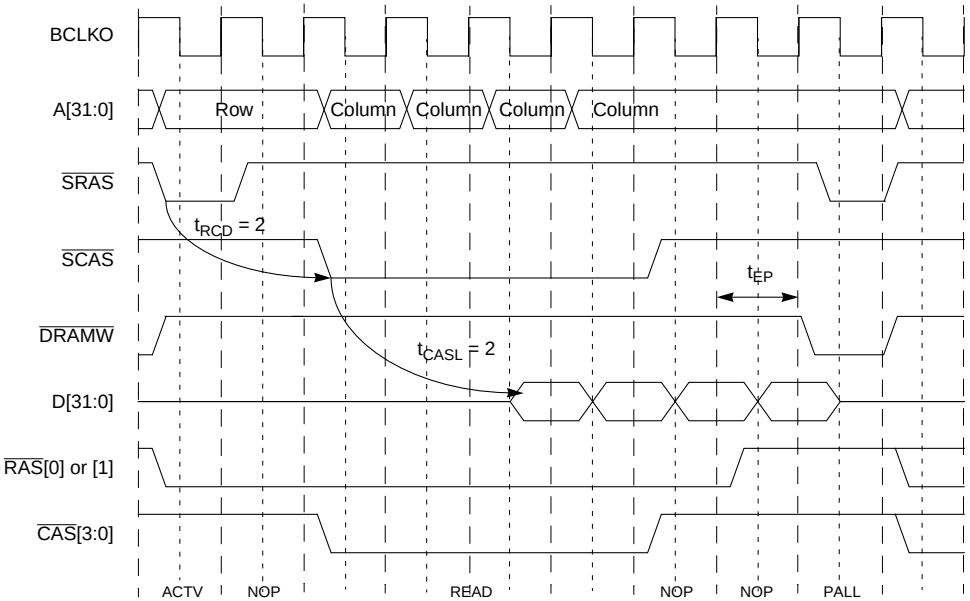


Figure 11-18. Burst Read SDRAM Access

Figure 11-19 shows the burst write operation. In this example, DACR[CASL] = 01, which creates an $\overline{SRAS}$-to-$\overline{SCAS}$ delay (t_{RCD}) of 2 BCLKO cycles. Note that data is available upon $\overline{SCAS}$ 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.

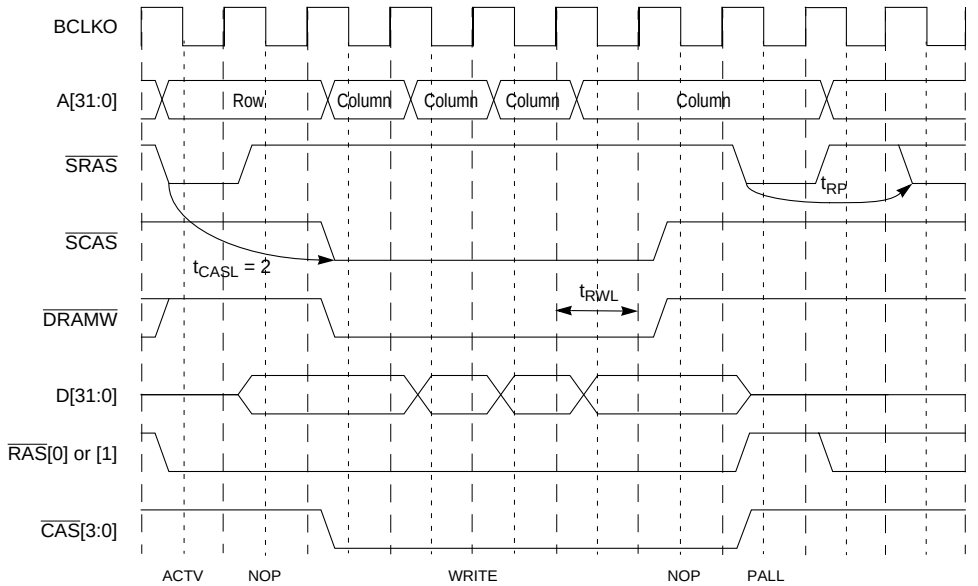


Figure 11-19. Burst Write SDRAM Access

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 $\overline{\text{CAS}}$ 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.

11.4.4.4 Continuous Page Mode

Continuous page mode is identical to burst page mode, except that it allows the processor core to handle successive bus cycles that hit the same page without having to close the page. When the current bus cycle finishes, the MCF5307 core internal pipelined bus can predict whether the upcoming cycle will hit in the same page.

- If the next bus cycle is not pending or misses in the page, the PALL command is generated to the SDRAM.
- If the next bus cycle is pending and hits in the page, the page is left open, and the next SDRAM access begins with a READ or WRITE command.

- Because of the nature of the internal CPU pipeline this condition does not occur often; however, the use of continuous page mode is recommended because it can provide a slight performance increase.

Figure 11-20 shows two read accesses in continuous page mode. Note that there is no precharge between the two accesses. Also notice that the second cycle begins with a read operation with no ACTV command.

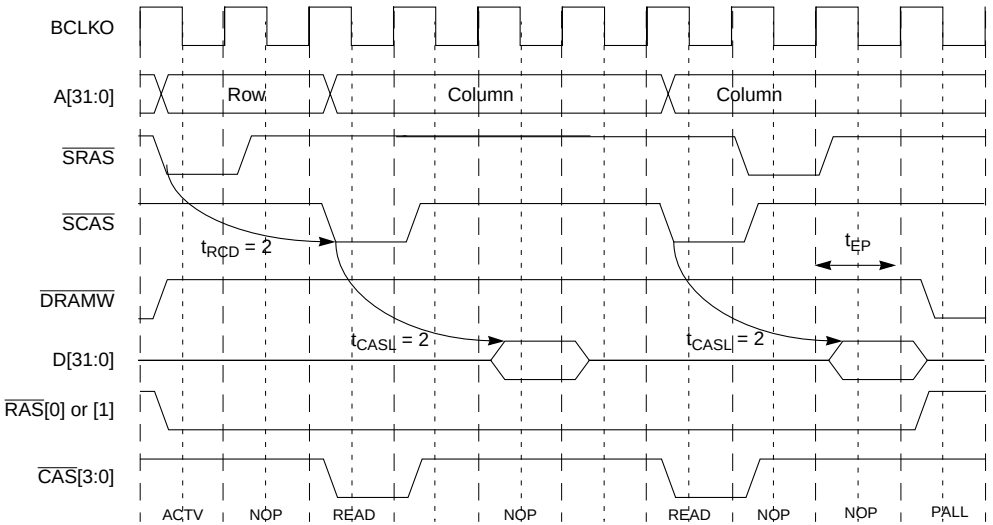


Figure 11-20. Synchronous, Continuous Page-Mode Access—Consecutive Reads

Figure 11-21 shows a write followed by a read in continuous page mode. Because the bus cycle is terminated with a WRITE command, the second cycle begins sooner after the write than after the read. A read requires data to be returned before the bus cycle can terminate. Note that in continuous page mode, secondary accesses output the column address only.

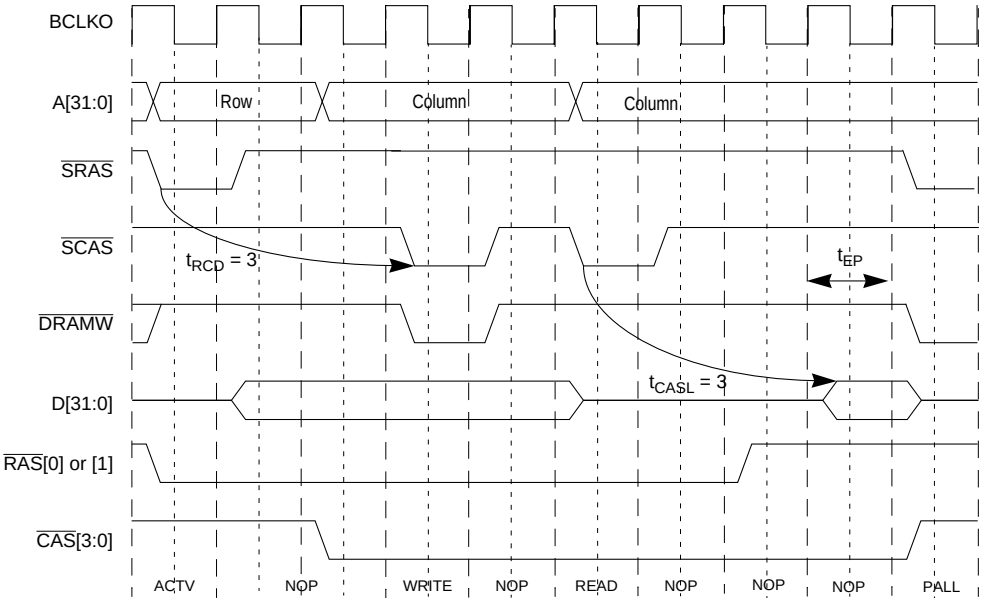


Figure 11-21. Synchronous, Continuous Page-Mode Access—Read after Write

11.4.4.5 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. 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 11-22 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.

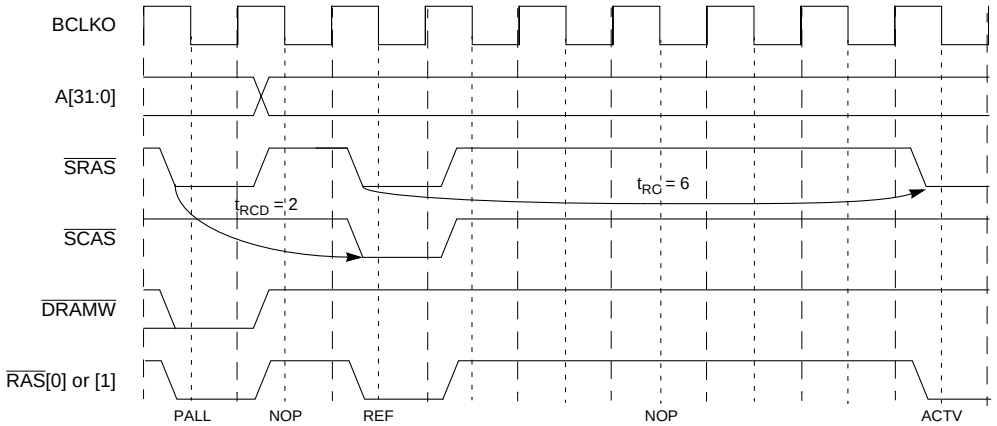


Figure 11-22. Auto-Refresh Operation

11.4.4.6 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 11-23 shows the self-refresh operation.

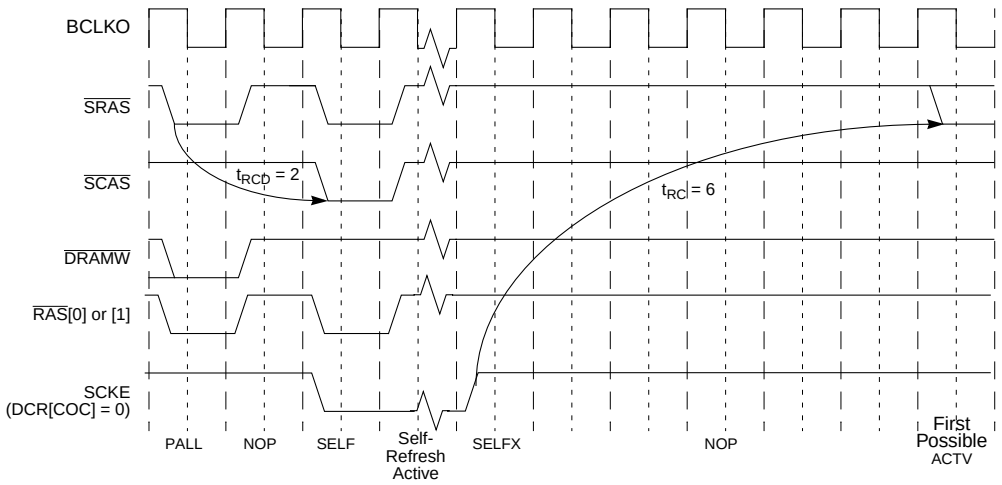


Figure 11-23. Self-Refresh Operation

11.4.5 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 DCR[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.

11.4.5.1 Mode Register Settings

It is possible to configure the operation of SDRAMs, namely their burst operation and $\overline{\text{CAS}}$ latency, through the SDRAM component's mode register. $\overline{\text{CAS}}$ latency is a function of the speed of the SDRAM and the bus clock of the DRAM controller. The DRAM controller operates at a $\overline{\text{CAS}}$ latency of 1, 2, or 3.

Although the MCF5307 DRAM controller supports bursting operations, it does not use the bursting features of the SDRAMs. Because the MCF5307 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 MCF5307 DRAM controller generates the burst cycles rather than the SDRAM device. Because the MCF5307 generates a new address and a READ or WRITE command for each transfer within the burst, the SDRAM mode register should be set either to a burst length of one or to not burst. This allows bursting to be controlled by the MCF5307 instead.

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 MCF5307 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 11-24 shows the MRS command, which occurs in the first clock of the bus cycle.

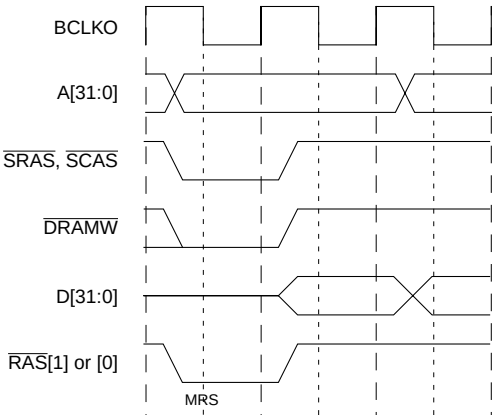


Figure 11-24. Mode Register Set (MRS) Command

11.5 SDRAM Example

This example interfaces a 2M x 32-bit x 4 bank SDRAM component to a MCF5307 operating at 40 MHz. Table 11-32 lists design specifications for this example.

Table 11-32. 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

11.5.1 SDRAM Interface Configuration

To interface this component to the MCF5307 DRAM controller, use the connection table that corresponds to a 32-bit port size with 8 columns (Table 11-26). Two pins select one of four banks when the part is functional. Table 11-33 shows the proper hardware hook-up.

Table 11-33. SDRAM Hardware Connections

MCF5307 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

11.5.2 DCR Initialization

At power-up, the DCR has the following configuration if synchronous operation and SDRAM address multiplexing is desired.

	15	14	13	12	11	10	9	8								0
Field	SO	res	NAM	COC	IS	RTIM		RC								
Setting	1	X	0	0	0	0	0	0	0	0	1	0	0	1	1	0
(hex)	8				0			2				6				

Figure 11-25. Initialization Values for DCR

This configuration results in a value of 0x8026 for DCR, as shown in Table 11-34.

Table 11-34. DCR Initialization Values

Bits	Name	Setting	Description
15	SO	1	Indicating synchronous operation
14	—	x	Don't care (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

11.5.3 DACR Initialization

As shown in Figure 11-26, in this example 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 0xFF80_0000. Continuous page mode feature is used.

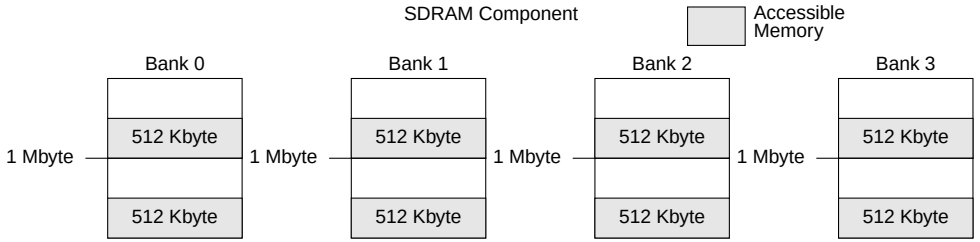


Figure 11-26. SDRAM Configuration

The DACRs should be programmed as shown in Figure 11-27.

	31 … 18	17–16
Field	BA	—
Setting	1111_1111_1000_10	xx
(hex)	15 15 8	8

	15	14	13–12	11	10–8	7	6	5–4	3	2	1–0
Field	RE	—	CASL	—	CBM	—	IMRS	PS	IP	PM	—
Setting	0	X	00	X	011	X	0	00	0	1	xx
(hex)	0			3		0			4		

Figure 11-27. DACR Register Configuration

This configuration results in a value of DACR0 = 0xFF88_0304, as described in Table 11-35. 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 11-35. DACR Initialization Values

Bits	Name	Setting	Description
31–18	BA		Base address. So DACR0[31–16] = 0xFF88, which places the starting address of the SDRAM accessible memory at 0xFF88_0000.
17–16	—		Reserved. Don't care.
15	RE	0	0, which 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 $\overline{\text{CAS}}$ 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.
3	IP	0	Indicates precharge has not been initiated.

Table 11-35. DACR Initialization Values

Bits	Name	Setting	Description
2	PM	1	Indicates continuous page mode
1–0	—		Reserved. Don't care.

11.5.4 DMR Initialization

In this example, again, 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	0	0	0	0	0	0	0	0	0	1	1	1	0	1		X	X
(hex)	0				0				7				4				

	15						9	8	7	6	5	4	3	2	1	0
Field	—							WP	—	C/I	AM	SC	SD	UC	UD	V
Setting	X	X	X	X	X	X	X	0	X	1	1	1	0	1	0	1
(hex)	0				0				7				5			

Figure 11-28. DMR0 Register

With this configuration, the DMR0 = 0x0074_0075, as described in Table 11-36.

Table 11-36. DMR0 Initialization Values

Bits	Name	Setting	Description
31–16	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.
15–9	—		Reserved. Don't care.
8	WP	0	Allow reads and writes
7	—		Reserved
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.

11.5.5 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 MCF5307 address pins must be determined while being aware of masking requirements.

Table 11-37 lists the desired initialization setting:

Table 11-37. Mode Register Initialization

MCF5307 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	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	X
(hex)	0				0				0				0			

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field																V
Setting	0	0	0	0	1	0	0	X	X	X	X	X	X	X	X	X
(hex)	0				8				0				0			

Figure 11-29. Mode Register Mapping to MCF5307 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.

11.5.6 Initialization Code

The following assembly code initializes the SDRAM example.

Power-Up Sequence:

```
move.w  #0x8026, 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 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
```

Part III
Peripheral Module

Intended Audience

Part III describes the operation and configuration of the MCF5307 DMA, timer, UART, and parallel port modules, and describes how they interface with the system integration unit, described in Part II.

Contents

Part III contains the following chapters:

- Chapter 12, “DMA Controller Module,” provides an overview of the DMA controller 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, showing timing diagrams for various operations.
- Chapter 13, “Timer Module,” describes configuration and operation of the two general-purpose timer modules, timer 0 and timer 1. It includes programming examples.
- Chapter 14, “UART Modules,” describes the use of the universal asynchronous/synchronous receiver/transmitters (UARTs) implemented on the MCF5307 and includes programming examples.
- Chapter 15, “Parallel Port (General-Purpose I/O),” describes the operation and programming model of the parallel port pin assignment, direction-control, and data registers. It includes a code example for setting up the parallel port.

Acronyms and Abbreviations

Table III-i describes acronyms and abbreviations used in Part III.

Table III-i. Acronyms and Abbreviated Terms

Term	Meaning
ADC	Analog-to-digital conversion
BIST	Built-in self test
CODEC	Code/decode
DAC	Digital-to-analog conversion
DMA	Direct memory access
DSP	Digital signal processing
EDO	Extended data output (DRAM)
FIFO	First-in, first-out
GPIO	
I^2C	Inter-integrated circuit
IEEE	Institute for Electrical and Electronics Engineers
IFP	Instruction fetch pipeline
IPL	Interrupt priority level
JEDEC	Joint Electron Device Engineering Council
JTAG	Joint Test Action Group
LIFO	Last-in, first-out
LRU	Least recently used
LSB	Least-significant byte
lsb	Least-significant bit
MAC	Multiple accumulate unit
MBAR	Memory base address register
MSB	Most-significant byte
msb	Most-significant bit
Mux	Multiplex
NOP	No operation
OEP	Operand execution pipeline
PC	Program counter
PCLK	Processor clock
PLL	Phase-locked loop

Table III-i. Acronyms and Abbreviated Terms (Continued)

Term	Meaning
PLRU	Pseudo least recently used
POR	Power-on reset
PQFP	Plastic quad flat pack
RISC	Reduced instruction set computing
Rx	Receive
SIM	System integration module
SOF	Start of frame
TAP	Test access port
TTL	Transistor-to-transistor logic
Tx	Transmit
UART	Universal asynchronous/synchronous receiver transmitter

Chapter 12
DMA Controller Module

This chapter describes the MCF5307 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.

12.1 Overview

The direct memory access (DMA) controller module provides an efficient way to move blocks of data with minimal processor interaction. The DMA module, shown in Figure 12-1, provides four channels that allow byte, word, or longword operand transfers. Each channel has a dedicated set of registers that define the source and destination addresses (SAR*n* and DAR*n*), byte count (BCR*n*), and control and status (DCR*n* and DSR*n*). Transfers can be dual or single address to off-chip devices or dual address to on-chip devices, such as UART, SDRAM controller, and parallel port.

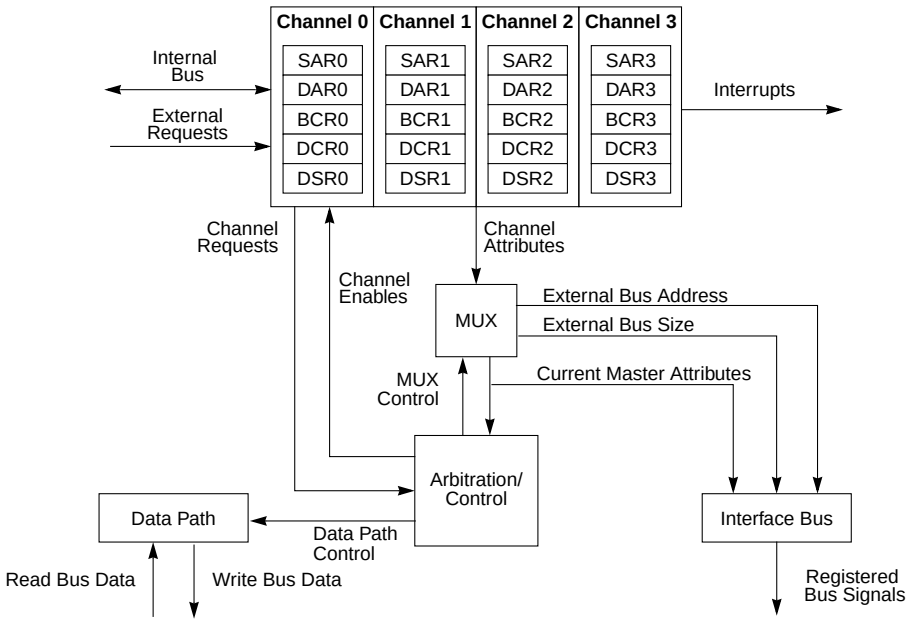


Figure 12-1. DMA Signal Diagram

12.1.1 DMA Module Features

The DMA controller module features are as follows:

- Four fully independent, programmable DMA controller channels/bus modules
- Auto-alignment feature for source or destination accesses
- Dual- and single-address transfers
- Two external request pins ($\overline{\text{DREQ}}$[1:0]) provided for channels 1 and 0
- Channel arbitration on transfer boundaries
- Data transfers in 8-, 16-, 32-, or 128-bit blocks using a 16-byte buffer
- Continuous-mode and cycle-steal transfers
- Independent transfer widths for source and destination
- Independent source and destination address registers
- Data transfer can occur in as few as two clocks

12.2 DMA Signal Description

Table 12-1 briefly describes the DMA module signals that provide handshake control for either a source or destination external device.

Table 12-1. DMA Signals

Signal	I/O	Description
$\overline{\text{DREQ}}$[1:0]/ PP[6:5]	I	External DMA request. $\overline{\text{DREQ}}$[1:0] can serve as the DMA request inputs or as two parallel port bits. They are programmable individually through the PAR. A peripheral device asserts these inputs to request an operand transfer between it and memory. $\overline{\text{DREQ}}$ signals are asserted to initiate DMA accesses in the respective channels. The system should drive unused $\overline{\text{DREQ}}$ signals to logic high. Although each channel has an individual $\overline{\text{DREQ}}$ signal, in the MCF5307 only channels 0 and 1 connect to external $\overline{\text{DREQ}}$ pins.$\overline{\text{DREQ}}$ signals for channels 2 and 3 are connected to the UART0 and UART1 bus interrupt signals.
TT[1:0]/ PP[1:0]	O	Transfer type. A DMA access is indicated by the transfer type pins, TT[1:0] = 01. The transfer modifier, TM[2:0] configurations shown below are meaningful only if TT[1:0] = 01, indicating an external master or DMA access.
TM[2:0] /PP[4:2]	O	Multiplexed transfer attribute pins. The encodings below are valid when TT[1:0] = 01 and internal DMA channels are driving the bus. DMA transfer information on TM[2:1] can be provided on every DMA transfer or only on the last transfer by programming DCR[AT]. TM[2:1] Encoding 00 DMA acknowledge information not provided 01 DMA transfer, channel 0 10 DMA transfer, channel 1 11 Reserved TM0 Encoding for DMA as master (TT = 01) 0 Single-address access negated 1 Single-address access For TT[1:0] = 01, the TM0 encoding is independent of TM[2:1]. If DCR[SAA] is set, TM0 designates a single-address DMA access.

12.3 DMA Transfer Overview

The DMA module usually transfers data faster than the ColdFire core can under software control. The term ‘direct memory access’ refers to peripheral device’s ability to access system memory directly, 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 MCF5307 on-chip peripherals do not support single-address transfers.

The processor generates DMA requests internally by setting DCR[START]; a device can generate a DMA request externally by using $\overline{\text{DREQ}}$ pins. The processor can program bus bandwidth for each channel. The channels support cycle-steal and continuous transfer modes; see Section 12.5.1, “Transfer Requests (Cycle-Steal and Continuous Modes).”

The DMA controller supports dual- and single-address transfers as follows. In both, the DMA channel supports 32 address bits and 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 an external device using $\overline{\text{DREQ}}$. Two types of transfer can occur, a read from a source device or a write to a destination device; see Figure 12-2.

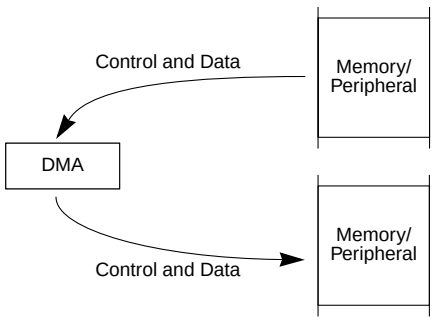


Figure 12-2. Dual-Address Transfer

- Single-address transfers—An external device can initiate a single-address transfer by asserting $\overline{\text{DREQ}}$. The MCF5307 provides address and control signals for single-address transfers. The external device reads to or writes from the specified address, as Figure 12-3 shows. External logic is required.

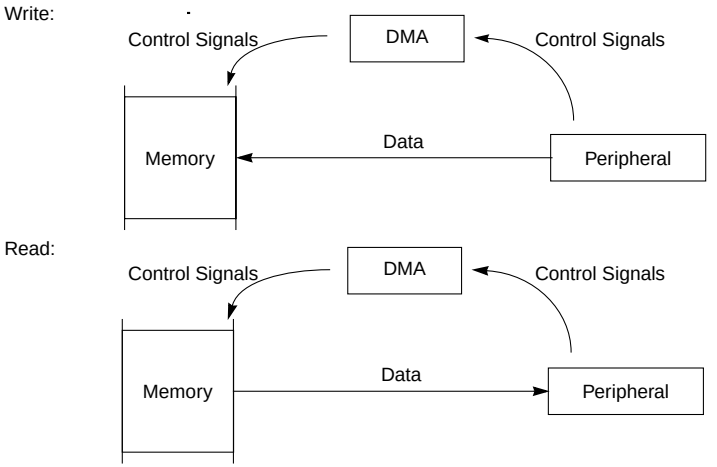


Figure 12-3. Single-Address Transfers

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 12.4.5, "DMA Status Registers (DSR0–DSR3)."

12.4 DMA Controller Module Programming Model

This section describes each internal register and its bit assignment. Note that there is no way to prevent a write to a control register during a DMA transfer. Table 12-2 shows the mapping of DMA controller registers. Note the differences for the byte count registers depending on the value of MPARK[BCR24BIT].

Table 12-2. Memory Map for DMA Controller Module Registers

DMA Channel	MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0	0x300	Source address register 0 (SAR0) [p. 12-6]			
	0x304	Destination address register 0 (DAR0) [p. 12-7]			
	0x308	DMA control register 0 (DCR0) [p. 12-8]			
	0x30C	Byte count register 0 (BCR24BIT = 0) [1]		Reserved	
	0x30C	Reserved	Byte count register 0 (BCR24BIT = 1) [1] (BCR0) [p. 12-7]		
	0x310	DMA status register 0 (DSR0) [p. 12-10]	Reserved		
	0x314	DMA interrupt vector register 0 (DIVR0) [p. 12-11]	Reserved		
1	0x340	Source address register 1 (SAR1) [p. 12-6]			
	0x344	Destination address register 1 (DAR1) [p. 12-7]			
	0x348	DMA control register 1 (DCR1) [p. 12-8]			
	0x34C	Byte count register 1 (BCR24BIT = 0) [1]		Reserved	
	0x34C	Reserved	Byte count register 1 (BCR24BIT = 1) [1] (BCR1) [p. 12-7]		
	0x350	DMA status register 1 (DSR1) [p. 12-10]	Reserved		
	0x354	DMA interrupt vector register 1 (DIVR1) [p. 12-11]	Reserved		
2	0x380	Source address register 2 (SAR2) [p. 12-6]			
	0x384	Destination address register 2 (DAR2) [p. 12-7]			
	0x388	DMA control register 2 (DCR2) [p. 12-8]			
	0x38C	Byte count register 2 (BCR24BIT = 0) [1]		Reserved	
	0x38C	Reserved	Byte count register 2 (BCR24BIT = 1) [1] (BCR2) [p. 12-7]		
	0x390	DMA status register 2 (DSR2) [p. 12-10]	Reserved		
	0x394	DMA interrupt vector register 2 (DIVR2) [p. 12-11]	Reserved		

Table 12-2. Memory Map for DMA Controller Module Registers (Continued)

DMA Channel	MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
3	0x3C0	Source address register 3 (SAR3) [p. 12-6]			
	0x3C4	Destination address register 3 (DAR3) [p. 12-7]			
	0x3C8	DMA control register 3 (DCR3) [p. 12-8]			
	0x3CC	Byte count register 3 (BCR24BIT = 0) [1]		Reserved	
	0x3CC	Reserved	Byte count register 3 (BCR24BIT = 1) [1] (BCR3) [p. 12-7]		
	0x3D0	DMA status register 3 (DSR3) [p. 12-10]	Reserved		
	0x3D4	DMA interrupt vector register 3 (DIVR3) [p. 12-11]	Reserved		

[1] On the original MCF5307 mask set (H55J), the BCR of the DMA channels can accommodate only 16 bits. However, because the revised MCF5307 supports a 24-bit byte count range, the position of the BCR in the memory map depends on whether a 16- or 24-bit byte counter is selected. The 24-bit byte count can be selected by setting BCR24BIT = 1, making DCR[AT] available. The AT bit selects whether DMA channels assert acknowledge during the entire transfer or only at the final transfer of a DMA transaction.

New applications should take advantage of the full range of the 24-bit byte counter, including the AT bit. The 16-bit byte count option (BCR24BIT = 0) retains compatibility with older MCF5307 revisions.

NOTE:

External masters cannot access MCF5307 on-chip memories or MBAR, but they can access DMA module registers.

12.4.1 Source Address Registers (SAR0–SAR3)

SAR*n*, Figure 12-4, contains the address from which the DMA controller requests data. In single-address mode, SAR*n* provides the address regardless of the direction.

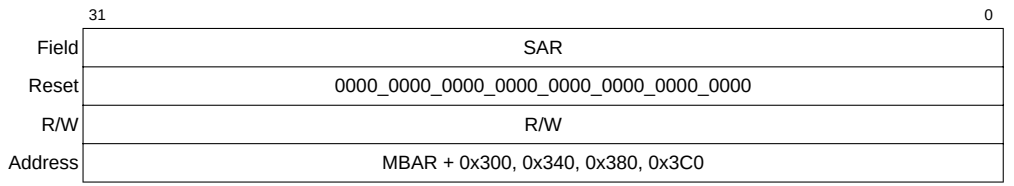


Figure 12-4. Source Address Registers (SAR*n*)

NOTE:

SAR/DAR address ranges cannot be programmed to on-chip SRAM because it cannot be accessed by on-chip DMA.

12.4.2 Destination Address Registers (DAR0–DAR3)

For dual-address transfers only, DAR*n*, Figure 12-5, 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	MBAR + 304, 0x344, 0x384, 0x3C4	

Figure 12-5. Destination Address Registers (DAR*n*)

NOTE:

On-chip DMAs do not maintain coherency with MCF5307 caches and so must not transfer data to cacheable memory.

12.4.3 Byte Count Registers (BCR0–BCR3)

BCR*n*, Figure 12-6 and Figure 12-7, holds the number of bytes yet to be transferred for a given block.The offset within the memory map is based on the value of MPARK[BCR24BIT]. BCR*n* decrements on the successful completion of the address transfer of either a write transfer in dual-address mode or any transfer in single-address mode. BCR*n* decrements by 1, 2, 4, or 16 for byte, word, longword, or line accesses, respectively.

Figure 12-6 shows BCR for BCR24BIT = 1.

	31 24	23 0
Field	—	BCR
Reset	—	0000_0000_0000_0000_0000_0000
R/W	R/W	
Address	MBAR + 0x30C, 0x34C, 0x38C, 0x3AC	

Figure 12-6. Byte Count Registers (BCR*n*)—BCR24BIT = 1

Figure 12-7 shows BCR for BCR24BIT = 0.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	BCR															
Reset	0000_0000_0000_0000															
R/W																
Addr	MBAR + 0x30C, 0x34C, 0x38C, 0x3AC															

Figure 12-7. BCR*n*—BCR24BIT = 0

DSR[DONE], shown in Figure 12-9, is set when the block transfer is complete.

When a transfer sequence is initiated and BCR*n*[BCR] is not divisible by 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 12.4.5, “DMA Status Registers (DSR0–DSR3).”

12.4.4 DMA Control Registers (DCR0–DCR3)

DCR*n*, Figure 12-8, is used for configuring the DMA controller module. Note that DCR[AT] is available only if BCR24BIT = 1.

	31	30	29	28	27	25	24	23	22	21	20	19	18	17	16
Field	INT	EEXT	CS	AA	BWC		SAA	S_RW	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	MBAR + 0x308, 0x348, 0x388, 0x3A8		

Figure 12-8. DMA Control Registers (DCR*n*)

[1] Available only if BCR24BIT = 1, otherwise reserved.

Table 12-3 describes DCR fields.

Table 12-3. 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 $\overline{\text{DREQ}}$ when EEXT = 1. 0 External request is ignored. 1 Enables external request to initiate transfer. Internal request is always enabled. It is initiated by writing a 1 to the START bit.

Table 12-3. DCR*n* Field Descriptions (Continued)

Bits	Name	Description
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 $\overline{\text{DREQ}}$.
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 12.5.4.2, “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.
27–25	BWC	Bandwidth control. Indicates the number of bytes in a block transfer. When the byte count reaches a multiple of the BWC value, the DMA releases the bus. For example, if BCR24BIT is 0, BWC is 001 (512 bytes or value of 0x0200), and BCR is 0x1000, the bus is relinquished after BCR values of 0x2000, 0x1E00, 0x1C00, 0x1A00, 0x1800, 0x1600, 0x1400, 0x1200, 0x1000, 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. BWC BCR24BIT = 0 BCR24BIT = 1 000 DMA has priority. It does not negate its request until its transfer completes. 001 512 16384 010 1024 32768 011 2048 65536 100 4096 131072 101 8192 262144 110 16384 524288 111 32768 1048576
24	SAA	Single-address access. Determines whether the DMA channel is in dual- or single-address mode 0 Dual-address mode. 1 Single-address mode. The DMA provides an address from the SAR and directional control, bit S_RW, to allow two peripherals (one might be memory) to exchange data within a single access. Data is not stored by the DMA.
23	S_RW	Single-address access read/write value. Valid only if SAA = 1. Specifies the value of the read signal during single-address accesses. This provides directional control to the bus controller. 0 Forces the read signal to 0. 1 Forces the read signal to 1.
22	SINC	Source increment. Controls whether a source address increments after each successful transfer. 0 No change to SAR after a successful transfer. 1 The SAR increments by 1, 2, 4, or 16, as determined by the transfer size.
21–20	SSIZE	Source size. Determines the data size of the source bus cycle for the DMA control module. 00 Longword 01 Byte 10 Word 11 Line
19	DINC	Destination increment. Controls whether a destination address increments after each successful transfer. 0 No change to the DAR after a successful transfer. 1 The DAR increments by 1, 2, 4, or 16, depending upon the size of the transfer.
18–17	DSIZE	Destination size. Determines the data size of the destination bus cycle for the DMA controller. 00 Longword 01 Byte 10 Word 11 Line

Table 12-3. DCR*n* Field Descriptions (Continued)

Bits	Name	Description
16	START	Start transfer. 0 DMA inactive 1 The DMA begins the transfer in accordance to the values in the control registers. START is cleared automatically after one clock and is always read as logic 0.
15	AT	AT is available only if 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.

12.4.5 DMA Status Registers (DSR0–DSR3)

In response to an event, the DMA controller writes to the appropriate DSR*n* bit, Figure 12-9. 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	—	0	0	0	—	0	0	0
R/W	R/W							
Address	MBAR + 0x310, 0x350, 0x390, 0x3D0							

Figure 12-9. DMA Status Registers (DSR*n*)

Table 12-4 describes DSR*n* fields.

Table 12-4. 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 either during the read portion of a transfer or during an access in single-address mode (SAA = 1).
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.

Table 12-4. DSR*n* Field Descriptions (Continued)

Bits	Name	Description
2	REQ	Request 0 No request is pending or the channel is currently active. Cleared when the channel is selected. 1 The DMA channel has a transfer remaining and the channel is not selected.
1	BSY	Busy 0 DMA channel is inactive. Cleared when the DMA has finished the last transaction. 1 BSY is set the first time the channel is enabled after a transfer is initiated.
0	DONE	Transactions done. Set when all DMA controller transactions complete normally, as determined by transfer count and 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 as an interrupt handler to clear the DMA interrupt and error bits.

12.4.6 DMA Interrupt Vector Registers (DIVR0–DIVR3)

The contents of a DMA interrupt vector register (DIVR*n*), Figure 12-10, are driven onto the internal bus in response to an interrupt acknowledge cycle.

	7	0
Field	Interrupt Vector Bits	
Reset	0000_1111	
R/W	R/W	
Address	MBAR + 0x314, 0x354, 0x394, 0x3D4	

Figure 12-10. DMA Interrupt Vector Registers (DIVR*n*)

12.5 DMA Controller Module Functional Description

In the following discussion, the term ‘DMA request’ implies that DCR[START] or DCR[EEXT] is set, followed by assertion of $\overline{\text{DREQ}}$. The START bit is cleared when the channel begins an internal access.

Before initiating a dual-address access, the DMA module verifies that DCR[SSIZE,DSIZE] are consistent with the source and destination addresses. If the source and destination are not the same size, the configuration error bit, DSR[CE], is also set. If misalignment is detected, no transfer occurs, CE is set, and, depending on the DCR configuration, an interrupt event is issued. Note that if the auto-align bit, DCR[AA], is set, error checking is performed on 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 SSIZE and DSIZE. See Section 12.4.4, “DMA Control Registers (DCR0–DCR3).”

Source and destination address registers (SAR and DAR) can be programmed in the DCR

to increment at the completion of a successful transfer. BCR decrements when an address transfer write completes for a single-address access (DCR[SAA] = 0) or when SAA = 1.

12.5.1 Transfer Requests (Cycle-Steal and Continuous Modes)

The DMA channel supports internal and external requests. A request is issued by setting DCR[START] or by asserting $\overline{\text{DREQ}}$. Setting DCR[EEXT] enables recognition of external interrupts. Internal interrupts are always recognized. Bus usage is minimized for either internal or external requests by selecting between cycle-steal and continuous modes.

- Cycle-steal mode (DCR[CS] = 1)—Only one complete transfer from source to destination occurs for each request. If DCR[EEXT] is set, a request can be either internal or external. Internal request is selected by setting DCR[START]. An external request is initiated by asserting $\overline{\text{DREQ}}$ while EEXT is set.
- Continuous mode (DCR[CS] = 0)—After an internal or external request, the DMA continuously transfers data until BCR reaches zero or a multiple of DCR[BWC] or DSR[DONE] is set. If BCR 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[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 the BCR 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 minimum time that the DMA loses bus control is one bus cycle.

12.5.2 Data Transfer Modes

Each channel supports dual- and single-address transfers, described in the next sections.

12.5.2.1 Dual-Address Transfers

Dual-address transfers consist of a source operand read and a destination operand write. The DMA controller module begins a dual-address transfer sequence when DCR[SAA] is cleared during a DMA request. If no error condition exists, DSR[REQ] is set.

- Dual-address read—The DMA controller drives the SAR value onto the internal address bus. If DCR[SINC] is set, the SAR 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 wider than the source), the DMA initiates the write portion of the transfer.

  If a termination error occurs, DSR[BES,DONE] are set and DMA transactions stop.

- Dual-address write—The DMA controller drives the DAR value onto the address bus. If DCR[DINC] is set, DAR increments by the appropriate number of bytes at the completion of a successful write cycle. The BCR decrements by the appropriate number of bytes. DSR[DONE] is set when BCR reaches zero. If the BCR is greater than zero, another read/write transfer is initiated. If the BCR is a multiple of DCR[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[BES,DONE] are set and DMA transactions stop.

12.5.2.2 Single-Address Transfers

Single-address transfers consist of one DMA bus cycle, allowing either a read or a write cycle to occur. The DMA controller begins a single-address transfer sequence when DCR[SAA] is set during a DMA request. If no error condition exists, DSR[REQ] is set. When the channel is enabled, DSR[BSY] is set and REQ is cleared. SAR contents are then driven onto the address bus and the value of DCR[S_RW] is driven on R/$\overline{W}$. The BCR decrements on each successful address access until it is zero, when DSR[DONE] is set.

If a termination error occurs, DSR[BES,DONE] are set and DMA transactions stop.

12.5.3 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.

12.5.3.1 Channel Prioritization

The four DMA channels are prioritized in ascending order (channel 0 having highest priority and channel 3 having the lowest) or as determined by DCR[BWC]. If BWC 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 DCR1[BWC] = DCR2[BWC] = 000, DMA 1 has priority over DMA 0 and DMA 2. DCR2[BWC] = 000 in this case does not affect prioritization.

Prioritization of simultaneous external requests is either ascending or as determined by each channel's BWC bits as described in the previous paragraphs.

12.5.3.2 Programming the DMA Controller Module

Note the following general guidelines for programming the DMA:

- No mechanism exists to prevent writes to control registers during DMA accesses.
- If the BWC of sequential channels are equal, channel priority is in ascending order.

The SAR 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. In single-address mode, this data register is used regardless of transfer direction.

The DAR should contain the destination (write) address. If the transfer is from a peripheral device to memory, or memory to memory, the DAR is loaded with the starting address of the data block to be written. If the transfer is from memory to a peripheral device, DAR is loaded with the address of the peripheral data register. This address can be any aligned byte address. DAR is not used in single-address mode.

SAR and DAR change after each cycle depending on DCR[SSIZE,DSIZE,SINC,DINC] and on the starting address. Increment values can be 1, 2, 4, or 16 for byte, word, longword, or 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 must be cleared for channel startup.

As soon as the channel has been initialized, it is started by writing a one to DCR[START] or asserting $\overline{\text{DREQ}}$, depending on the status of DCR[EEXT]. Programming the channel for internal request causes the channel to request the bus and start transferring data immediately. If the channel is programmed for external request, $\overline{\text{DREQ}}$ must be asserted before the channel requests the bus.

Changes to DCR are effective immediately while the channel is active. To avoid problems with changing a DMA channel setup, write a one to DSR[DONE] to stop the DMA channel.

12.5.4 Data Transfer

This section includes timing diagrams that illustrate the interaction of signals in DMA data transfers. It also describes auto-alignment and bandwidth control.

12.5.4.1 External Request and Acknowledge Operation

Channels 0 and 1 initiate transfers to an external module by means of $\overline{\text{DREQ}}$[1:0]. The request for channels 2 and 3 are connected internally to the UART0 and UART1 interrupt signals, respectively. If DCR[EEXT] = 1 and the channel is idle, the DMA initiates a transfer when $\overline{\text{DREQ}}$ is asserted.

Figure 12-11 shows the minimum 4-clock cycle delay from when $\overline{\text{DREQ}}$ is sampled asserted to when a DMA bus cycle begins. This delay may be longer, depending on DMA priority, bus arbitration, DRAM refresh operations, and other factors.

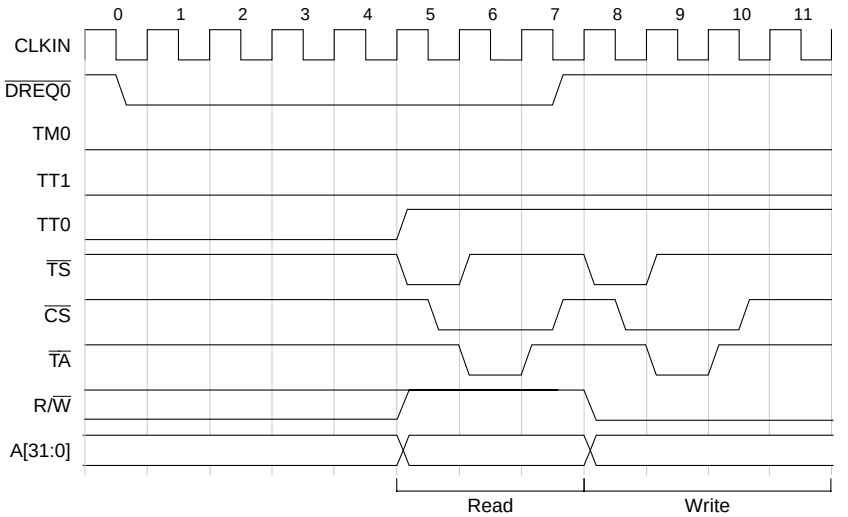


Figure 12-11. DREQ Timing Constraints, Dual-Address DMA Transfer

Although Figure 12-11 does not show TM0 signaling a DMA acknowledgement, this signal can provide an external request acknowledge response, as shown in subsequent diagrams.

To initiate a request, DREQ need only be asserted long enough to be sampled on one rising clock edge. However, note the following regarding the negation of DREQ:

- In cycle-steal mode (DCR[CS] = 1), the read/write transaction is limited to a single transfer. DREQ must be negated appropriately to avoid generating another request.
  - For dual-address transfers, DREQ must be negated before TS is asserted for the write portion, as shown in Figure 12-11, clock cycle 7.
  - For single-address transfers, DREQ must be negated before TS is asserted for the transfer, as shown in Figure 12-13, clock cycle 4.
- In burst mode, (DCR[CS] = 0), multiple read/write transfers can occur on the bus as programmed. DREQ need not be negated until DSR[DONE] is set, indicating the block transfer is complete. Another transfer cannot be initiated until the DMA registers are reprogrammed.

Figure 12-12 shows a dual-address, peripheral-to-SDRAM DMA transfer. The DMA is not parked on the bus, so the diagram shows how the CPU can generate multiple bus cycles during DMA transfers. It also shows TM0 timing. The TT signals indicate whether the CPU (0) or DMA (1) has bus mastership. TM2 indicates dual-address mode.

If DCR[AT] is 1, TM is asserted during the final transfer. If DCR[AT] is 0, TM asserts during all DMA accesses.

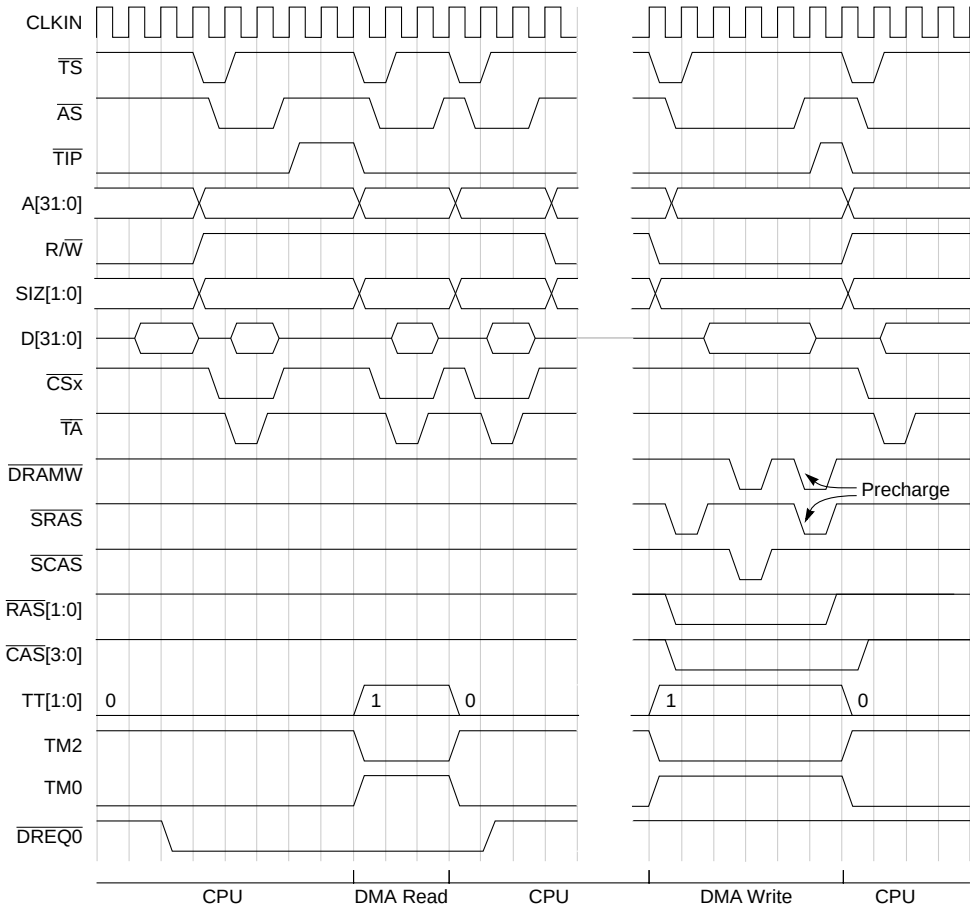


Figure 12-12. Dual-Address, Peripheral-to-SDRAM, Lower-Priority DMA Transfer

Figure 12-13 shows a single-address DMA transfer in which the peripheral is reading from memory. Note that TM2 is high, indicating a single-address transfer. Note that $\overline{\text{DREQ}}$ is negated in clock 4, before the assertion of $\overline{\text{TS}}$ in clock 6.

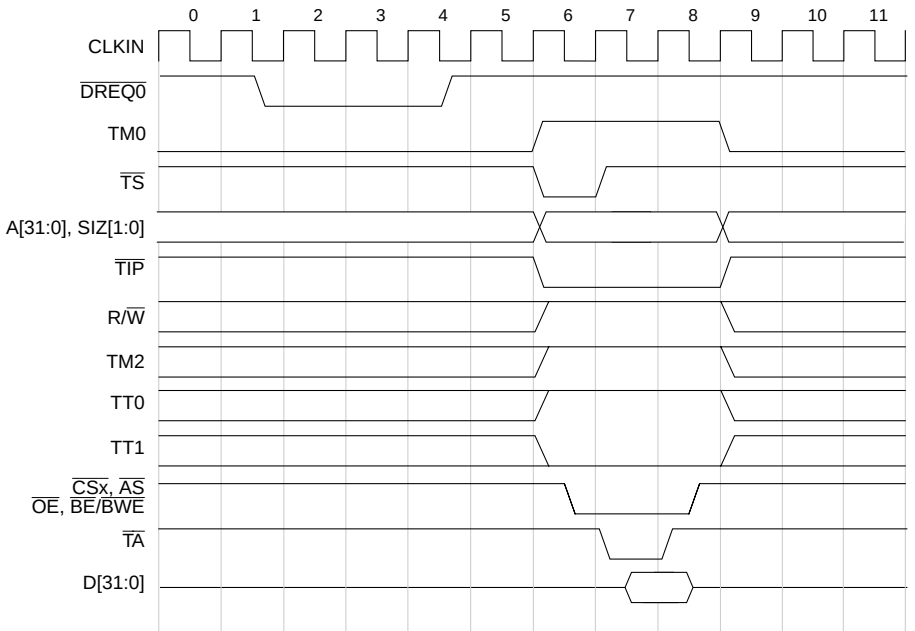


Figure 12-13. Single-Address DMA Transfer

12.5.4.2 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[AA] must be set. The source is auto-aligned if SSIZE indicates a transfer size larger than 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 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 is less than 16 at the start of a transfer, the number of bytes remaining dictates transfer size. For example, AA = 1, SAR = 0x0001, BCR = 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.
2. Read word from 0x0002—write 2 bytes, increment SAR.
3. Read longword from 0x0004—write 4 bytes, increment SAR.

4. Repeat longwords until SAR = 0x00F0.
5. Read byte from 0x00F0—write byte, increment SAR.

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.

12.5.4.3 Bandwidth Control

Bandwidth control makes it possible to force the DMA off the bus to allow access to another device. DCR[BWC] provides seven levels of block transfer sizes. If the BCR 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[AA] = 1, the BCR 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 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 6.2.10.1, “Default Bus Master Park Register (MPARK).”

12.5.5 Termination

An unsuccessful transfer can terminate for one of the following reasons:

- Error conditions—When the MCF5307 encounters a read or write cycle that terminates with an error condition, DSR[BES] is set for a read and DSR[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[INT] is set, the DMA drives the appropriate internal interrupt signal. The processor can read DSR to determine whether the transfer terminated successfully or with an error. DSR[DONE] is then written with a one to clear the interrupt and the DONE and error bits.

Chapter 13
Timer Module

This chapter describes the configuration and operation of the two general-purpose timer modules (timer 0 and timer 1). It includes programming examples.

13.1 Overview

The timer module incorporates two independent, general-purpose 16-bit timers, timer 0 and timer 1. The output of an 8-bit prescaler clocks each timer. There are two sets of registers, one for each timer. The timers can operate from the system bus clock (BCLKO) or from an external clocking source using one of the TIN signals. If BCLKO is selected, it can be divided by 16 or 1.

Figure 13-1 is a block diagram of one of the two identical ti5mer modules.

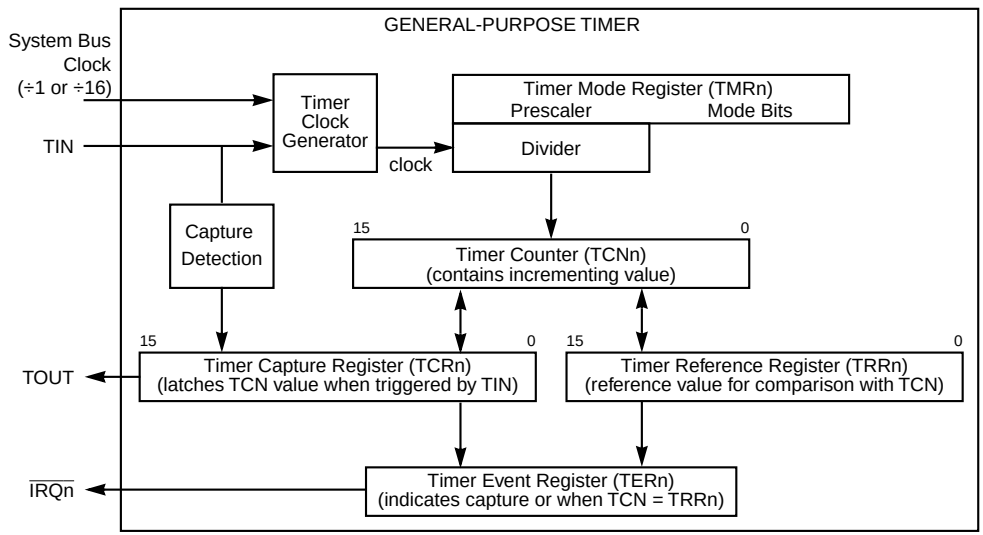


Figure 13-1. Timer Block Diagram

13.1.1 Key Features

Each general-purpose 16-bit timer unit has the following features:

- Maximum period of 5.96 seconds at 45 MHz
- 27-nS resolution at 45 MHz
- Programmable sources for the clock input, including external clock
- 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

13.2 General-Purpose Timer Units

The general-purpose timer units provide the following features:

- Each timer can be programmed to count and compare to a reference value stored in a register or capture the timer value at an edge detected on TIN.
- System bus clock can be divided by 16 or 1. This clock is input to the prescaler.
- TIN is fed directly into the 8-bit prescaler. The maximum value of TIN is 1/5 of CLKIN, as described in Chapter 20, “Electrical Specifications.”
- The 8-bit prescaler clock divides the clocking source and is user-programmable from 1 to 256.
- Programmed events generate interrupts.
- The timer output signal (TOUT) can be configured to toggle or pulse on an event.

13.3 General-Purpose Timer Programming Model

The following features are programmable through the timer registers, shown in Table 13-1:

- Prescaler—The prescaler clock input is selected from BCLKO (divided by 1 or 16) or from the corresponding timer input, TIN. TIN is synchronized to BCLKO. The synchronization delay is between two and three BCLKO clocks. The corresponding TMR*n*[ICLK] selects the clock input source. A programmable prescaler divides the clock input by values from 1 to 256. The prescaler is an input to the 16-bit counter.
- Capture mode—Each timer has a 16-bit timer capture register (TCR0 and TCR1) that latches the counter value when the corresponding input capture edge detector senses a defined TIN transition. The capture edge bits (TMR*n*[CE]) select the type of transition that triggers the capture, sets the timer event register capture event bit, TER*n*[CAP], and issues a maskable interrupt.

- Reference compare—A timer can be configured to count up to a reference value, at which point TER*n*[REF] is set. If TMR*n*[ORI] is one, an interrupt is issued. If the free run/restart bit TMR*n*[FRR] is set, a new count starts. If it is clear, the timer keeps running.
- Output mode—When a timer reaches the reference value selected by TMR*n*[OM], it can send an output signal on TOUT*n*. TOUT*n* can be an active-low pulse or a toggle of the current output under program control.

NOTE:

Although external devices cannot access MCF5307 on-chip memories or MBAR, they can access timer module registers.

The timer module registers, shown in Table 13-1, can be modified at any time.

Table 13-1. General-Purpose Timer Module Memory Map

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x140	Timer 0 mode register (TMR0) [p. 13-3]		Reserved	
0x144	Timer 0 reference register (TRR0) [p. 13-4]		Reserved	
0x148	Timer 0 capture register (TCR0) [p. 13-4]		Reserved	
0x14C	Timer 0 counter (TCN0) [p. 13-5]		Reserved	
0x150	Reserved	Timer 0 event register (TER0) [p. 13-5]	Reserved	
0x180	Timer 1 mode register (TMR1) [p. 13-3]		Reserved	
0x184	Timer 1 reference register (TRR1) [p. 13-4]		Reserved	
0x188	Timer 1 capture register (TCR1) [p. 13-4]		Reserved	
0x18C	Timer 1 counter (TCN1) [p. 13-5]		Reserved	
0x190	Reserved	Timer 1 event register (TER1) [p. 13-5]	Reserved	

13.3.1 Timer Mode Registers (TMR0/TMR1)

Timer mode registers (TMR0/TMR1), Figure 13-2, program the prescaler and various timer modes.

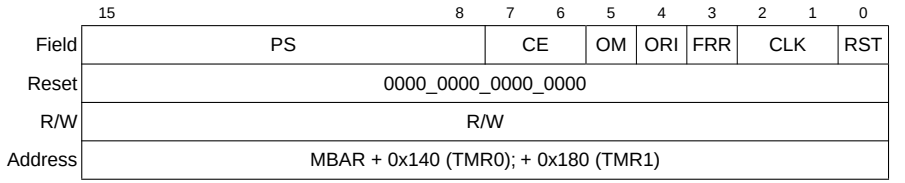


Figure 13-2. Timer Mode Registers (TMR0/TMR1)

Table 13-2 describes TMR*n* fields.

Table 13-2. TMR*n* Field Descriptions

Bits	Name	Description
15–8	PS	Prescaler value. The prescaler is programmed to divide the clock input (BCLKO/(16 or 1) or clock on TIN) by values from 1 (PS = 0000_0000) to 256 (PS = 1111_1111).
7–6	CE	Capture edge and enable interrupt 00 Disable interrupt on capture event 01 Capture on rising edge only and enable interrupt on capture event 10 Capture on falling edge only and enable interrupt on capture event 11 Capture on any edge and enable interrupt on capture event
5	OM	Output mode 0 Active-low pulse for one BCLKO cycle (22 nS at 45 MHz, 33 nS at 30 MHz, 44 nS at 22.5 MHz). 1 Toggle output.
4	ORI	Output reference interrupt enable. If ORI is set when TERn[REF] = 1, an interrupt occurs. 0 Disable interrupt for reference reached (does not affect interrupt on capture function). 1 Enable interrupt upon reaching the reference value.
3	FRR	Free run/restart 0 Free run. Timer count continues to increment after reaching the reference value. 1 Restart. Timer count is reset immediately after reaching the reference value.
2–1	CLK	Input clock source for the timer 00 Stop count 01 System bus clock divided by 1 10 System bus clock divided by 16. Note that this clock source is not synchronized to the timer; thus successive time-outs may vary slightly. 11 TIN pin (falling edge)
0	RST	Reset timer. Performs a software timer reset similar to an external reset, although other register values can still be written while RST = 0. A transition of RST from 1 to 0 resets register values. The timer counter is not clocked unless the timer is enabled. 0 Reset timer (software reset) 1 Enable timer

13.3.2 Timer Reference Registers (TRR0/TRR1)

Each timer reference register (TRR0/TRR1), Figure 13-3, contains the reference value compared with the respective free-running timer counter (TCN0/TCN1) as part of the output-compare function. The reference value is not matched until TCN*n* equals TRR*n*.

	15 … 0
Field	REF
Reset	1111_1111_1111_1111
R/W	R/W
Address	MBAR + 0x144 (TRR0),+ 0x184 (TRR1)

Figure 13-3. Timer Reference Registers (TRR0/TRR1)

13.3.3 Timer Capture Registers (TCR0/TCR1)

Each timer capture register (TCR0/TCR1), Figure 13-4, latches the corresponding TCN*n* value during a capture operation when an edge occurs on TIN, as programmed in TMR*n*.

BCLKO is assumed to be the clock source. TIN cannot simultaneously function as a clocking source and as an input capture pin.

	15	0
Field	CAP (16-bit capture counter value)	
Reset	0000_0000_0000_0000	
R/W	Read only	
Address	MBAR + 0x148 (TCR0); + 0x188 (TCR1)	

Figure 13-4. Timer Capture Register (TCR0/TCR1)

13.3.4 Timer Counters (TCN0/TCN1)

The current value of the 16-bit, incrementing timer counters (TCN0/TCN1), Figure 13-5, can be read anytime without affecting counting. Writing to TCN*n* clears it. The timer counter decrements on the clock source rising edge (BCLKO ÷ 1, BCLKO ÷ 16, or TIN).

	15	0
Field	16-bit timer counter value count	
Reset	0000_0000_0000_0000	
R/W	R/W (to reset)	
Address	MBAR + 0x14C (TCN0); + 0x18C (TCN1)	

Figure 13-5. Timer Counters (TCN0/TCN1)

13.3.5 Timer Event Registers (TER0/TER1)

Each timer event register (TER0/TER1), Figure 13-6, reports capture or reference events events the timer recognizes by setting TER*n*[CAP] or TER*n*[REF], which it does regardless of the corresponding interrupt-enable bit values, TMR*n*[ORI,CE].

Writing a 1 to either REF or CAP clears it (writing a 0 does not affect bit value); both bits can be cleared at the same time. REF and CAP must be cleared early in the exception handler, before the timer negates the $\overline{\text{IRQ}n}$ to the interrupt controller.

	7	2	1	0
Field	—		REF	CAP
Reset	0000_0000			
R/W	R/W (ones clear/zeros have no effect)			
Address	MBAR + 0x151 (TER0); + 0x191 (TER1)			

Figure 13-6. Timer Event Registers (TER0/TER1)

Table 13-3 describes TER*n* fields.

Table 13-3. TER*n* Field Descriptions

Bits	Name	Description
7–2	—	Reserved
1	REF	Output reference event. The counter has reached the TRRn value. Setting TMRn[ORI] enables the interrupt request caused by this event. Writing a one to REF clears the event condition.
0	CAP	Capture event. The counter value has been latched into TCRn. Setting TMRn[CE] enables the interrupt request caused by this event. Writing a 1 to CAP clears the event condition.

13.4 Code Example

The following code provides an example of how to initialize timer 0 and how to use the timer for counting time-out periods.

```
MBARx EQU 0x10000 ;Defines the module base address at 0x10000
TMR0 EQU MBARx+0x140;Timer 0 register
TMR1 EQU MBARx+0x180 ;Timer 1 register
TRR0 EQU MBARx+0x144 ;Timer 0 reference register
TRR1 EQU MBARx+0x184 ;Timer 1 reference register
TCR0 EQU MBARx+0x148 ;Timer 0 capture register
TCR1 EQU MBARx+0x188 ;Timer 1 capture register
TCN0 EQU MBARx+0x14C ;Timer 0 counter
TCN1 EQU MBARx+0x18C ;Timer 1 counter
TER0 EQU MBARx+0x151 ;Timer 0 event register
TER1 EQU MBARx+0x191 ;Timer 1 event register

* TMR0 is defined as: *
*[PS]= 0xFF, divide clock by 256
*[CE] = 00disable interrupt
*[OM] = 0 output=active-low pulse
*[ORI] = 0, disable ref.interrupt
*[FRR] = 1, restart mode enabled
*[CLK] = 10, BCLKO/16
*[RST] = 0, timer 0 disabled

    move.w #0xFF0C,D0
    move.w D0,TMR0

    move.w #0x0000,D0;writing to the timer counter with any
    move.w DO,TCN0 ;value resets it to zero

    move.w #AFAF,DO ;set the timer 0 reference to be
    move.w #D0,TRR0 ;defined as 0xAFAF
```

The simple example below uses 0 to count time-out loops. A time-out occurs when the reference value, 0xAFAF, is reached.

```
timer0_ex
    clr.l DO
    clr.l D1
    clt.l D2

    move.w #0x0000,D0
    move,w 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
```

13.5 Calculating Time-Out Values

The formula below determines time-out periods for various reference values:

Time-out period = (1/clock frequency) x (1 or 16) x (TMR*n*[PS] + 1) x (TRR*n*[REF])

When calculating time-out periods, add 1 to the prescaler to simplify calculating, because TMR*n*[PS] = 0x00 yields a prescaler of 1 and TMR*n*[PS] = 0xFF yields a prescaler of 256. For example, if a 45-MHz timer clock is divided by 16, TMR*n*[PS] = 0x7F, and the timer is referenced at 0xABCD (43,981 decimal), the time-out period is as follows:

Time-out period = (1/45) x (16) x (127 + 1) x (43,981) = 1.67 S

The time-out values in Table 13-5 represent the time it takes the counter value in TCN*n* value to go from 0x0000 to the default reference value, TRR*n*[REF] = 0xFFFF. Time-out values shown for BCLKO are divided by 1 and by 16 (TMR*n*[CLK] is 01 or 10, respectively).

Any clock source (BCLKO ÷ 1, BCLKO ÷ 16, or TIN) can be prescaled using TMR*n*[PS].

The BCLKO frequency depends on the prescaler value (TMR*n*[PS]) and on the PLL clock setting, as described inChapter 7, “Phase-Locked Loop (PLL).”

Table 13-5. Calculated Time-out Values (90-MHz Processor Clock)

TMR[PS]		TMR[CLK] = 10 (System Bus Clock/16)			TMR[CLK] = 01 (System Bus Clock/1)		
Decimal	Hex	45 MHz	30 MHz	22.5 MHz	45 MHz	30 MHz	22.5 MHz
0	0	0.0233	0.03495	0.0466	0.00146	0.00218	0.00291
1	1	0.0466	0.06991	0.09321	0.00291	0.00437	0.00583
2	2	0.06991	0.10486	0.13981	0.00437	0.00655	0.00874
3	3	0.09321	0.13981	0.18641	0.00583	0.00874	0.01165
4	4	0.11651	0.17476	0.23302	0.00728	0.01092	0.01456
5	5	0.13981	0.20972	0.27962	0.00874	0.01311	0.01748

Table 13-5. Calculated Time-out Values (90-MHz Processor Clock) (Continued)

TMR[PS]		TMR[CLK] = 10 (System Bus Clock/16)			TMR[CLK] = 01 (System Bus Clock/1)		
Decimal	Hex	45 MHz	30 MHz	22.5 MHz	45 MHz	30 MHz	22.5 MHz
6	6	0.16311	0.24467	0.32622	0.01019	0.01529	0.02039
7	7	0.18641	0.27962	0.37283	0.01165	0.01748	0.0233
8	8	0.20972	0.31457	0.41943	0.01311	0.01966	0.02621
9	9	0.23302	0.34953	0.46603	0.01456	0.02185	0.02913
10	0A	0.25632	0.38448	0.51264	0.01602	0.02403	0.03204
11	0B	0.27962	0.41943	0.55924	0.01748	0.02621	0.03495
12	0C	0.30292	0.45438	0.60584	0.01893	0.0284	0.03787
13	0D	0.32622	0.48934	0.65245	0.02039	0.03058	0.04078
14	0E	0.34953	0.52429	0.69905	0.02185	0.03277	0.04369
15	0F	0.37283	0.55924	0.74565	0.0233	0.03495	0.0466
16	10	0.39613	0.59419	0.79226	0.02476	0.03714	0.04952
17	11	0.41943	0.62915	0.83886	0.02621	0.03932	0.05243
18	12	0.44273	0.6641	0.88546	0.02767	0.04151	0.05534
19	13	0.46603	0.69905	0.93207	0.02913	0.04369	0.05825
20	14	0.48934	0.734	0.97867	0.03058	0.04588	0.06117
21	15	0.51264	0.76896	1.02527	0.03204	0.04806	0.06408
22	16	0.53594	0.80391	1.07188	0.0335	0.05024	0.06699
23	17	0.55924	0.83886	1.11848	0.03495	0.05243	0.06991
24	18	0.58254	0.87381	1.16508	0.03641	0.05461	0.07282
25	19	0.60584	0.90877	1.21169	0.03787	0.0568	0.07573
26	1A	0.62915	0.94372	1.25829	0.03932	0.05898	0.07864
27	1B	0.65245	0.97867	1.30489	0.04078	0.06117	0.08156
28	1C	0.67575	1.01362	1.3515	0.04223	0.06335	0.08447
29	1D	0.69905	1.04858	1.3981	0.04369	0.06554	0.08738
30	1E	0.72235	1.08353	1.4447	0.04515	0.06772	0.09029
31	1F	0.74565	1.11848	1.49131	0.0466	0.06991	0.09321
32	20	0.76896	1.15343	1.53791	0.04806	0.07209	0.09612
33	21	0.79226	1.18839	1.58451	0.04952	0.07427	0.09903
34	22	0.81556	1.22334	1.63112	0.05097	0.07646	0.10194
35	23	0.83886	1.25829	1.67772	0.05243	0.07864	0.10486
36	24	0.86216	1.29324	1.72432	0.05389	0.08083	0.10777
37	25	0.88546	1.3282	1.77093	0.05534	0.08301	0.11068
38	26	0.90877	1.36315	1.81753	0.0568	0.0852	0.1136
39	27	0.93207	1.3981	1.86414	0.05825	0.08738	0.11651
40	28	0.95537	1.43305	1.91074	0.05971	0.08957	0.11942
41	29	0.97867	1.46801	1.95734	0.06117	0.09175	0.12233
42	2A	1.00197	1.50296	2.00395	0.06262	0.09393	0.12525
43	2B	1.02527	1.53791	2.05055	0.06408	0.09612	0.12816
44	2C	1.04858	1.57286	2.09715	0.06554	0.0983	0.13107
45	2D	1.07188	1.60782	2.14376	0.06699	0.10049	0.13398

Table 13-5. Calculated Time-out Values (90-MHz Processor Clock) (Continued)

TMR[PS]		TMR[CLK] = 10 (System Bus Clock/16)			TMR[CLK] = 01 (System Bus Clock/1)		
Decimal	Hex	45 MHz	30 MHz	22.5 MHz	45 MHz	30 MHz	22.5 MHz
46	2E	1.09518	1.64277	2.19036	0.06845	0.10267	0.1369
47	2F	1.11848	1.67772	2.23696	0.06991	0.10486	0.13981
48	30	1.14178	1.71267	2.28357	0.07136	0.10704	0.14272
49	31	1.16508	1.74763	2.33017	0.07282	0.10923	0.14564
50	32	1.18839	1.78258	2.37677	0.07427	0.11141	0.14855
51	33	1.21169	1.81753	2.42338	0.07573	0.1136	0.15146
52	34	1.23499	1.85248	2.46998	0.07719	0.11578	0.15437
53	35	1.25829	1.88744	2.51658	0.07864	0.11796	0.15729
54	36	1.28159	1.92239	2.56319	0.0801	0.12015	0.1602
55	37	1.30489	1.95734	2.60979	0.08156	0.12233	0.16311
56	38	1.3282	1.99229	2.65639	0.08301	0.12452	0.16602
57	39	1.3515	2.02725	2.703	0.08447	0.1267	0.16894
58	3A	1.3748	2.0622	2.7496	0.08592	0.12889	0.17185
59	3B	1.3981	2.09715	2.7962	0.08738	0.13107	0.17476
60	3C	1.4214	2.1321	2.84281	0.08884	0.13326	0.17768
61	3D	1.4447	2.16706	2.88941	0.09029	0.13544	0.18059
62	3E	1.46801	2.20201	2.93601	0.09175	0.13763	0.1835
63	3F	1.49131	2.23696	2.98262	0.09321	0.13981	0.18641
64	40	1.51461	2.27191	3.02922	0.09466	0.14199	0.18933
65	41	1.53791	2.30687	3.07582	0.09612	0.14418	0.19224
66	42	1.56121	2.34182	3.12243	0.09758	0.14636	0.19515
67	43	1.58451	2.37677	3.16903	0.09903	0.14855	0.19806
68	44	1.60782	2.41172	3.21563	0.10049	0.15073	0.20098
69	45	1.63112	2.44668	3.26224	0.10194	0.15292	0.20389
70	46	1.65442	2.48163	3.30884	0.1034	0.1551	0.2068
71	47	1.67772	2.51658	3.35544	0.10486	0.15729	0.20972
72	48	1.70102	2.55153	3.40205	0.10631	0.15947	0.21263
73	49	1.72432	2.58649	3.44865	0.10777	0.16166	0.21554
74	4A	1.74763	2.62144	3.49525	0.10923	0.16384	0.21845
75	4B	1.77093	2.65639	3.54186	0.11068	0.16602	0.22137
76	4C	1.79423	2.69135	3.58846	0.11214	0.16821	0.22428
77	4D	1.81753	2.7263	3.63506	0.1136	0.17039	0.22719
78	4E	1.84083	2.76125	3.68167	0.11505	0.17258	0.2301
79	4F	1.86414	2.7962	3.72827	0.11651	0.17476	0.23302
80	50	1.88744	2.83116	3.77487	0.11796	0.17695	0.23593
81	51	1.91074	2.86611	3.82148	0.11942	0.17913	0.23884
82	52	1.93404	2.90106	3.86808	0.12088	0.18132	0.24176
83	53	1.95734	2.93601	3.91468	0.12233	0.1835	0.24467
84	54	1.98064	2.97097	3.96129	0.12379	0.18569	0.24758
85	55	2.00395	3.00592	4.00789	0.12525	0.18787	0.25049

Table 13-5. Calculated Time-out Values (90-MHz Processor Clock) (Continued)

TMR[PS]		TMR[CLK] = 10 (System Bus Clock/16)			TMR[CLK] = 01 (System Bus Clock/1)		
Decimal	Hex	45 MHz	30 MHz	22.5 MHz	45 MHz	30 MHz	22.5 MHz
86	56	2.02725	3.04087	4.05449	0.1267	0.19005	0.25341
87	57	2.05055	3.07582	4.1011	0.12816	0.19224	0.25632
88	58	2.07385	3.11078	4.1477	0.12962	0.19442	0.25923
89	59	2.09715	3.14573	4.1943	0.13107	0.19661	0.26214
90	5A	2.12045	3.18068	4.24091	0.13253	0.19879	0.26506
91	5B	2.14376	3.21563	4.28751	0.13398	0.20098	0.26797
92	5C	2.16706	3.25059	4.33411	0.13544	0.20316	0.27088
93	5D	2.19036	3.28554	4.38072	0.1369	0.20535	0.27379
94	5E	2.21366	3.32049	4.42732	0.13835	0.20753	0.27671
95	5F	2.23696	3.35544	4.47392	0.13981	0.20972	0.27962
96	60	2.26026	3.3904	4.52053	0.14127	0.2119	0.28253
97	61	2.28357	3.42535	4.56713	0.14272	0.21408	0.28545
98	62	2.30687	3.4603	4.61373	0.14418	0.21627	0.28836
99	63	2.33017	3.49525	4.66034	0.14564	0.21845	0.29127
100	64	2.35347	3.53021	4.70694	0.14709	0.22064	0.29418
101	65	2.37677	3.56516	4.75354	0.14855	0.22282	0.2971
102	66	2.40007	3.60011	4.80015	0.15	0.22501	0.30001
103	67	2.42338	3.63506	4.84675	0.15146	0.22719	0.30292
104	68	2.44668	3.67002	4.89335	0.15292	0.22938	0.30583
105	69	2.46998	3.70497	4.93996	0.15437	0.23156	0.30875
106	6A	2.49328	3.73992	4.98656	0.15583	0.23375	0.31166
107	6B	2.51658	3.77487	5.03316	0.15729	0.23593	0.31457
108	6C	2.53988	3.80983	5.07977	0.15874	0.23811	0.31749
109	6D	2.56319	3.84478	5.12637	0.1602	0.2403	0.3204
110	6E	2.58649	3.87973	5.17297	0.16166	0.24248	0.32331
111	6F	2.60979	3.91468	5.21958	0.16311	0.24467	0.32622
112	70	2.63309	3.94964	5.26618	0.16457	0.24685	0.32914
113	71	2.65639	3.98459	5.31279	0.16602	0.24904	0.33205
114	72	2.67969	4.01954	5.35939	0.16748	0.25122	0.33496
115	73	2.703	4.05449	5.40599	0.16894	0.25341	0.33787
116	74	2.7263	4.08945	5.4526	0.17039	0.25559	0.34079
117	75	2.7496	4.1244	5.4992	0.17185	0.25777	0.3437
118	76	2.7729	4.15935	5.5458	0.17331	0.25996	0.34661
119	77	2.7962	4.1943	5.59241	0.17476	0.26214	0.34953
120	78	2.8195	4.22926	5.63901	0.17622	0.26433	0.35244
121	79	2.84281	4.26421	5.68561	0.17768	0.26651	0.35535
122	7A	2.86611	4.29916	5.73222	0.17913	0.2687	0.35826
123	7B	2.88941	4.33411	5.77882	0.18059	0.27088	0.36118
124	7C	2.91271	4.36907	5.82542	0.18204	0.27307	0.36409
125	7D	2.93601	4.40402	5.87203	0.1835	0.27525	0.367

Table 13-5. Calculated Time-out Values (90-MHz Processor Clock) (Continued)

TMR[PS]		TMR[CLK] = 10 (System Bus Clock/16)			TMR[CLK] = 01 (System Bus Clock/1)		
Decimal	Hex	45 MHz	30 MHz	22.5 MHz	45 MHz	30 MHz	22.5 MHz
126	7E	2.95931	4.43897	5.91863	0.18496	0.27744	0.36991
127	7F	2.98262	4.47392	5.96523	0.18641	0.27962	0.37283
128	80	3.00592	4.50888	6.01184	0.18787	0.2818	0.37574
129	81	3.02922	4.54383	6.05844	0.18933	0.28399	0.37865
130	82	3.05252	4.57878	6.10504	0.19078	0.28617	0.38157
131	83	3.07582	4.61373	6.15165	0.19224	0.28836	0.38448
132	84	3.09912	4.64869	6.19825	0.1937	0.29054	0.38739
133	85	3.12243	4.68364	6.24485	0.19515	0.29273	0.3903
134	86	3.14573	4.71859	6.29146	0.19661	0.29491	0.39322
135	87	3.16903	4.75354	6.33806	0.19806	0.2971	0.39613
136	88	3.19233	4.7885	6.38466	0.19952	0.29928	0.39904
137	89	3.21563	4.82345	6.43127	0.20098	0.30147	0.40195
138	8A	3.23893	4.8584	6.47787	0.20243	0.30365	0.40487
139	8B	3.26224	4.89335	6.52447	0.20389	0.30583	0.40778
140	8C	3.28554	4.92831	6.57108	0.20535	0.30802	0.41069
141	8D	3.30884	4.96326	6.61768	0.2068	0.3102	0.4136
142	8E	3.33214	4.99821	6.66428	0.20826	0.31239	0.41652
143	8F	3.35544	5.03316	6.71089	0.20972	0.31457	0.41943
144	90	3.37874	5.06812	6.75749	0.21117	0.31676	0.42234
145	91	3.40205	5.10307	6.80409	0.21263	0.31894	0.42526
146	92	3.42535	5.13802	6.8507	0.21408	0.32113	0.42817
147	93	3.44865	5.17297	6.8973	0.21554	0.32331	0.43108
148	94	3.47195	5.20793	6.9439	0.217	0.3255	0.43399
149	95	3.49525	5.24288	6.99051	0.21845	0.32768	0.43691
150	96	3.51856	5.27783	7.03711	0.21991	0.32986	0.43982
151	97	3.54186	5.31279	7.08371	0.22137	0.33205	0.44273
152	98	3.56516	5.34774	7.13032	0.22282	0.33423	0.44564
153	99	3.58846	5.38269	7.17692	0.22428	0.33642	0.44856
154	9A	3.61176	5.41764	7.22352	0.22574	0.3386	0.45147
155	9B	3.63506	5.4526	7.27013	0.22719	0.34079	0.45438
156	9C	3.65837	5.48755	7.31673	0.22865	0.34297	0.4573
157	9D	3.68167	5.5225	7.36333	0.2301	0.34516	0.46021
158	9E	3.70497	5.55745	7.40994	0.23156	0.34734	0.46312
159	9F	3.72827	5.59241	7.45654	0.23302	0.34953	0.46603
160	A0	3.75157	5.62736	7.50314	0.23447	0.35171	0.46895
161	A1	3.77487	5.66231	7.54975	0.23593	0.35389	0.47186
162	A2	3.79818	5.69726	7.59635	0.23739	0.35608	0.47477
163	A3	3.82148	5.73222	7.64295	0.23884	0.35826	0.47768
164	A4	3.84478	5.76717	7.68956	0.2403	0.36045	0.4806
165	A5	3.86808	5.80212	7.73616	0.24176	0.36263	0.48351

Table 13-5. Calculated Time-out Values (90-MHz Processor Clock) (Continued)

TMR[PS]		TMR[CLK] = 10 (System Bus Clock/16)			TMR[CLK] = 01 (System Bus Clock/1)		
Decimal	Hex	45 MHz	30 MHz	22.5 MHz	45 MHz	30 MHz	22.5 MHz
166	A6	3.89138	5.83707	7.78276	0.24321	0.36482	0.48642
167	A7	3.91468	5.87203	7.82937	0.24467	0.367	0.48934
168	A8	3.93799	5.90698	7.87597	0.24612	0.36919	0.49225
169	A9	3.96129	5.94193	7.92257	0.24758	0.37137	0.49516
170	AA	3.98459	5.97688	7.96918	0.24904	0.37356	0.49807
171	AB	4.00789	6.01184	8.01578	0.25049	0.37574	0.50099
172	AC	4.03119	6.04679	8.06238	0.25195	0.37792	0.5039
173	AD	4.05449	6.08174	8.10899	0.25341	0.38011	0.50681
174	AE	4.0778	6.11669	8.15559	0.25486	0.38229	0.50972
175	AF	4.1011	6.15165	8.20219	0.25632	0.38448	0.51264
176	B0	4.1244	6.1866	8.2488	0.25777	0.38666	0.51555
177	B1	4.1477	6.22155	8.2954	0.25923	0.38885	0.51846
178	B2	4.171	6.2565	8.342	0.26069	0.39103	0.52138
179	B3	4.1943	6.29146	8.38861	0.26214	0.39322	0.52429
180	B4	4.21761	6.32641	8.43521	0.2636	0.3954	0.5272
181	B5	4.24091	6.36136	8.48181	0.26506	0.39759	0.53011
182	B6	4.26421	6.39631	8.52842	0.26651	0.39977	0.53303
183	B7	4.28751	6.43127	8.57502	0.26797	0.40195	0.53594
184	B8	4.31081	6.46622	8.62162	0.26943	0.40414	0.53885
185	B9	4.33411	6.50117	8.66823	0.27088	0.40632	0.54176
186	BA	4.35742	6.53612	8.71483	0.27234	0.40851	0.54468
187	BB	4.38072	6.57108	8.76144	0.27379	0.41069	0.54759
188	BC	4.40402	6.60603	8.80804	0.27525	0.41288	0.5505
189	BD	4.42732	6.64098	8.85464	0.27671	0.41506	0.55342
190	BE	4.45062	6.67593	8.90125	0.27816	0.41725	0.55633
191	BF	4.47392	6.71089	8.94785	0.27962	0.41943	0.55924
192	C0	4.49723	6.74584	8.99445	0.28108	0.42161	0.56215
193	C1	4.52053	6.78079	9.04106	0.28253	0.4238	0.56507
194	C2	4.54383	6.81574	9.08766	0.28399	0.42598	0.56798
195	C3	4.56713	6.8507	9.13426	0.28545	0.42817	0.57089
196	C4	4.59043	6.88565	9.18087	0.2869	0.43035	0.5738
197	C5	4.61373	6.9206	9.22747	0.28836	0.43254	0.57672
198	C6	4.63704	6.95555	9.27407	0.28981	0.43472	0.57963
199	C7	4.66034	6.99051	9.32068	0.29127	0.43691	0.58254
200	C8	4.68364	7.02546	9.36728	0.29273	0.43909	0.58545
201	C9	4.70694	7.06041	9.41388	0.29418	0.44128	0.58837
202	CA	4.73024	7.09536	9.46049	0.29564	0.44346	0.59128
203	CB	4.75354	7.13032	9.50709	0.2971	0.44564	0.59419
204	CC	4.77685	7.16527	9.55369	0.29855	0.44783	0.59711
205	CD	4.80015	7.20022	9.6003	0.30001	0.45001	0.60002

Table 13-5. Calculated Time-out Values (90-MHz Processor Clock) (Continued)

TMR[PS]		TMR[CLK] = 10 (System Bus Clock/16)			TMR[CLK] = 01 (System Bus Clock/1)		
Decimal	Hex	45 MHz	30 MHz	22.5 MHz	45 MHz	30 MHz	22.5 MHz
206	CE	4.82345	7.23517	9.6469	0.30147	0.4522	0.60293
207	CF	4.84675	7.27013	9.6935	0.30292	0.45438	0.60584
208	D0	4.87005	7.30508	9.74011	0.30438	0.45657	0.60876
209	D1	4.89335	7.34003	9.78671	0.30583	0.45875	0.61167
210	D2	4.91666	7.37498	9.83331	0.30729	0.46094	0.61458
211	D3	4.93996	7.40994	9.87992	0.30875	0.46312	0.61749
212	D4	4.96326	7.44489	9.92652	0.3102	0.46531	0.62041
213	D5	4.98656	7.47984	9.97312	0.31166	0.46749	0.62332
214	D6	5.00986	7.51479	10.01973	0.31312	0.46967	0.62623
215	D7	5.03316	7.54975	10.06633	0.31457	0.47186	0.62915
216	D8	5.05647	7.5847	10.11293	0.31603	0.47404	0.63206
217	D9	5.07977	7.61965	10.15954	0.31749	0.47623	0.63497
218	DA	5.10307	7.6546	10.20614	0.31894	0.47841	0.63788
219	DB	5.12637	7.68956	10.25274	0.3204	0.4806	0.6408
220	DC	5.14967	7.72451	10.29935	0.32185	0.48278	0.64371
221	DD	5.17297	7.75946	10.34595	0.32331	0.48497	0.64662
222	DE	5.19628	7.79441	10.39255	0.32477	0.48715	0.64953
223	DF	5.21958	7.82937	10.43916	0.32622	0.48934	0.65245
224	E0	5.24288	7.86432	10.48576	0.32768	0.49152	0.65536
225	E1	5.26618	7.89927	10.53236	0.32914	0.4937	0.65827
226	E2	5.28948	7.93423	10.57897	0.33059	0.49589	0.66119
227	E3	5.31279	7.96918	10.62557	0.33205	0.49807	0.6641
228	E4	5.33609	8.00413	10.67217	0.33351	0.50026	0.66701
229	E5	5.35939	8.03908	10.71878	0.33496	0.50244	0.66992
230	E6	5.38269	8.07404	10.76538	0.33642	0.50463	0.67284
231	E7	5.40599	8.10899	10.81198	0.33787	0.50681	0.67575
232	E8	5.42929	8.14394	10.85859	0.33933	0.509	0.67866
233	E9	5.4526	8.17889	10.90519	0.34079	0.51118	0.68157
234	EA	5.4759	8.21385	10.95179	0.34224	0.51337	0.68449
235	EB	5.4992	8.2488	10.9984	0.3437	0.51555	0.6874
236	EC	5.5225	8.28375	11.045	0.34516	0.51773	0.69031
237	ED	5.5458	8.3187	11.0916	0.34661	0.51992	0.69323
238	EE	5.5691	8.35366	11.13821	0.34807	0.5221	0.69614
239	EF	5.59241	8.38861	11.18481	0.34953	0.52429	0.69905
240	F0	5.61571	8.42356	11.23141	0.35098	0.52647	0.70196
241	F1	5.63901	8.45851	11.27802	0.35244	0.52866	0.70488
242	F2	5.66231	8.49347	11.32462	0.35389	0.53084	0.70779
243	F3	5.68561	8.52842	11.37122	0.35535	0.53303	0.7107
244	F4	5.70891	8.56337	11.41783	0.35681	0.53521	0.71361
245	F5	5.73222	8.59832	11.46443	0.35826	0.5374	0.71653

Table 13-5. Calculated Time-out Values (90-MHz Processor Clock) (Continued)

TMR[PS]		TMR[CLK] = 10 (System Bus Clock/16)			TMR[CLK] = 01 (System Bus Clock/1)		
Decimal	Hex	45 MHz	30 MHz	22.5 MHz	45 MHz	30 MHz	22.5 MHz
246	F6	5.75552	8.63328	11.51103	0.35972	0.53958	0.71944
247	F7	5.77882	8.66823	11.55764	0.36118	0.54176	0.72235
248	F8	5.80212	8.70318	11.60424	0.36263	0.54395	0.72527
249	F9	5.82542	8.73813	11.65084	0.36409	0.54613	0.72818
250	FA	5.84872	8.77309	11.69745	0.36555	0.54832	0.73109
251	FB	5.87203	8.80804	11.74405	0.367	0.5505	0.734
252	FC	5.89533	8.84299	11.79065	0.36846	0.55269	0.73692
253	FD	5.91863	8.87794	11.83726	0.36991	0.55487	0.73983
254	FE	5.94193	8.9129	11.88386	0.37137	0.55706	0.74274
255	FF	5.96523	8.94785	11.93046	0.37283	0.55924	0.74565

Chapter 14
UART Modules

This chapter describes the use of the universal asynchronous/synchronous receiver/transmitters (UARTs) implemented on the MCF5307 and includes programming examples. All references to UART refer to one of these modules.

14.1 Overview

The MCF5307 contains two independent UARTs. Each UART can be clocked by BCLKO, eliminating the need for an external crystal. As Figure 14-1 shows, each UART module interfaces directly to the CPU and consists of the following:

- Serial communication channel
- Programmable transmitter and receiver clock generation
- Internal channel control logic
- Interrupt control logic

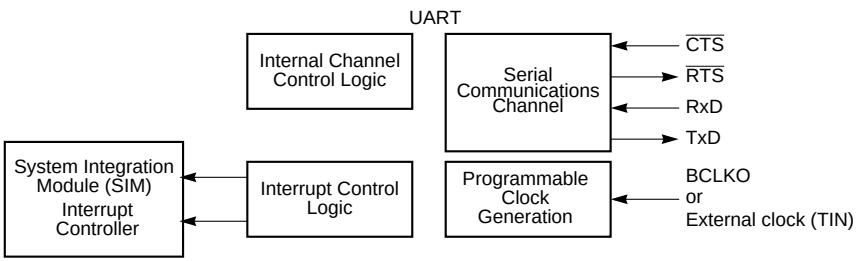


Figure 14-1. Simplified Block Diagram

The serial communication channel provides a full-duplex asynchronous/synchronous receiver and transmitter deriving an operating frequency from BCLKO or an external clock using the timer pin. The transmitter converts parallel data from the CPU to a serial bit stream, inserting appropriate start, stop, and parity bits. It outputs the resulting stream on the channel transmitter serial data output (TxD). See Section 14.5.2.1, “Transmitting.”

The receiver converts serial data from the channel receiver serial data input (RxD) to parallel format, checks for a start, stop, and parity bits, or break conditions, and transfers the assembled character onto the bus during read operations. The receiver may be polled- or interrupt-driven. See Section 14.5.2.2, “Receiver.”

14.2 Serial Module Overview

The MCF5307 contains two independent UART modules, whose features are as follows:

- Each can be clocked by BCLKO, eliminating a need for an external crystal
- 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
- UART0 and UART1 have interrupt capability to DMA channels 2 and 3, respectively, when either the RxRDY or FFULL bit is set in the USR.
- 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

14.3 Register Descriptions

This section contains a detailed description of each register and its specific function. Flowcharts in Section 14.5.6, “Programming,” describe basic UART module programming. The operation of the UART module is controlled by writing control bytes into the appropriate registers. Table 14-1 is a memory map for UART module registers.

Table 14-1. UART Module Programming Model

MBAR Offset		[31:24]	[23:16]	[15:8]	[7:0]
UART0	UART1				
0x1C0	0x200	UART mode registers[1]—(UMR1n) [p. 14-4], (UMR2n) [p. 14-6]	—		
0x1C4	0x204	(Read) UART status registers—(USRn) [p. 14-7]	—		
		(Write) UART clock-select register[1]—(UCSRn) [p. 14-8]	—		
0x1C8	0x208	(Read) Do not access[2]	—		
		(Write) UART command registers—(UCRn) [p. 14-9]	—		
0x1CC	0x20C	(UART/Read) UART receiver buffers—(URBn) [p. 14-11]	—		
		(UART/Write) UART transmitter buffers—(UTBn) [p. 14-11]	—		
0x1D0	0x210	(Read) UART input port change registers—(UIPCRn) [p. 14-12]	—		
		(Write) UART auxiliary control registers[1]—(UACRn) [p. 14-12]	—		
0x1D4	0x214	(Read) UART interrupt status registers—(UISRn) [p. 14-13]	—		
		(Write) UART interrupt mask registers—(UIMRn) [p. 14-13]	—		
0x1D8	0x218	UART divider upper registers—(UDUn) [p. 14-14]	—		
0x1DC	0x21C	UART divider lower registers—(UDLn) [p. 14-14]	—		

Table 14-1. UART Module Programming Model (Continued)

MBAR Offset		[31:24]	[23:16]	[15:8]	[7:0]
UART0	UART1				
0x1E0–0x1EC	0x220–0x22C	Do not access[2]	—		
0x1F0	0x230	UART interrupt vector register—(UIVRn) [p. 14-15]	—		
0x1F4	0x234	(Read) UART input port registers—(UIPn) [p. 14-15]	—		
		(Write) Do not access[2]	—		
0x1F8	0x238	(Read) Do not access[2]	—		
		(Write) UART output port bit set command registers—(UOP1n[3]) [p. 14-15]	—		
0x1FC	0x23C	(Read) Do not access[2]	—		
		(Write) UART output port bit reset command registers—(UOP0n[3]) [p. 14-15]	—		

1 UMR1n, UMR2n, and UCSRn should be changed only after the receiver/transmitter is issued a software reset command. That is, if channel operation is not disabled, undesirable results may occur.

2 This address is for factory testing. Reading this location results in undesired effects and possible incorrect transmission or reception of characters. Register contents may also be changed.

3 Address-triggered commands

NOTE:

UART registers are accessible only as bytes. Although external masters cannot access on-chip memories or MBAR, they can access any UART registers.

14.3.1 UART Mode Registers 1 (UMR1*n*)

The UART mode registers 1 (UMR1*n*) control configuration. UMR1*n* can be read or written when the mode register pointer points to it, at RESET or after a RESET MODE REGISTER POINTER command using UCR*n*[MISC]. After UMR1*n* is read or written, the pointer points to UMR2*n*.

	7	6	5	4	3	2	1	0
Field	RxRTS	RxIRQ/FFULL	ERR	PM		PT	B/C	
Reset	0000_0000							
R/W	R/W							
Address	MBAR + 0x1C0 (UART0), 0x200 (UART1). After UMR1n is read or written, the pointer points to UMR2n.							

Figure 14-2. UART Mode Registers 1 (UMR1*n*)

Table 14-2 describes UMR1*n* fields.

Table 14-2. UMR1*n* Field Descriptions

Bits	Name	Description
7	RxRTS	Receiver request-to-send. Allows the $\overline{\text{RTS}}$ output to control the $\overline{\text{CTS}}$ input of the transmitting device to prevent receiver overrun. If both the receiver and transmitter are incorrectly programmed for $\overline{\text{RTS}}$ control, $\overline{\text{RTS}}$ control is disabled for both. Transmitter RTS control is configured in UMR2n[TxRTS]. 0 The receiver has no effect on $\overline{\text{RTS}}$. 1 When a valid start bit is received, $\overline{\text{RTS}}$ is negated if the UART's FIFO is full. $\overline{\text{RTS}}$ is reasserted when the FIFO has an empty position available.
6	RxIRQ/ FFULL	Receiver interrupt select. 0 RxRDY is the source that generates IRQ. 1 FFULL is the source that generates IRQ.
5	ERR	Error mode. Configures the FIFO status bits, USRn[RB,FE,PE]. 0 Character mode. The USRn 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 USRn values are the logical OR of the status for all characters reaching the top of the FIFO because the last RESET ERROR STATUS command for the channel was issued. See Section 14.3.5, “UART Command Registers (UCRn).”
4–3	PM	Parity mode. Selects the parity or multidrop mode for the channel. The parity bit is added to the transmitted character, and the receiver performs a parity check on incoming data. The value of PM affects PT, as shown below.
2	PT	Parity type. PM and PT together select parity type (PM = 0x) or determine whether a data or address character is transmitted (PM = 11).
1–0	B/C	Bits per character. Select the number of data bits per character to be sent. The values shown do not include start, parity, or stop bits. 00 5 bits 01 6 bits 10 7 bits 11 8 bits

PM	Parity Mode	Parity Type (PT= 0)	Parity Type (PT= 1)
00	With parity	Even parity	Odd parity
01	Force parity	Low parity	High parity
10	No parity	n/a	
11	Multidrop mode	Data character	Address character

14.3.2 UART Mode Register 2 (UMR2*n*)

UART mode registers 2 (UMR2*n*) 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.

	7	6	5	4	3	0
Field	CM		TxRTS	TxCTS	SB	
Reset	0000_0000					
R/W	R/W					
Address	MBAR + 0x1C0, 0x200. After UMR1n is read or written, the pointer points to UMR2n.					

Figure 14-3. UART Mode Register 2 (UMR2*n*)

Table 14-3 describes UMR2*n* fields.

Table 14-3. UMR2*n* Field Descriptions

Bits	Name	Description
7–6	CM	Channel mode. Selects a channel mode. Section 14.5.3, “Looping Modes,” describes individual modes. 00 Normal 01 Automatic echo 10 Local loop-back 11 Remote loop-back
5	TxRTS	Transmitter ready-to-send. Controls negation of $\overline{\text{RTS}}$ to automatically terminate a message transmission. Attempting to program a receiver and transmitter in the same channel for $\overline{\text{RTS}}$ control is not permitted and disables $\overline{\text{RTS}}$ control for both. 0 The transmitter has no effect on $\overline{\text{RTS}}$. 1 In applications where the transmitter is disabled after transmission completes, setting this bit automatically clears UOP[RTS] one bit time after any characters in the channel transmitter shift and holding registers are completely sent, including the programmed number of stop bits.
4	TxCTS	Transmitter clear-to-send. If both TxCTS and TxRTS are enabled, TxCTS controls the operation of the transmitter. 0 $\overline{\text{CTS}}$ has no effect on the transmitter. 1 Enables clear-to-send operation. The transmitter checks the state of $\overline{\text{CTS}}$ each time it is ready to send a character. If $\overline{\text{CTS}}$ is asserted, the character is sent; if it is negated, the channel TxD remains in the high state and transmission is delayed until $\overline{\text{CTS}}$ is asserted. Changes in $\overline{\text{CTS}}$ as a character is being sent do not affect its transmission.

Table 14-3. UMR2*n* Field Descriptions (Continued)

Bits	Name	Description
3–0	SB	Stop-bit length control. Selects the length of the stop bit appended to the transmitted character. Stop-bit lengths of 9/16th to 2 bits are programmable for 6–8 bit characters. Lengths of 1 1/16th to 2 bits are programmable for 5-bit characters. In all cases, the receiver checks only for a high condition at the center of the first stop-bit position, that is, one bit time after the last data bit or after the parity bit, if parity is enabled. If an external 1x clock is used for the transmitter, clearing bit 3 selects one stop bit and setting bit 3 selects 2 stop bits for transmission.

SB	5 Bits	6–8 Bits	SB	5 Bits	6–8 Bits	SB	5–8 Bits	SB	5–8 Bits
0000	1.063	0.563	0100	1.313	0.813	1000	1.563	1100	1.813
0001	1.125	0.625	0101	1.375	0.875	1001	1.625	1101	1.875
0010	1.188	0.688	0110	1.438	0.938	1010	1.688	1110	1.938
0011	1.250	0.750	0111	1.500	1.000	1011	1.750	1111	2.000

14.3.3 UART Status Registers (USR*n*)

The USR*n*, Figure 14-4, shows status of the transmitter, the receiver, and the FIFO.

	7	6	5	4	3	2	1	0
Field	RB	FE	PE	OE	TxEMP	TxRDY	FFULL	RxRDY
Reset	0000_0000							
R/W	Read only							
Address	MBAR + 0x1C4 (USR0), 0x204 (USR1)							

Figure 14-4. UART Status Register (USR*n*)

Table 14-4 describes USR*n* fields.

Table 14-4. USR*n* Field Descriptions

Bits	Name	Description
7	RB	Received break. The received break circuit detects breaks that originate in the middle of a received character. However, a break in the middle of a character must persist until the end of the next detected character time. 0 No break was received. 1 An all-zero character of the programmed length was received without a stop bit. RB is valid only when RxRDY = 1. Only a single FIFO position is occupied when a break is received. Further entries to the FIFO are inhibited until RxD returns to the high state for at least one-half bit time, which is equal to two successive edges of the UART clock.
6	FE	Framing error. 0 No framing error occurred. 1 No stop bit was detected when the corresponding data character in the FIFO was received. The stop-bit check occurs in the middle of the first stop-bit position. FE is valid only when RxRDY = 1.

Table 14-4. USR*n* Field Descriptions (Continued)

Bits	Name	Description
5	PE	Parity error. Valid only if RxRDY = 1. 0 No parity error occurred. 1 If UMR1n[PM] = 0x (with parity or force parity), the corresponding character in the FIFO was received with incorrect parity. If UMR1n[PM] = 11 (multidrop), PE stores the received A/D bit.
4	OE	Overrun error. Indicates whether an overrun occurs. 0 No overrun occurred. 1 One or more characters in the received data stream have been lost. OE is set upon receipt of a new character when the FIFO is full and a character is already in the shift register waiting for an empty FIFO position. When this occurs, the character in the receiver shift register and its break detect, framing error status, and parity error, if any, are lost. OE is cleared by the RESET ERROR STATUS command in UCRn.
3	TxEMP	Transmitter empty. 0 The transmitter buffer is not empty. Either a character is being shifted out, or the transmitter is disabled. The transmitter is enabled/disabled by programming UCRn[TC]. 1 The transmitter has underrun (both the transmitter holding register and transmitter shift registers are empty). This bit is set after transmission of the last stop bit of a character if there are no characters in the transmitter holding register awaiting transmission.
2	TxRDY	Transmitter ready. 0 The CPU loaded the transmitter holding register or the transmitter is disabled. 1 The transmitter holding register is empty and ready for a character. TxRDY is set when a character is sent to the transmitter shift register and when the transmitter is first enabled. If the transmitter is disabled, characters loaded into the transmitter holding register are not sent.
1	FFULL	FIFO full. 0 The FIFO is not full but may hold up to two unread characters. 1 A character was received and is waiting in the receiver buffer FIFO.
0	RxRDY	Receiver ready 0 The CPU has read the receiver buffer and no characters remain in the FIFO after this read. 1 One or more characters were received and are waiting in the receiver buffer FIFO.

14.3.4 UART Clock-Select Registers (UCSR*n*)

The UART clock-select registers (UCSR*n*) select an external clock on the TIN input (divided by 1 or 16) or a prescaled BCLKO as the clocking source for the transmitter and receiver. See Section 14.5.1, “Transmitter/Receiver Clock Source.” The transmitter and receiver can use different clock sources. To use BCLKO for both, set UCSR*n* to 0xDD.

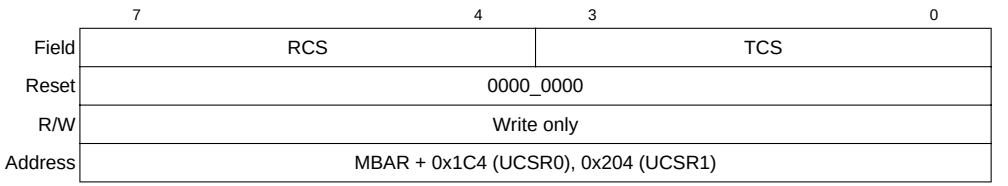


Figure 14-5. UART Clock-Select Register (UCSR*n*)

Table 14-5 describes UCSR*n* fields.

Table 14-5. UCSR*n* Field Descriptions

Bits	Name	Description
7–4	RCS	Receiver clock select. Selects the clock source for the receiver channel. 1101 Prescaled BCLKO 1110 TIN divided by 16 1111 TIN
3–0	TCS	Transmitter clock select. Selects the clock source for the transmitter channel. 1101 Prescaled BCLKO 1110 TIN divided by 16 1111 TIN

14.3.5 UART Command Registers (UCR*n*)

The UART command registers (UCR*n*), Figure 14-6, 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.

	7	6	4	3	2	1	0
Field	—	MISC		TC		RC	
Reset	0000_0000						
R/W	Write only						
Address	MBAR + 0x1C8, 0x208						

Figure 14-6. UART Command Register (UCR*n*)

Table 14-6 describes UCR*n* fields and commands. Examples in Section 14.5.2, “Transmitter and Receiver Operating Modes,” show how these commands are used.

Table 14-6. UCR*n* Field Descriptions

Bits	Value	Command	Description
7	—	—	Reserved, should be cleared.

Table 14-6. UCR*n* Field Descriptions (Continued)

Bits	Value	Command	Description
6–4	**MISC Field** (This field selects a single command.)		
	000	NO COMMAND	—
	001	RESET MODE REGISTER POINTER	Causes the mode register pointer to point to UMR1n.
	010	RESET RECEIVER	Immediately disables the receiver, clears USRn[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	disables the transmitter and clears USRn[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	lears USRn[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, UISRn[DB].
	110	START BREAK	Forces TxD low. If the transmitter is empty, the break may be delayed up to one bit time. If the transmitter is active, the break starts when character transmission completes. The break is delayed until any character in the transmitter shift register is sent. Any character in the transmitter holding register is sent after the break. The transmitter must be enabled for the command to be accepted. This command ignores the state of $\overline{\text{CTS}}$.
	111	STOP BREAK	Causes TxD to go high (mark) within two bit times. Any characters in the transmitter buffer are sent.
3–2	**TC Field** (This field selects a single command)		
	00	NO ACTION TAKEN	Causes the transmitter to stay in its current mode: if the transmitter is enabled, it remains enabled; if the transmitter is disabled, it remains disabled.
	01	TRANSMITTER ENABLE	Enables operation of the channel's transmitter. USRn[TxEMP,TxRDY] are set. If the transmitter is already enabled, this command has no effect.
	10	TRANSMITTER DISABLE	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.
	11	—	Reserved, do not use.

Table 14-6. UCR*n* Field Descriptions (Continued)

Bits	Value	Command	Description
1–0	**RC** (This field selects a single command)		
	00	NO ACTION TAKEN	Causes the receiver to stay in its current mode. If the receiver is enabled, it remains enabled; if disabled, it remains disabled.
	01	RECEIVER ENABLE	If the UART module is not in multidrop mode (UMR1n[PM] ≠ 11), RECEIVER ENABLE enables the channel's receiver and forces it into search-for-start-bit state. If the receiver is already enabled, this command has no effect.
	10	RECEIVER DISABLE	Disables the receiver immediately. Any character being received is lost. The command does not affect receiver status bits or other control registers. If the UART module is programmed for local loop-back or multidrop mode, the receiver operates even though this command is selected. If the receiver is already disabled, the command has no effect.
	11	—	Reserved, do not use.

14.3.6 UART Receiver Buffers (URB*n*)

The receiver buffers contain one serial shift register and three receiver holding registers, which act as a FIFO. RxD is connected to the serial shift register. The CPU reads from the top of the stack while the receiver shifts and updates from the bottom when the shift register is full (see Figure 14-20). RB contains the character in the receiver.

	7	0
Field	RB	
Reset	0000_0000	
R/W	Read only	
Address	MBAR + 0x1CC,0x20C	

Figure 14-7. UART Receiver Buffer (URB0)

14.3.7 UART Transmitter Buffers (UTB*n*)

The transmitter buffers consist of the transmitter holding register and the transmitter shift register. The holding register accepts characters from the bus master if channel's USR*n*[TxRDY] is set. A write to the transmitter buffer clears 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 transmitter buffer when the channel's TxRDY = 0 and when the transmitter is disabled have no effect on the transmitter buffer.

Figure 14-8 shows UTB0. TB contains the character in the transmitter buffer.

	7	0
Field	TB	
Reset	0000_0000	
R/W	Write only	
Address	MBAR + 0x1CC,0x20C	

Figure 14-8. UART Transmitter Buffer (UTB0)

14.3.8 UART Input Port Change Registers (UIPCR*n*)

The input port change registers (UIPCR*n*), Figure 14-9, hold the current state and the change-of-state for $\overline{\text{CTS}}$.

	7	5	4	3	1	0
Field	—		COS	111		CTS
Reset	0000			0	11	$\overline{\text{CTS}}$
R/W	Read only					
Address	MBAR + 0x1D0 (UIPCR0), 0x210 (UIPCR1)					

Figure 14-9. UART Input Port Change Register (UIPCR*n*)

Table 14-7 describes UIPCR*n* fields.

Table 14-7. UIPCR*n* Field Descriptions

Bits	Name	Description
7–5	—	Reserved, should be cleared.
4	COS	Change of state (high-to-low or low-to-high transition). 0 No change-of-state since the CPU last read UIPCRn. Reading UIPCRn clears UISRn[COS]. 1 A change-of-state longer than 25–50 μs occurred on the $\overline{\text{CTS}}$ input. UACRn can be programmed to generate an interrupt to the CPU when a change of state is detected.
3–1	—	Reserved, should be cleared.
0	CTS	Current state. Starting two serial clock periods after reset, CTS reflects the state of $\overline{\text{CTS}}$. If $\overline{\text{CTS}}$ is detected asserted at that time, COS is set, which initiates an interrupt if UACRn[IEC] is enabled. 0 The current state of the $\overline{\text{CTS}}$ input is asserted. 1 The current state of the $\overline{\text{CTS}}$ input is negated.

14.3.9 UART Auxiliary Control Register (UACR*n*)

The UART auxiliary control registers (UACR*n*), Figure 14-7, control the input enable.

	7	1	0
Field	—		IEC
Reset	0000_0000		
R/W	Write only		
Address	MBAR + 0x1D0 (UACR0), 0x210 (UACR1)		

Figure 14-10. UART Auxiliary Control Register (UACR*n*)

Table 14-8 describes UACR*n* fields.

Table 14-8. UACR*n* Field Descriptions

Bits	Name	Description
7–1	—	Reserved, should be cleared.
0	IEC	Input enable control. 0 Setting the corresponding UIPCRn bit has no effect on UISRn[COS]. 1 UISRn[COS] is set and an interrupt is generated when the UIPCRn[COS] is set by an external transition on the $\overline{\text{CTS}}$ input (if UIMRn[COS] = 1).

14.3.10 UART Interrupt Status/Mask Registers (UISR*n*/UIMR*n*)

The UART interrupt status registers (UISR*n*), Figure 14-11, provide status for all potential interrupt sources. UISR*n* contents are masked by UIMR*n*. If corresponding UISR*n* and UIMR*n* bits are set, the internal interrupt output is asserted. If a UIMR*n* bit is cleared, the state of the corresponding UISR*n* bit has no effect on the output.

NOTE:

True status is provided in the UISR*n* regardless of UIMR*n* settings. UISR*n* is cleared when the UART module is reset.

	7	6	3	2	1	0
Field	COS	—		DB	FFULL/RxRDY	TxRDY
Reset	0000_0000					
R/W	Read only for status, write only for mask.					
Address	MBAR + 0x1D4 (UISR0), 0x214 (UISR1); MBAR + 0x1D4 (UIMR0), 0x214 (UIMR1)					

Figure 14-11. UART Interrupt Status/Mask Registers (UISR*n*/UIMR*n*)

Table 14-9 describes UISR*n* and UIMR*n* fields.

Table 14-9. UISR*n*/UIMR*n* Field Descriptions

Bits	Name	Description
7	COS	Change-of-state. 0 UIPCRn[COS] is not selected. 1 Change-of-state occurred on $\overline{\text{CTS}}$ and was programmed in UACRn[IEC] to cause an interrupt.
6–3	—	Reserved, should be cleared.
2	DB	Delta break. 0 No new break-change condition to report. Section 14.3.5, “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	RxRDY (receiver ready) if UMR1n[FFULL/RxRDY] = 0; FIFO full (FFULL) if UMR1n[FFULL/RxRDY] = 1. Duplicate of USRn[FFULL/RxRDY]. If FFULL is enabled for UART0 or UART1, DMA channels 2 or 3 are respectively interrupted when the FIFO is full.
0	TxRDY	Transmitter ready. This bit is the duplication of USRn[TxRDY]. 0 The transmitter holding register was loaded by the CPU or the transmitter is disabled. Characters loaded into the transmitter holding register when TxRDY = 0 are not sent. 1 The transmitter holding register is empty and ready to be loaded with a character.

14.3.11 UART Divider Upper/Lower Registers (UDU*n*/UDL*n*)

The UDU*n* registers (formerly called UBG1*n*) holds the MSB, and the UDL*n* registers (formerly UBG2*n*) hold the LSB of the preload value. UDU*n* and UDL*n* concatenate to provide a divider to BCLKO for transmitter/receiver operation, as described in Section 14.5.1.2.1, “BCLKO Baud Rates.”

	7	0
Field	Divider MSB	
Reset	0000_0000	
R/W	R/W	
Address	MBAR + 0x1D8 (UDU0), 0x218 (UDU1)	

Figure 14-12. UART Divider Upper Register (UDU*n*)

	7	0
Field	Divider LSB	
Reset	0000_0000	
R/W	R/W	
Address	MBAR + 0x1DC (UDL0), 0x21C (UDL1)	

Figure 14-13. UART Divider Lower Register (UDL*n*)

NOTE:

The minimum value that can be loaded on the concatenation of UDU*n* with UDL*n* is 0x0002. Both UDU*n* and UDL*n* are write-only and cannot be read by the CPU.

14.3.12 UART Interrupt Vector Register (UIVR*n*)

The UIVR*n*, Figure 14-14, contain the 8-bit internal interrupt vector number (IVR).

	7	0
Field	IVR	
Reset	0000_1111	
R/W	R/W	
Address	MBAR + 0x1F0 (UIVR0), 0x230 (UIVR1)	

Figure 14-14. UART Interrupt Vector Register (UIVR*n*)

Table 14-10 describes UIVR*n* fields.

Table 14-10. UIVR*n* Field Descriptions

Bits	Name	Description
7–0	IVR	Interrupt vector. Indicates the vector number where the address of the exception handler for the specified interrupt is located. UIVRn is reset to 0x0F, indicating an uninitialized interrupt condition.

14.3.13 UART Input Port Register (UIP*n*)

The UART input port registers (UIP*n*), Figure 14-15, show the current state of the $\overline{\text{CTS}}$ input.

	7	1	0
Field	—		CTS
Reset	1111_1111		
R/W	Read only		
Address	MBAR + 0x1F4 (UIP0), 0x234 (UIP1)		

Figure 14-15. UART Input Port Register (UIP*n*)

Table 14-11 describes UIP*n* fields.

Table 14-11. UIP*n* Field Descriptions

Bits	Name	Description
7–1	—	Reserved, should be cleared.
0	CTS	Current state. The $\overline{\text{CTS}}$ value is latched and reflects the state of the input pin when UIPn is read. Note: This bit has the same function and value as UIPCRn[RTS]. 0 The current state of the $\overline{\text{CTS}}$ input is logic 0. 1 The current state of the $\overline{\text{CTS}}$ input is logic 1.

14.3.14 UART Output Port Command Registers (UOP1*n*/UOP0*n*)

In UART mode, the $\overline{\text{RTS}}$ output can be asserted by writing a 1 to UOP1*n*[RTS] and negated by writing a 1 to UOP0*n*[RTS]. See Figure 14-16.

	7	1	0
Field	—		RTS
Reset	0000_0000		
R/W	Write only		
Addr	UART0: MBAR + 0x1F8 (UOP1), 0x1FC (UOP0); UART1 0x238 (UOP1), 0x23C (UOP0)		

Figure 14-16. UART Output Port Command Register (UOP1/UOP0)

Table 14-12 describes UOP1 fields.

Table 14-12. UOP1/UOP0 Field Descriptions

Bits	Name	Description
7–1	—	Reserved, should be cleared.
0	RTS	Output port parallel output. Controls assertion (UOP1)/negation (UOP0) of $\overline{RTS}$ output. 0 Not affected. 1 Asserts $\overline{RTS}$ (UOP1). Negates $\overline{RTS}$ (UOP0).

14.4 UART Module Signal Definitions

Figure 14-17 shows both the external and internal signal groups.

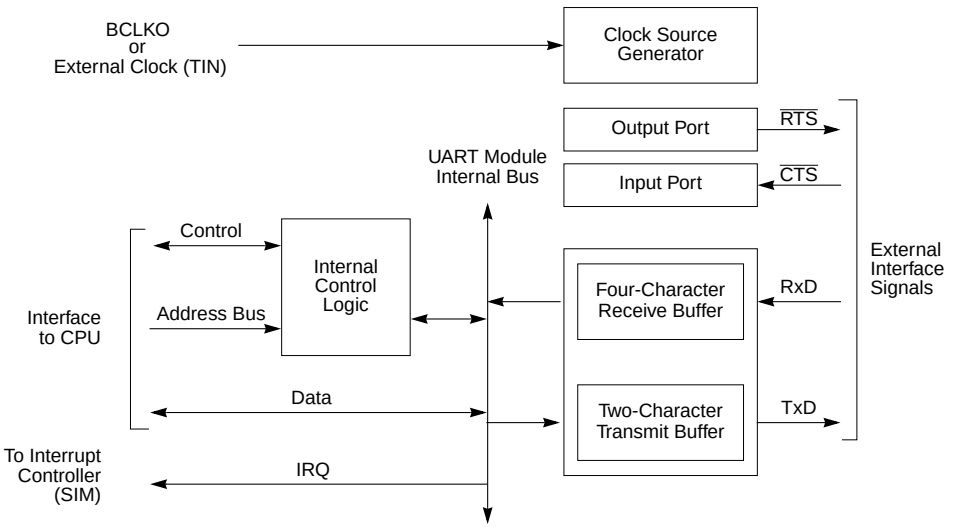


Figure 14-17. UART Block Diagram Showing External and Internal Interface Signals

An internal interrupt request signal ($\overline{IRQ}$) is provided to notify the interrupt controller of an interrupt condition. The output is the logical NOR of unmasked UISR*n* bits. The interrupt level of a UART module is programmed in the interrupt controller in the system integration module (SIM). The UART can use the autovector for the programmed interrupt level or supply the vector from the UIVR*n* when the UART interrupt is acknowledged.

The interrupt level, priority, and auto-vectoring capability is programmed in SIM register ICR4 for UART0 and ICR5 for UART1. See Section 9.2.1, “Interrupt Control Registers (ICR0–ICR9).”

Note that the UARTs can also automatically transfer data by using the DMA rather than interrupting the core. When UIMR[FFULL] is 1 and a receiver’s FIFO is full, it can send an interrupt to a DMA channel so the FIFO data can be transferred to memory. Note also that UART0 and UART1’s interrupt requests are connected to DMA channel 2 and channel 3, respectively.

Table 14-13 briefly describes the UART module signals.

NOTE:

The terms ‘assertion’ and ‘negation’ are used to avoid confusion between active-low and active-high signals. ‘Asserted’ indicates that a signal is active, independent of the voltage level; ‘negated’ indicates that a signal is inactive.

Table 14-13. UART Module Signals

Signal	Description
Transmitter Serial Data Output (TxD)	TxD is held high (mark condition) when the transmitter is disabled, idle, or operating in the local loop-back mode. Data is shifted out on TxD on the falling edge of the clock source, with the least significant bit (lsb) sent first.
Receiver Serial Data Input (RxD)	Data received on RxD is sampled on the rising edge of the clock source, with the lsb received first.
Clear-to-Send ($\overline{\text{CTS}}$)	This input can generate an interrupt on a change of state.
Request-to-Send ($\overline{\text{RTS}}$)	This output can be programmed to be negated or asserted automatically by either the receiver or the transmitter. When connected to a transmitter’s $\overline{\text{CTS}}$, $\overline{\text{RTS}}$ can control serial data flow.

Figure 14-18 shows a signal configuration for a UART/RS-232 interface.

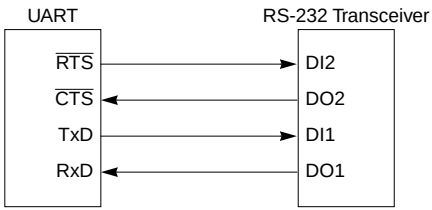


Figure 14-18. UART/RS-232 Interface

14.5 Operation

This section describes operation of the clock source generator, transmitter, and receiver.

14.5.1 Transmitter/Receiver Clock Source

BCLKO 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 the UART. The clock generator cannot produce standard baud rates if BCLKO is used, so the 16-bit divider should be used.

14.5.1.1 Programmable Divider

As Figure 14-19 shows, the UART transmitter and receiver can use the following clock sources:

- An external clock signal on the TIN pin that can be divided by 16. When not divided, TIN provides a synchronous clock mode; when divided by 16, it is asynchronous**.**
- BCLKO supplies an asynchronous clock source that is divided by 32 and then divided by the 16-bit value programmed in UDU*n* and UDL*n*. See Section 14.3.11, “UART Divider Upper/Lower Registers (UDUn/UDLn).”

The choice of TIN or BCLKO is programmed in the UCSR.

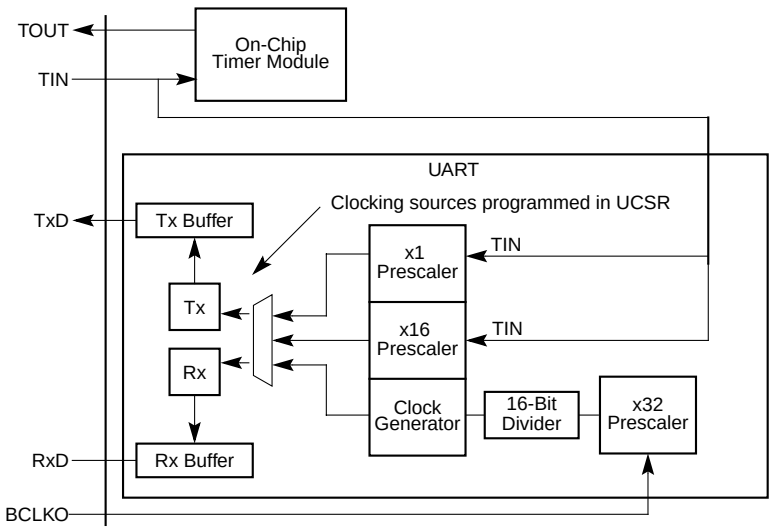


Figure 14-19. Clocking Source Diagram

NOTE:

If TIN is a clocking source for either the timer or UART, the timer module cannot use TIN for timer capture.

14.5.1.2 Calculating Baud Rates

The following sections describe how to calculate baud rates.

14.5.1.2.1 BCLKO Baud Rates

When BCLKO is the UART clocking source, it goes through a divide-by-32 prescaler and then passes through the 16-bit divider of the concatenated UDU*n* and UDL*n* registers. Using a 45-MHz BCLKO, the baud-rate calculation is as follows:

$$\text{Baudrate} = \frac{45\text{MHz}}{[32 \times \text{divider}]}$$

let baud rate = 9600, then

$$\text{Divider} = \frac{45\text{MHz}}{[32 \times 9600]} = 146(\text{decimal}) = 0092(\text{hexadecimal})$$

therefore UDU = 0x00 and UDL = 0x92.

14.5.1.2.2 External Clock

An external source clock (TIN) can be used as is or divided by 16.

$$\text{Baudrate} = \frac{\text{Externalclockfrequency}}{16\text{or}1}$$

14.5.2 Transmitter and Receiver Operating Modes

Figure 14-20 is a functional block diagram of the transmitter and receiver showing the command and operating registers, which are described generally in the following sections and described in detail in Section 14.3, “Register Descriptions.”

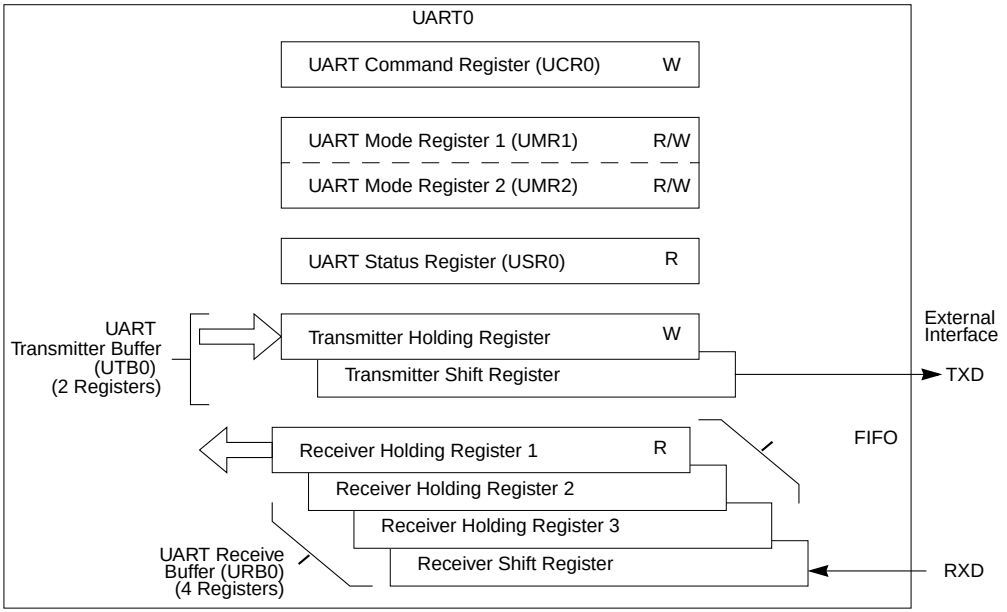


Figure 14-20. Transmitter and Receiver Functional Diagram

14.5.2.1 Transmitting

The transmitter is enabled through the UART command register (UCR*n*). When it is ready to accept a character, the UART sets USR*n*[TxRDY]. The transmitter converts parallel data from the CPU to a serial bit stream on TxD. It automatically sends a start bit followed by the programmed number of data bits, an optional parity bit, and the programmed number of stop bits. The lsb is sent first. Data is shifted from the transmitter output on the falling edge of the clock source.

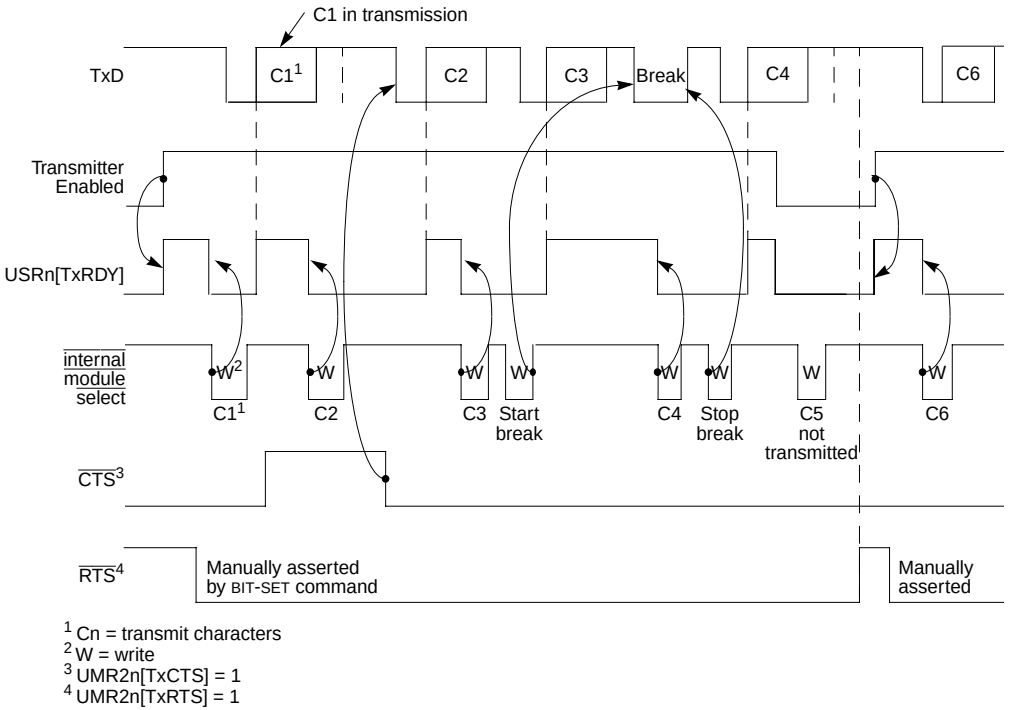
After the stop bits are sent, if no new character is in the transmitter holding register, the TxD output remains high (mark condition) and the transmitter empty bit, USR*n*[TxEMP], is set. Transmission resumes and TxEMP is cleared when the CPU loads a new character into the UART transmitter buffer (UTB*n*). 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 14.3.5, “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, $\overline{\text{CTS}}$ must be asserted for the character to be transmitted. If $\overline{\text{CTS}}$ is negated in the middle of a transmission, the character in the shift register is sent and TxD remains in mark state until $\overline{\text{CTS}}$ is reasserted. If the transmitter is forced to send a continuous low condition by issuing a SEND BREAK command, the transmitter ignores the state of $\overline{\text{CTS}}$.

If the transmitter is programmed to automatically negate $\overline{\text{RTS}}$ when a message transmission completes, $\overline{\text{RTS}}$ must be asserted manually before a message is sent. In applications in which the transmitter is disabled after transmission is complete and $\overline{\text{RTS}}$ is appropriately programmed, $\overline{\text{RTS}}$ is negated one bit time after the character in the shift register is completely transmitted. The transmitter must be manually reenabled by reasserting $\overline{\text{RTS}}$ before the next message is to be sent.

Figure 14-21 shows the functional timing information for the transmitter.



[1] Cn = transmit characters
[2] W = write
[3] UMR2n[TxCTS] = 1
[4] UMR2n[TxRTS] = 1

Figure 14-21. Transmitter Timing Diagram

14.5.2.2 Receiver

The receiver is enabled through its UCR*n*, as described in Section 14.3.5, "UART Command Registers (UCRn)." Figure 14-22 shows receiver functional timing.

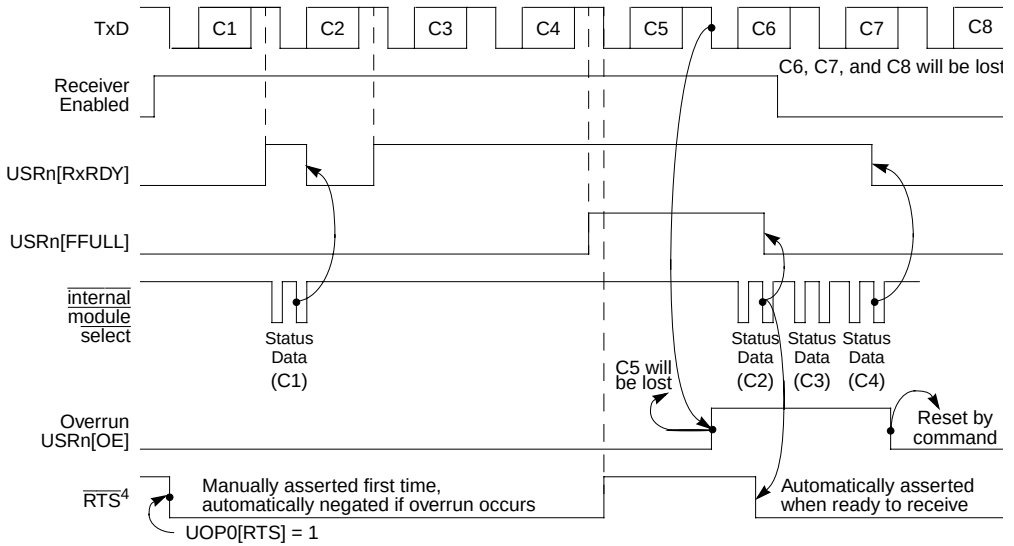


Figure 14-22. Receiver Timing

When the receiver detects a high-to-low (mark-to-space) transition of the start bit on RxD, the state of RxD is sampled each 16× clock for eight clocks, starting one-half clock after the transition (asynchronous operation) or at the next rising edge of the bit time clock (synchronous operation). If RxD is sampled high, the start bit is invalid and the search for the valid start bit begins again.

If RxD is still low, a valid start bit is assumed and the receiver continues sampling the input at one-bit time intervals, at the theoretical center of the bit, until the proper number of data bits and parity, if any, is assembled and one stop bit is detected. Data on the RxD input is sampled on the rising edge of the programmed clock source. The lsb is received first. The data is then transferred to a receiver holding register and USR*n*[RxRDY] is set. If the character is less than eight bits, the most significant unused bits in the receiver holding register are cleared.

After the stop bit is detected, the receiver immediately looks for the next start bit. However, if a non-zero character is received without a stop bit (framing error) and RxD remains low for one-half of the bit period after the stop bit is sampled, the receiver operates as if a new start bit were detected. Parity error, framing error, overrun error, and received break conditions set the respective PE, FE, OE, RB error and break flags in the USR*n* at the received character boundary and are valid only if USR*n*[RxRDY] is set.

If a break condition is detected (RxD is low for the entire character including the stop bit), a character of all zeros is loaded into the receiver holding register (RHR) and USR*n*[RB,RxRDY] are set. RxD must return to a high condition for at least one-half bit time before a search for the next start bit begins.

The receiver detects the beginning of a break in the middle of a character if the break persists through the next character time. If the break begins in the middle of a character, the receiver places the damaged character in the Rx FIFO stack 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].

14.5.2.3 FIFO Stack

The FIFO stack is used in the UART's receiver buffer logic. The stack consists of three receiver holding registers. The receive buffer consists of the FIFO and a receiver shift register connected to the RxD (see Figure 14-20). Data is assembled in the receiver shift register and loaded into the top empty receiver holding register position of the FIFO. Thus, data flowing from the receiver to the CPU is quadruple-buffered.

In addition to the data byte, three status bits, parity error (PE), framing error (FE), and received break (RB), are appended to each data character in the FIFO; OE (overrun error) is not appended. By programming the ERR bit in the channel's mode register (UMR1*n*), status is provided in character or block modes.

USR*n*[RxRDY] is set when at least one character is available to be read by the CPU. A read of the receiver buffer produces an output of data from the top of the FIFO stack. After the read cycle, the data at the top of the FIFO stack and its associated status bits are popped and the receiver shift register can add new data at the bottom of the stack. The FIFO-full status bit (FFULL) is set if all three stack positions are filled with data. Either the RxRDY or FFULL bit can be selected to cause an interrupt.

The two error modes are selected by UMR1*n*[ERR] as follows:

- In character mode (UMR1*n*[ERR] = 0, status is given in the USR*n* for the character at the top of the FIFO.
- In block mode, the USR*n* shows a logical OR of all characters reaching the top of the FIFO stack since the last RESET ERROR STATUS command. Status is updated as characters reach the top of the FIFO stack. 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 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 $\overline{\text{RTS}}$, in which case the receiver automatically negates $\overline{\text{RTS}}$ when a valid start bit is detected and the FIFO stack is full. The receiver asserts $\overline{\text{RTS}}$ when a FIFO position becomes

available; therefore, overrun errors can be prevented by connecting $\overline{RTS}$ to the $\overline{CTS}$ input of the transmitting device.

NOTE:

The receiver can still read characters in the FIFO stack if the receiver is disabled. If the receiver is reset, the FIFO stack, $\overline{RTS}$ control, all receiver status bits, and interrupt requests are reset. No more characters are received until the receiver is reenabled.

14.5.3 Looping Modes

The UART can be configured to operate in various looping modes as shown in Figure 14-22 on page 14-23. These modes are useful for local and remote system diagnostic functions. The modes are described in the following paragraphs and in Section 14.3, “Register Descriptions.”

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.

14.5.3.1 Automatic Echo Mode

In automatic echo mode, shown in Figure 14-23, the UART automatically resends received data bit by bit. The local CPU-to-receiver communication continues normally, but the CPU-to-transmitter link is disabled. In this mode, received data is clocked on the receiver clock and resent on TxD. The receiver must be enabled, but the transmitter need not be.

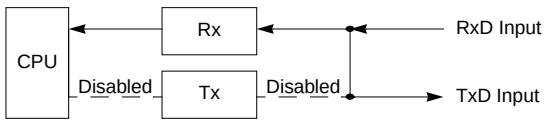


Figure 14-23. Automatic Echo

Because the transmitter is inactive, USR*n*[TxEMP,TxRDY] are inactive and data is sent as it is received. Received parity is checked but is 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.

14.5.3.2 Local Loop-Back Mode

Figure 14-24 shows how TxD and RxD are internally connected in local loop-back mode. This mode is for testing the operation of a local UART module channel by sending data to the transmitter and checking data assembled by the receiver to ensure proper operations.

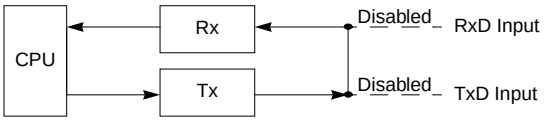


Figure 14-24. Local Loop-Back

Features of this local loop-back mode are as follows:

- Transmitter and CPU-to-receiver communications continue normally in this mode.
- RxD input data is ignored
- TxD is held marking
- The receiver is clocked by the transmitter clock. The transmitter must be enabled, but the receiver need not be.

14.5.3.3 Remote Loop-Back Mode

In remote loop-back mode, shown in Figure 14-25, the channel automatically transmits received data bit by bit on the TxD output. The local CPU-to-transmitter link is disabled. This mode is useful in testing receiver and transmitter operation of a remote channel. For this mode, the transmitter uses the receiver clock.

Because the receiver is not active, received data cannot be read by the CPU and error status conditions are inactive. Received parity is not checked and is not recalculated for transmission. Stop bits are sent as they are received. A received break is echoed as received until the next valid start bit is detected.

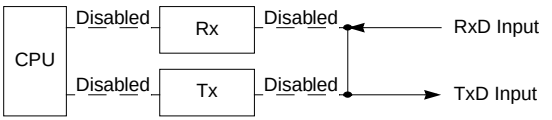


Figure 14-25. Remote Loop-Back

14.5.4 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 channel receivers disabled, they continuously monitor the master's data stream. When the master sends an address character, the slave receiver channel notifies its respective CPU by setting 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. Slave stations not addressed continue monitoring the data stream. Data fields in the data stream are separated by an address character. After

a slave receives a block of data, its CPU disables the receiver and repeats the process.Functional timing information for multidrop mode is shown in Figure 14-26.

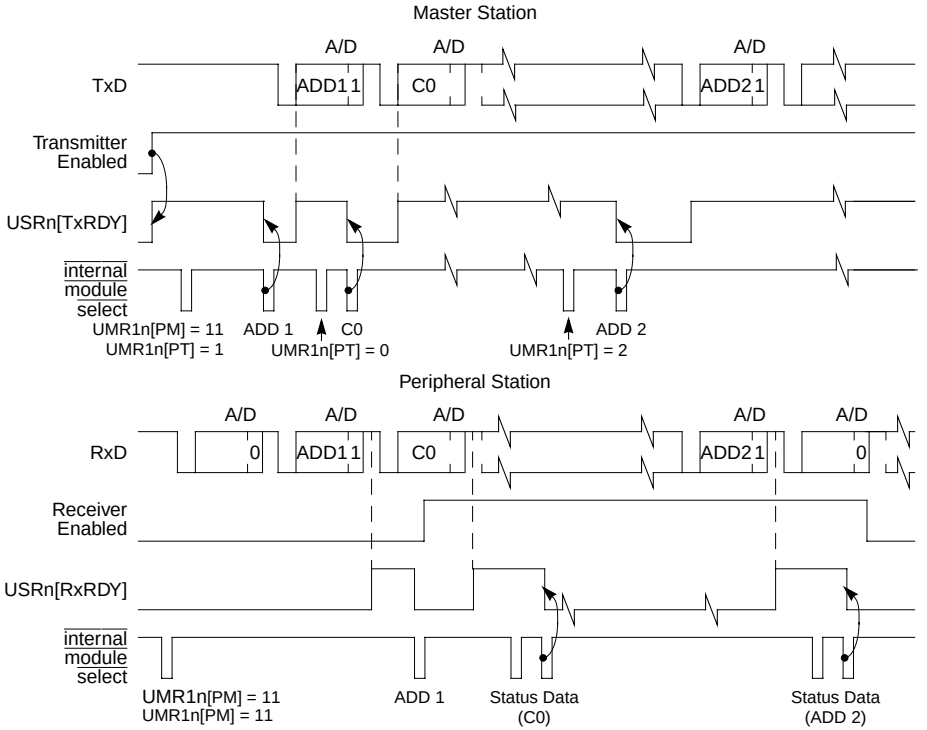


Figure 14-26. Multidrop Mode Timing Diagram

A character sent from the master station consists of a start bit, a programmed number of data bits, an address/data (A/D) bit flag, and a programmed number of stop bits. A/D = 1 indicates an address character; A/D = 0 indicates a data character. The polarity of A/D is selected through 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 stack provided the received A/D bit is a one (address tag). The character is discarded if the received A/D bit is zero (data tag). If the receiver is enabled, all received characters are transferred to the CPU through the receiver holding register stack during read operations.

In either case, the data bits are loaded into the data portion of the stack while the A/D bit is loaded into the status portion of the stack 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 still contain error detection and correction information. One way to provide error detection, if 8-bit characters are not required, is to use software to calculate parity and append it to the 5-, 6-, or 7-bit character.

14.5.5 Bus Operation

This section describes bus operation during read, write, and interrupt acknowledge cycles to the UART module.

14.5.5.1 Read Cycles

The UART module responds to reads with byte data. Reserved registers return zeros.

14.5.5.2 Write Cycles

The UART module accepts write data as bytes. Write cycles to read-only or reserved registers complete normally without exception processing, but data is ignored.

NOTE:

The UART module is accessed by the CPU with zero wait states, as BCLKO is used for the UART module.

14.5.5.3 Interrupt Acknowledge Cycles

The UART module supplies the interrupt vector in response to a UART IACK cycle. If UIVR*n* is not initialized to provide a vector number, a spurious exception is taken if an interrupt is generated. This works in conjunction with the interrupt controller, which allows a programmable priority level.

14.5.6 Programming

The software flowchart, Figure 14-27, consists of the following:

- UART module initialization—These routines consist of SINIT and CHCHK (sheets 1 and 2). Before SINIT is called at system initialization, the calling routine allocates 2 words on the system stack. On return to the calling routine, SINIT passes UART status data on the stack. If SINIT finds no errors, the transmitter and receiver are enabled. SINIT calls CHCHK to perform the checks. When called, SINIT places the UART in local loop-back mode and checks for the following errors:
  - Transmitter never ready
  - Receiver never ready
  - Parity error
  - Incorrect character received
- I/O driver routine—This routine (sheets 4 and 5) consists of INCH, the terminal input character routine which gets a character from the receiver, and OUTCH, which sends a character to the transmitter.

- Interrupt handling—Consists of SIRQ (sheet 4), 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.

14.5.6.1 UART Module Initialization Sequence

NOTE:

UART module registers can be accessed by word or byte operations, but only data byte D[7:0] is valid.

Table 14-14 shows the UART module initialization sequence.

Table 14-14. UART Module Initialization Sequence

Register	Setting
UCRn	Reset the receiver and transmitter. Reset the mode pointer (MISC[2–0] = 0b001).
UIVRn	Program the vector number for a UART module interrupt.
UIMRn	Enable the preferred interrupt sources.
UACRn	Initialize the input enable control (IEC bit).
UCSRn	Select the receiver and transmitter clock. Use timer as source if required.
UMR1n	If preferred, program operation of receiver ready-to-send (RxRTS bit). Select receiver-ready or FIFO-full notification (RxRDY/FFULL bit). Select character or block error mode (ERR bit). Select parity mode and type (PM and PT bits). Select number of bits per character (B/Cx bits).
UMR2n	Select the mode of operation (CMx bits). If preferred, program operation of transmitter ready-to-send (TxRTS). If preferred, program operation of clear-to-send (TxCTS bit). Select stop-bit length (SBx bits).
UCR	

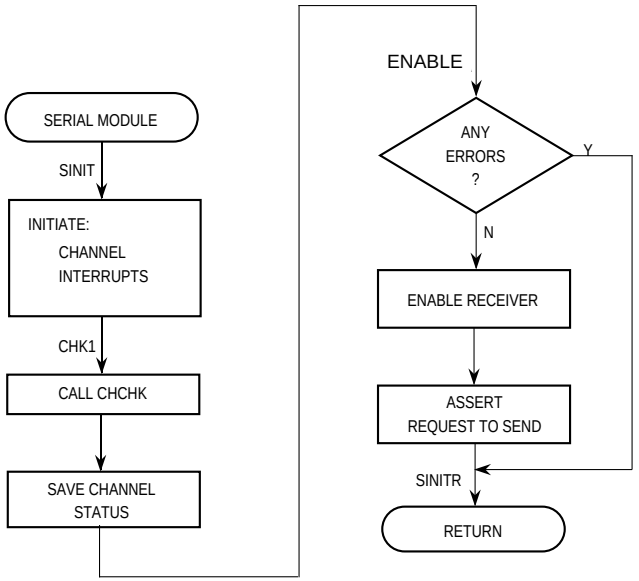


Figure 14-27. UART Mode Programming Flowchart (Sheet 1 of 5)

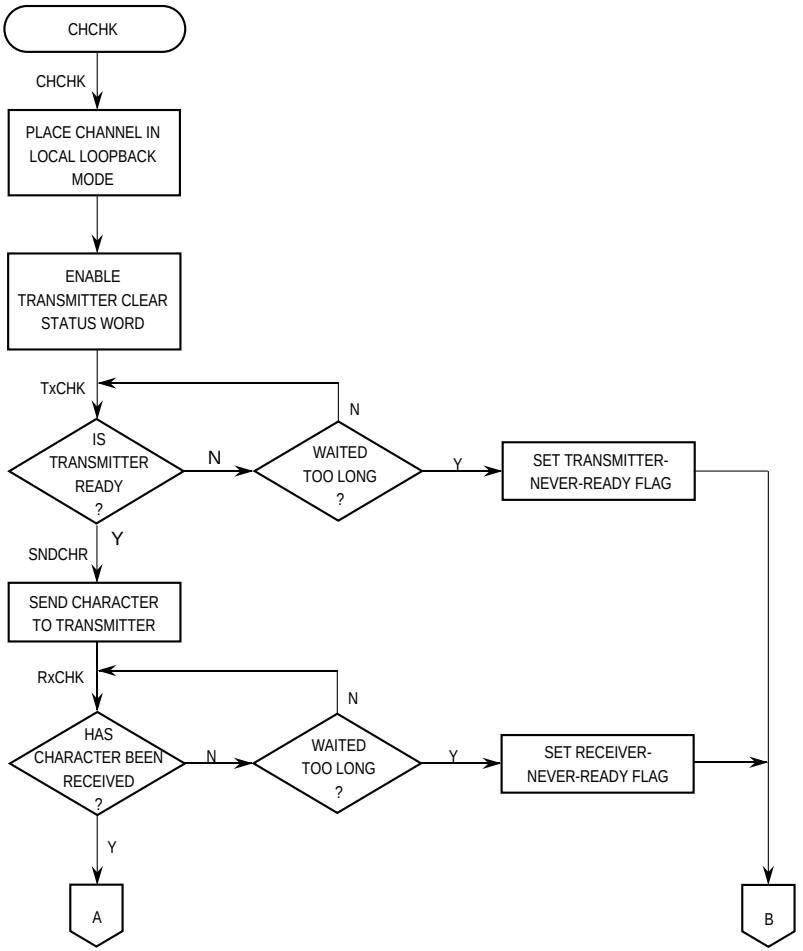


Figure 14-27. UART Mode Programming Flowchart (Sheet 2 of 5)

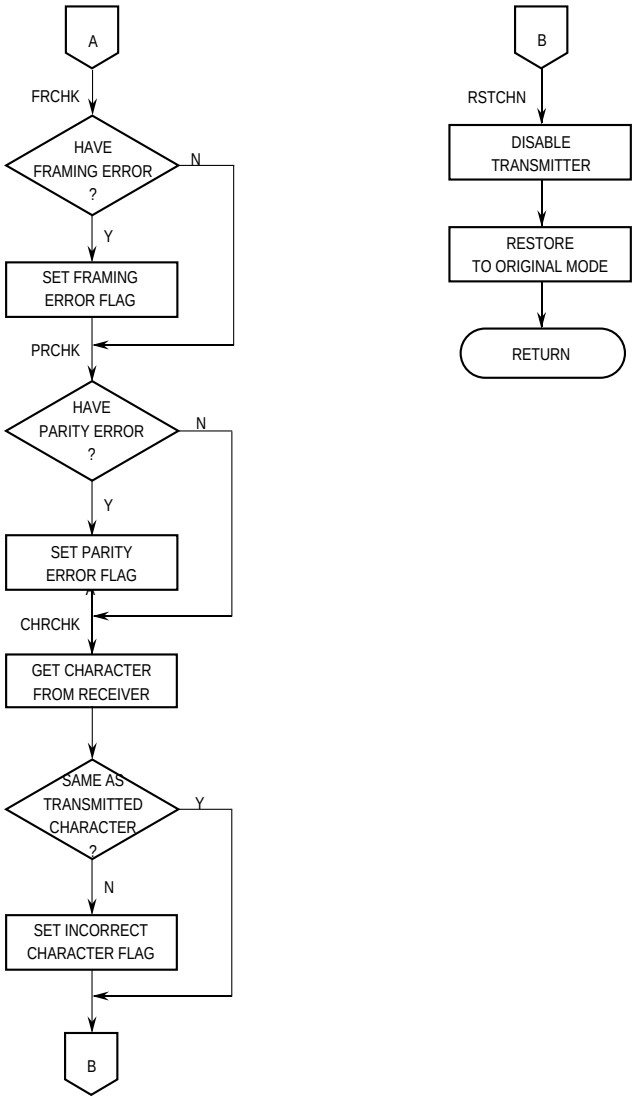


Figure 14-27. UART Mode Programming Flowchart (Sheet 3 of 5)

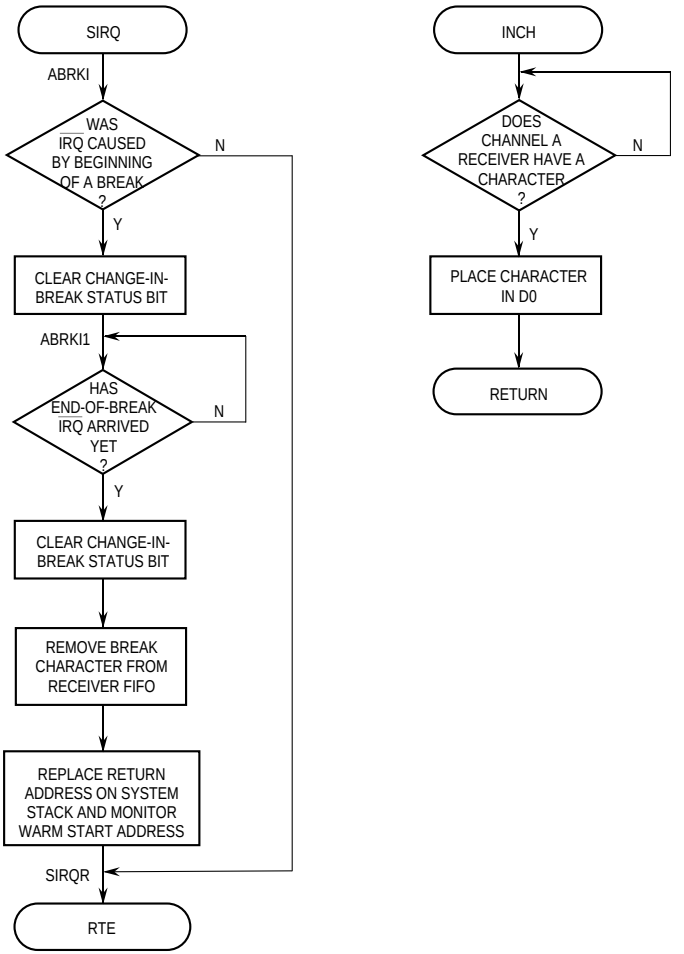


Figure 14-27. UART Mode Programming Flowchart (Sheet 4 of 5)

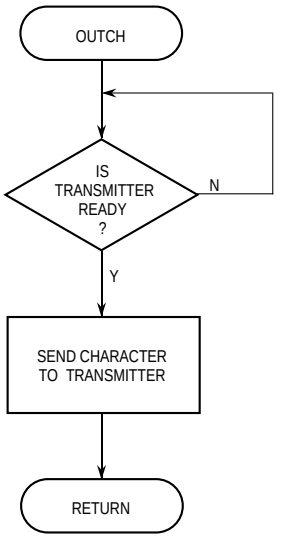


Figure 14-27. UART Mode Programming Flowchart (Sheet 5 of 5)

Chapter 15
Parallel Port (General-Purpose I/O)

This chapter describes the operation and programming model of the parallel port pin assignment, direction-control, and data registers. It includes a code example for setting up the parallel port.

15.1 Parallel Port Operation

The MCF5307 parallel port module has 16 signals, which are programmed as follows:

- The pin assignment register (PAR) selects the function of the 16 multiplexed pins.
- Port A data direction register (PADDR) determines whether pins configured as parallel port signals are inputs or outputs.
- The Port A data register (PADAT) shows the status of the parallel port signals.

The operations of the PAR, PADDR, and PADAT are described in the following sections.

15.1.1 Pin Assignment Register (PAR)

The pin assignment register (PAR), which is part of the system integration module (SIM), defines how each PAR bit determines each pin function, as shown in Figure 15-1.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	PAR15	PAR14	PAR13	PAR12	PAR11	PAR10	PAR9	PAR8	PAR7	PAR6	PAR5	PAR4	PAR3	PAR2	PAR1	PAR0
PAR[n] = 0	PP15	PP14	PP13	PP12	PP11	PP10	PP9	PP8	PP7	PP6	PP5	PP4	PP3	PP2	PP1	PP0
PAR[n] = 1	A31	A30	A29	A28	A27	A26	A25	A24	$\overline{\text{TIP}}$	$\overline{\text{DREQ0}}$	$\overline{\text{DREQ1}}$	TM2	TM1	TM0	TT1	TT0
Reset	Determined by driving D4/ADDR_CONFIG with a 1 or 0 when $\overline{\text{RSTI}}$ negates. The system is configured as PP[15:0] if D4 is low; otherwise alternate pin functions selected by PAR[n] = 1 are used.															
R/W	R/W															
Address	Address MBAR + 0x004															

Figure 15-1. Parallel Port Pin Assignment Register (PAR)

If PP[9:8]/A[25:24] are unavailable because A[25:0] are needed for external addressing, PP[15:10]/A[31:26] can be configured as general-purpose I/O. Table 15-1 summarizes MCF5307 parallel port pins, described in detail in Chapter 17, “Signal Descriptions.”

Table 15-1. Parallel Port Pin Descriptions

Pin	Description
PP[15:8]/ A[31:24]	MSB of the address bus/parallel port. Programmed through PAR[15–8]. If a PAR bit is 0, the associated pin functions as a parallel port signal. If a bit is 1, the pin functions as an address bus signal. If all pins are address signals, as much as 4 Gbytes of memory space are available.
$\overline{\text{TIP}}$/PP7	Transfer-in-progress output/parallel port bit 7. Programmed through PAR[7]. Assertion indicates a bus transfer is in progress; negation indicates an idle bus cycle if the bus is still granted to the processor. Note that $\overline{\text{TIP}}$ is held asserted on back-to-back bus cycles.
$\overline{\text{DREQ}}$[1:0]/ PP[6:5]	DMA request inputs/two bits of the parallel port. Programmed through PAR[6–5]. These inputs are asserted by a peripheral device to request a DMA transfer.
TM[2:0]/ PP[4:2]]	Transfer type outputs/parallel port bits 4–2. Programmed through PAR[4–2]. For DMA transfers, these signals provide acknowledge information. For emulation transfers, TM[2:0] indicate user or data transfer types. For CPU space transfers, TM[2:0] are low. For interrupt acknowledge transfers, TM[2:0] carry the interrupt level being acknowledged.
TT[1:0]/ PP[1:0]	Transfer type outputs/parallel port bits 1–0. Programmed through PAR[1–0]. When the MCF5307 is bus master, it outputs these signals. They indicate the current bus access type.

15.1.2 Port A Data Direction Register (PADDR)

The PADDR determines the signal direction of each parallel port pin programmed as a general-purpose I/O port in the PAR.

	15	0
Field	PADDR	
Reset	0000_0000_0000_0000	
R/W	R/W	
Address	Address MBAR + 0x244	

Figure 15-2. Port A Data Direction Register (PADDR)

Table 15-2 describes PADDR fields.

Table 15-2. PADDR Field Description

Bits	Name	Description
15–0	PADDR	Data direction bits. Each data direction bit selects the direction of the signal as follows: 0 Signal is defined as an input. 1 Signal is defined as an output.

15.1.3 Port A Data Register (PADAT)

The PADAT value for inputs corresponds to the logic level at the pin; for outputs, the value corresponds to the logic level driven onto the pin. Note the following:

- PADAT has no effect on pins not configured for general-purpose I/O.
- PADAT settings do not affect inputs. PADAT bit values determine the corresponding logic levels of pins configured as outputs.

- PADAT can be written to anytime. A read from PADAT returns values of corresponding pins configured as general-purpose I/O in the PAR and designated as inputs by the PADDR.

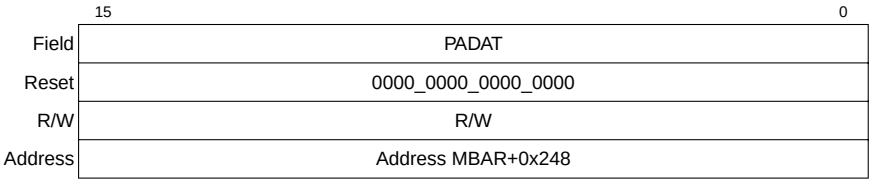


Figure 15-3. Port A Data Register (PADAT)

Table 15-3 shows relationships between PADAT bits and parallel port pins when PADAT is accessed. The effect differs when the parallel port pin is an input or output.

The following results occur when a parallel port pin is configured as an input:

- When the PADAT is read, the value returned is the logic value on the pin.
- When the PADAT is written, the register contents are updated without affecting the logic value on the pin.

The following results occur when a parallel port pin is configured as an output:

- When the PADAT is read, the register contents are returned and the pin is the logic value of the register.
- When the PADAT is written, the register contents are updated and the pin is the logic value of the register.

These relationships are also described in Table 15-3.

Table 15-3. Relationship between PADAT Register and Parallel Port Pin (PP)

PP Status	PADAT R/W	Effect on PADAT	Effect on PP
Input	Read	Register bit value is the pin's logic value	No effect. Source of logic value
	Write	Register contents updated	No effect on the logic value at the pin
Output	Read	Register contents are returned	Pin is the logic value of the register bit
	Write	Register contents updated	Pin is the logic value of the register bit

NOTE:

Although external devices cannot access the MCF5307's on-chip memories or MBAR, they can access any parallel port module registers in the SIM.

15.1.4 Code Example

The following code example shows how to set up the parallel port. Here, PP[7:0] are general-purpose I/O, PP[3:0] are inputs, and PP[7:4] are outputs.

```
MBARx    EQU   0x00010000
PAR      EQU   MBARx+0x004
PADDR    EQU   MBARx+0x244
PADAT    EQU   MBARx+0x248

move.l   #MBARx,D0       ;because MBAR is an internal register, MBARx is used as
movec    D0, MBAR        ;label for the memory map address
move.w   #0x00FF,D0
move.w   D0,PAR          ;set up the PAR. PP[7:0] set up as I/O
move.w   #0x00F0,D0
move.w   D0,PADDR        ;set PP[7:4] as outputs; PP[3:0] as inputs
move.b   #0xA0,D0
move.b   D0,PADAT        ;0xA0 written into PADAT; PP[7:4] being outputs,
                         ;PP[7:4] becomes 1010; i.e. PP7, PP5 = 1 and
                         ;PP6, PP4 = 0
```

Part IV
Hardware Interface

Intended Audience

Part IV is intended for hardware designers who need to know the functions and electrical characteristics of the MCF5407 interface. It includes a pinout, and both electrical and functional descriptions of the MCF5307 signals. It also describes how these signals interact to support the variety of bus operations shown in timing diagrams.

Contents

Part IV contains the following chapters:

- Chapter 16, “Mechanical Data,” provides a functional pin listing and package diagram for the MCF5307.
- Chapter 17, “Signal Descriptions,” provides an alphabetical listing of MCF5307 signals. This chapter describes the MCF5307 signals. In particular, it shows which are inputs or outputs, how they are multiplexed, which signals require pull-up resistors, and the state of each signal at reset.
- Chapter 18, “Bus Operation,” describes data transfers, error conditions, bus arbitration, and reset operations. It describes transfers initiated by the MCF5307 and by an external bus master, and includes detailed timing diagrams showing the interaction of signals in supported bus operations. Note that Chapter 11, “Synchronous/Asynchronous DRAM Controller Module,” describes DRAM cycles.
- Chapter 19, “IEEE 1149.1 Test Access Port (JTAG),” describes configuration and operation of the MCF5307 JTAG test implementation. It describes the use of JTAG instructions and provides information on how to disable JTAG functionality.
- Chapter 20, “Electrical Specifications,” describes AC and DC electrical specifications and thermal characteristics for the MCF5307. Because additional speeds may have become available since the publication of this book, consult Motorola’s ColdFire web page, http://www.motorola.com/coldfire, to confirm that this is the latest information.

Suggested Reading

The following literature may be helpful with respect to the topics in Part IV:

- *IEEE Standard Test Access Port and Boundary-Scan Architecture*
- *IEEE Supplement to Standard Test Access Port and Boundary-Scan Architecture (1149.1)*

Acronyms and Abbreviations

Table IV-i describes acronyms and abbreviations used in Part IV.

Table IV-i. Acronyms and Abbreviated Terms

Term	Meaning
BDM	Background debug mode
BIST	Built-in self test
BSDL	Boundary-scan description language
DMA	Direct memory access
DSP	Digital signal processing
EDO	Extended data output (DRAM)
GPIO	
I^2C	Inter-integrated circuit
IEEE	Institute for Electrical and Electronics Engineers
IPL	Interrupt priority level
JEDEC	Joint Electron Device Engineering Council
JTAG	Joint Test Action Group
LSB	Least-significant byte
lsb	Least-significant bit
MAC	Multiple accumulate unit
MBAR	Memory base address register
MSB	Most-significant byte
msb	Most-significant bit
Mux	Multiplex
PCLK	Processor clock
PLL	Phase-locked loop
POR	Power-on reset
PQFP	Plastic quad flat pack

Table IV-i. Acronyms and Abbreviated Terms (Continued)

Term	Meaning
RISC	Reduced instruction set computing
Rx	Receive
SIM	System integration module
TAP	Test access port
TTL	Transistor-to-transistor logic
Tx	Transmit

Chapter 16
Mechanical Data

This chapter provides a function pin listing and package diagram for the MCF5307. See the website [http://www.motorola.com/coldfire] for any updated information.

16.1 Package

The MCF5307 is assembled in a 208-pin, thermally enhanced plastic QFP package.

16.2 Pinout

The MCF5307 pinout is detailed in the following tables, including the primary and secondary functions of multiplexed signals. Additional columns indicate the output drive capability of each pin, whether it is internally synchronized, and if the signal can change on a negative clock transition.

These tables show MCF5307 pin numbers, including signal multiplexing. Additional columns indicate the direction, description, and output drive capability of each pin.

Table 16-1. Pins 1–52 (Left, Top-to-Bottom)

Pin		Alternate Function	I/O	Description	Drive (mA)
No	Name				
1	VCC	—	—	Power input	—
2	A0	—	I/O	Address bus bit	8
3	A1	—	I/O	Address bus bit	8
4	GND	—	—	Ground pin	—
5	A2	—	I/O	Address bus bit	8
6	A3	—	I/O	Address bus bit	8
7	VCC	—	—	Power input	—
8	A4	—	I/O	Address bus bit	8
9	A5	—	I/O	Address bus bit	8
10	GND	—	—	Ground pin	—
11	A6	—	I/O	Address bus bit	8
12	A7	—	I/O	Address bus bit	8

Table 16-1. Pins 1–52 (Left, Top-to-Bottom) (Continued)

Pin		Alternate Function	I/O	Description	Drive (mA)
No	Name				
13	VCC	—	—	Power input	—
14	A8	—	I/O	Address bus bit	8
15	A9	—	I/O	Address bus bit	8
16	A10	—	I/O	Address bus bit	8
17	GND	—	—	Ground pin	—
18	A11	—	I/O	Address bus bit	8
19	A12	—	I/O	Address bus bit	8
20	A13	—	I/O	Address bus bit	8
21	VCC	—	—	Power input	—
22	A14	—	I/O	Address bus bit	8
23	A15	—	I/O	Address bus bit	8
24	A16	—	I/O	Address bus bit	8
25	GND	—	—	Ground pin	—
26	A17	—	I/O	Address bus bit	8
27	A18	—	I/O	Address bus bit	8
28	A19	—	I/O	Address bus bit	8
29	VCC	—	—	Power input	—
30	A20	—	I/O	Address bus bit	8
31	A21	—	I/O	Address bus bit	8
32	A22	—	I/O	Address bus bit	8
33	GND	—	—	Ground pin	—
34	A23	—	I/O	Address bus bit	8
35	PP8	A24	I/O	Parallel port bit/Address bus bit	8
36	PP9	A25	I/O	Parallel port bit/Address bus bit	8
37	VCC	—	—	Power input	—
38	PP10	A26	I/O	Parallel port bit/Address bus bit	8
39	PP11	A27	I/O	Parallel port bit/Address bus bit	8
40	PP12	A28	I/O	Parallel port bit/Address bus bit	8
41	GND	—	—	Ground pin	—
42	PP13	A29	I/O	Parallel port bit/Address bus bit	8
43	PP14	A30	I/O	Parallel port bit/Address bus bit	8
44	PP15	A31	I/O	Parallel port bit/Address bus bit	8
45	VCC	—	—	Power input	—
46	SIZ0	—	I/O	Size attribute	8
47	SIZ1	—	I/O	Size attribute	8

Table 16-1. Pins 1–52 (Left, Top-to-Bottom) (Continued)

Pin		Alternate Function	I/O	Description	Drive (mA)
No	Name				
48	GND	—	—	Ground pin	—
49	$\overline{OE}$	—	O	Output enable for chip selects	8
50	$\overline{CS0}$	—	O	Chip select	8
51	$\overline{CS1}$	—	O	Chip select	8
52	VCC	—	—	Power input	—

Table 16-2. Pins 53–104 (Bottom, Left-to-Right)

Pin		Alternate Function	I/O	Description	Drive (mA)
No	Name				
53	GND	—	—	Ground pin	—
54	$\overline{CS2}$	—	O	Chip select	8
55	$\overline{CS3}$	—	O	Chip select	8
56	$\overline{CS4}$	—	O	Chip select	8
57	VCC	—	—	Power input	—
58	$\overline{CS5}$	—	O	Chip select	8
59	$\overline{CS6}$	—	O	Chip select	8
60	$\overline{CS7}$	—	O	Chip select	8
61	GND	—	—	Ground pin	—
62	$\overline{AS}$	—	I/O	Address strobe	8
63	R/$\overline{W}$	—	I/O	Read/$\overline{Write}$	8
64	$\overline{TA}$	—	I/O	Transfer acknowledge	8
65	VCC	—	—	Power input	—
66	$\overline{TS}$	—	I/O	Transfer start	8
67	$\overline{RSTI}$	—	I	Reset	—
68	$\overline{IRQ7}$	—	I	Interrupt request	—
69	GND	—	—	Ground pin	—
70	$\overline{IRQ5}$	$\overline{IRQ4}$	I	Interrupt request	—
71	$\overline{IRQ3}$	$\overline{IRQ6}$	I	Interrupt request	—
72	$\overline{IRQ1}$	$\overline{IRQ2}$	I	Interrupt request	—
73	VCC	—	—	Power input	—
74	$\overline{BR}$	—	O	Bus request	8
75	$\overline{BD}$	—	O	Bus driven	8
76	$\overline{BG}$	—	I	Bus grant	—
77	GND	—	—	Ground pin	—

Table 16-2. Pins 53–104 (Bottom, Left-to-Right) (Continued)

Pin		Alternate Function	I/O	Description	Drive (mA)
No	Name				
78	TOUT1	—	O	Timer output	8
79	TOUT0	—	O	Timer output	8
80	TIN0	—	I	Timer input	—
81	VCC	—	—	Power input	—
82	TIN1	—	I	Timer input	—
83	$\overline{\text{RAS0}}$	—	O	DRAM row address strobe	16
84	$\overline{\text{RAS1}}$	—	O	DRAM row address strobe	16
85	GND	—	—	Ground pin	—
86	$\overline{\text{CAS0}}$	—	O	DRAM column address strobe	16
87	$\overline{\text{CAS1}}$	—	O	DRAM column address strobe	16
88	$\overline{\text{CAS2}}$	—	O	DRAM column address strobe	16
89	VCC	—	—	Power input	—
90	$\overline{\text{CAS3}}$	—	O	DRAM column address strobe	16
91	$\overline{\text{DRAMW}}$	—	O	DRAM write	16
92	$\overline{\text{SRAS}}$	—	O	SDRAM row address strobe	16
93	GND	—	—	Ground pin	—
94	$\overline{\text{SCAS}}$	—	O	SDRAM column address strobe	16
95	SCKE	—	O	SDRAM clock enable	16
96	$\overline{\text{BE0}}$	$\overline{\text{BWE0}}$	O	Byte enable/byte write enable	8
97	VCC	—	—	Power input	—
98	$\overline{\text{BE1}}$	$\overline{\text{BWE1}}$	O	Byte enable/byte write enable	8
99	$\overline{\text{BE2}}$	$\overline{\text{BWE2}}$	O	Byte enable/byte write enable	8
100	$\overline{\text{BE3}}$	$\overline{\text{BWE3}}$	O	Byte enable/byte write enable	8
101	GND	—	—	Ground pin	—
102	SCL	—	I/OD [1]	Serial clock line	8
103	SDA	—	I/OD [1]	Serial data line	8
104	GND	—	—	Ground pin	—

[1] OD: Open-drain output

Table 16-3. Pins 105–156 (Right, Bottom-to-Top)

Pin		Alternate Function	I/O	Description	Drive (mA)
No	Name				
105	VCC	—	—	Power input	—
106	D31	—	I/O	Data bus	8

Table 16-3. Pins 105–156 (Right, Bottom-to-Top) (Continued)

Pin		Alternate Function	I/O	Description	Drive (mA)
No	Name				
107	D30	—	I/O	Data bus	8
108	D29	—	I/O	Data bus	8
109	GND	—	—	Ground pin	—
110	D28	—	I/O	Data bus	8
111	D27	—	I/O	Data bus	8
112	D26	—	I/O	Data bus	8
113	VCC	—	—	Power input	—
114	D25	—	I/O	Data bus	8
115	D24	—	I/O	Data bus	8
116	D23	—	I/O	Data bus	8
117	GND	—	—	Ground pin	—
118	D22	—	I/O	Data bus	8
119	D21	—	I/O	Data bus	8
120	D20	—	I/O	Data bus	8
121	VCC	—	—	Power input	—
122	D19	—	I/O	Data bus	8
123	D18	—	I/O	Data bus	8
124	D17	—	I/O	Data bus	8
125	GND	—	—	Ground pin	—
126	D16	—	I/O	Data bus	8
127	D15	—	I/O	Data bus	8
128	D14	—	I/O	Data bus	8
129	VCC	—	—	Power input	—
130	D13	—	I/O	Data bus	8
131	D12	—	I/O	Data bus	8
132	D11	—	I/O	Data bus	8
133	GND	—	—	Ground pin	—
134	D10	—	I/O	Data bus	8
135	D9	—	I/O	Data bus	8
136	D8	—	I/O	Data bus	8
137	VCC	—	—	Power input	—
138	D7	CS_CONF2	I/O	Data bus/Chip select configuration	8
139	D6	CS_CONF1	I/O	Data bus/Chip select configuration	8
140	D5	CS_CONF0	I/O	Data bus/Chip select configuration	8
141	GND	—	—	Ground pin	—

Table 16-3. Pins 105–156 (Right, Bottom-to-Top) (Continued)

Pin		Alternate Function	I/O	Description	Drive (mA)
No	Name				
142	D4	ADDR_CONF	I/O	Data bus/Address configuration	8
143	D3	FREQ1	I/O	Data bus/CLKIN Frequency	8
144	D2	FREQ0	I/O	Data bus/CLKIN Frequency	8
145	VCC	—	—	Power input	—
146	D1	DIVIDE1	I/O	Data bus/Divide control PCLK:BCLKO	8
147	D0	DIVIDE0	I/O	Data bus/Divide control PCLK:BCLKO	8
148	GND	—	—	Ground pin	—
149	DSCLK	$\overline{\text{TRST}}$	I	Debug serial clock/JTAG Reset	—
150	TCK	TCK	I	JTAG clock	—
151	DSO	TDO	O	Debug serial out/JTAG data out	8
152	VCC	—	—	Power input	—
153	DSI	TDI	I	Debug serial input/JTAG data in	—
154	$\overline{\text{BKPT}}$	TMS	I	Debug breakpoint/JTAG mode select	—
155	$\overline{\text{HIZ}}$	—	I	High impedance override	—
156	GND	—	—	Ground pin	—

Table 16-4. Pins 157–208 (Top, Right-to-Left)

Pin		Alternate Function	I/O	Description	Drive (mA)
No	Name				
157	VCC	—	—	Power input	—
158	$\overline{\text{CTS1}}$	—	I	UART1 clear-to-send	—
159	$\overline{\text{RTS1}}$	—	O	UART1 request-to-send	8
160	RXD1	—	I	UART1 receive data	—
161	TXD1	—	O	UART1 transmit data	8
162	GND	—	—	Ground pin	—
163	$\overline{\text{CTS0}}$	—	I	UART0 clear-to-send	—
164	$\overline{\text{RTS0}}$	—	O	UART0 request-to-send	8
165	RXD0	—	I	UART0 receive data	—
166	TXD0	—	O	UART0 transmit data	8
167	VCC	—	—	Power input	—
168	EDGESEL	—	I	SDRAM bus clock edge select	—
169	GND	—	—	Ground pin	—
170	BCLKO	—	O	Bus clock output	16

Table 16-4. Pins 157–208 (Top, Right-to-Left) (Continued)

Pin		Alternate Function	I/O	Description	Drive (mA)
No	Name				
171	VCC	—	—	Power input	—
172	$\overline{\text{RSTO}}$	—	O	Processor reset output	8
173	GND	—	—	Ground pin	—
174	CLKIN	—	I	Clock input	—
175	VCC	—	—	Power input	—
176	MTMOD0	—	I	JTAG/BDM select (Tie high or low)	—
177	MTMOD1	—	I	Tie high or low	—
178	PGND	—	—	PLL ground pin	—
179	NC	—	O		—
180	PVCC	—	—	Filter supply for PLL	—
181	MTMOD2	—	I	Tie high or low	—
182	MTMOD3	—	I	Tie high or low	—
183	GND	—	—	Ground pin	—
184	PSTCLK	—	O	Processor status clock	8
185	VCC	—	—	Power input	—
186	DDATA0	—	O	Debug data	8
187	DDATA1	—	O	Debug data	8
188	GND	—	—	Ground pin	—
189	DDATA2	—	O	Debug data	8
190	DDATA3	—	O	Debug data	8
191	VCC	—	—	Power input	—
192	PST0	—	O	Processor status	8
193	PST1	—	O	Processor status	8
194	GND	—	—	Ground pin	—
195	PST2	—	O	Processor status	8
196	PST3	—	O	Processor status	8
197	VCC	—	—	Power input	—
198	PP7	$\overline{\text{TIP}}$	I/O	Parallel port bit/transfer in progress	8
199	PP6	$\overline{\text{DREQ0}}$	I/O	Parallel port bit/DMA request	8
200	PP5	$\overline{\text{DREQ1}}$	I/O	Parallel port bit/DMA request	8
201	GND	—	—	Ground pin	—
202	PP4	TM2	I/O	Parallel port bit/Transfer modifier	8
203	PP3	TM1	I/O	Parallel port bit/Transfer modifier	8
204	PP2	TM0	I/O	Parallel port bit/Transfer modifier	8
205	VCC	—	—	Power input	—

Table 16-4. Pins 157–208 (Top, Right-to-Left) (Continued)

Pin		Alternate Function	I/O	Description	Drive (mA)
No	Name				
206	PP1	TT1	I/O	Parallel port bit/Transfer type	8
207	PP0	TT0	I/O	Parallel port bit/Transfer type	8
208	GND	—	—	Ground pin	—

16.3 Mechanical Diagram

Figure 16-1 is a mechanical diagram of the 208-pin QFP MCF5307.

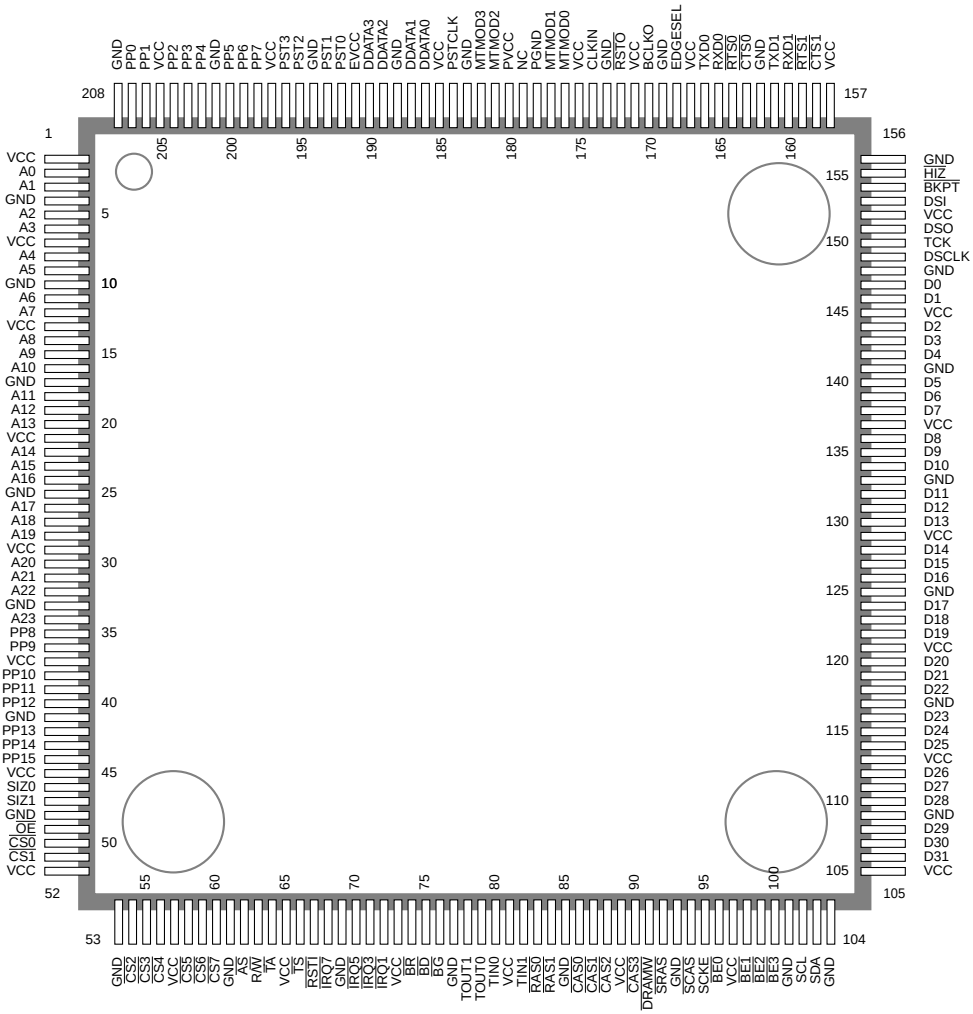


Figure 16-1. Mechanical Diagram

16.4 Case Drawing

Figure 16-2 and Figure 16-3 show the MCF5307 case drawings.

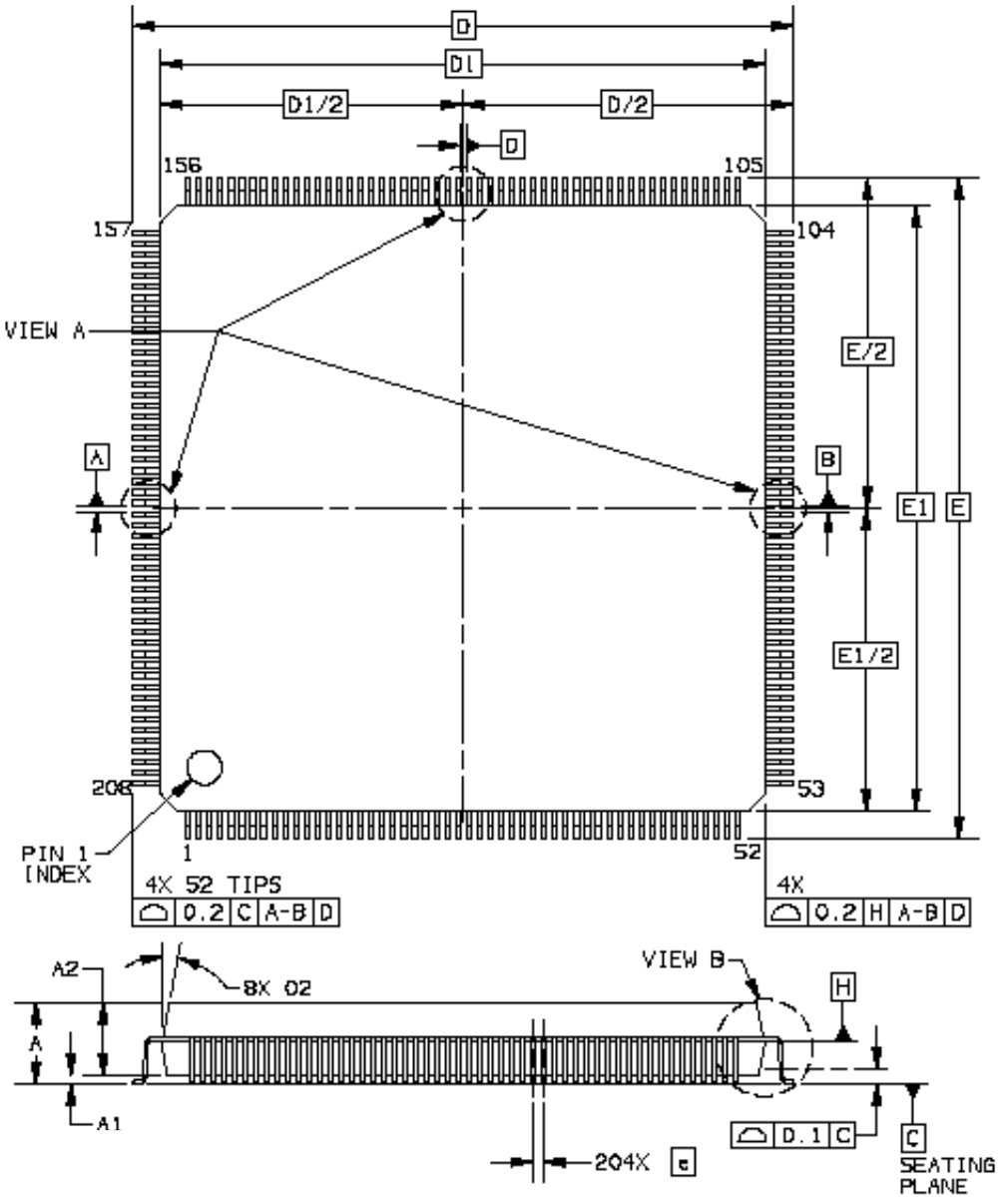
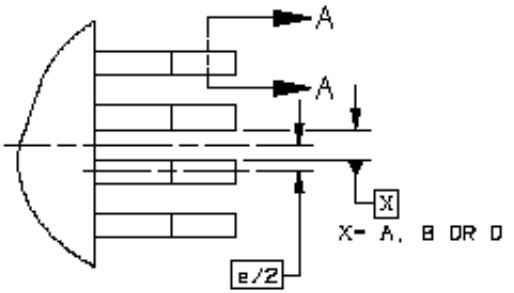
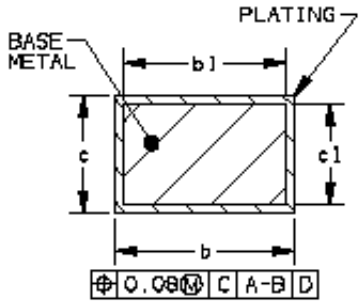


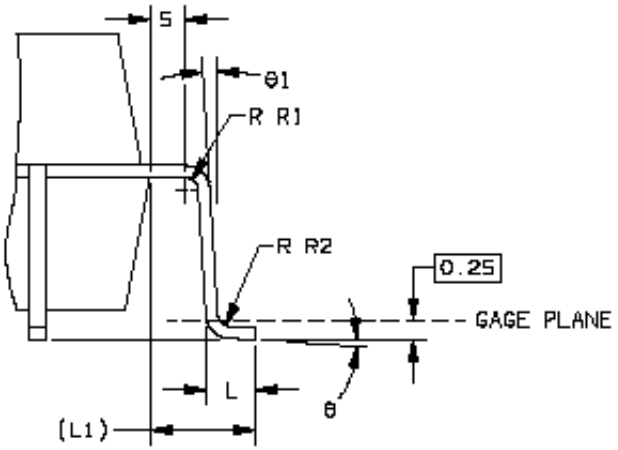
Figure 16-2. MCF5307 Case Drawing (General View)



View A: Three Places



Section A-A: 160 Places Rotated 90° CW



View B

Figure 16-3. Case Drawing (Details)

The dimensions in Figure 16-2 and Figure 16-3 are referenced in Table 16-5.

Table 16-5. Dimensions

Reference	Dimension (Millimeters)	
	Minimum	Maximum
A	—	4.10
A1	0.25	0.50
A2	3.20	3.60
b	0.17	0.27
b1	0.17	0.23
c	0.09	0.20
c1	0.09	0.16
D	30.60 BSC	

Table 16-5. Dimensions (Continued)

Reference	Dimension (Millimeters)	
	Minimum	Maximum
D1	28.00 BSC	
e	0.50 BSC	
E	30.60 BSC	
E1	28.00 BSC	
L	0.45	0.75
L1	1.30 REF	
R1	0.08	—
R2	0.08	0.25
S	0.20	—
ϑ	0*	8*
ϑ1	0*	—
ϑ2	5*	16*

Chapter 17
Signal Descriptions

This chapter describes MCF5307 signals. It includes an alphabetical listing of signals, showing multiplexing, whether it is an input or output to the MCF5307, the state at reset, and whether a pull-up resistor should be used. The following chapter, Chapter 18, “Bus Operation,” describes how these signals interact.

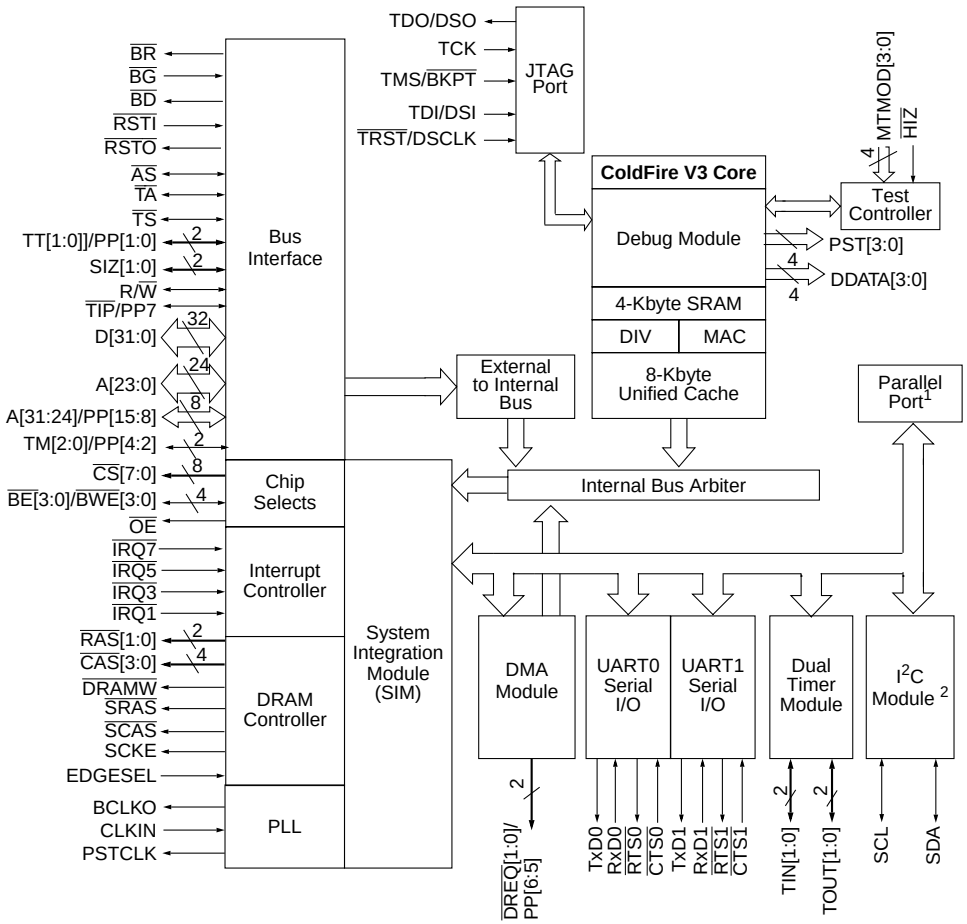
NOTE:

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.

17.1 Overview

Figure 17-1 shows the block diagram of the MCF5307 with the signal interface.



[1] Note: Parallel port pins (PPn) are multiplexed with other bus functions as shown.
[2] I2C is a Philips proprietary interface

Figure 17-1. MCF5307 Block Diagram with Signal Interfaces

Table 17-1 lists the MCF5307 signals grouped by functionality.

Table 17-1. MCF5307 Signal Index

Signal Name	Abbreviation	Function	I/O	Reset	Pull-Up	Page
Section 17.2, "MCF5307 Bus Signals"						17-7
Address	A[31:0]	32-bit address bus. A[4:2] indicate the interrupt level for external interrupts.	I/O	Three state		17-7
Data	D[31:0]	Data bus. D[7:0] are loaded at reset for bus configuration.	I/O	Three state		17-8
Read/Write	R/$\overline{W}$	Identifies read and write transfers	I/O	Three state	Up	17-8
Size	SIZ[1:0]	Indicates the data transfer size	I/O	Three state		17-8
Transfer start	$\overline{TS}$	Indicates the start of a bus transfer	I/O	Three state		17-9
Address strobe	$\overline{AS}$	Indicates a bus cycle has been initiated and address is stable	I/O	Three state	Up	17-9
Transfer acknowledge	$\overline{TA}$	Assertion terminates transfer synchronously	I/O	Three state	Up	17-9
Transfer in progress	$\overline{TIP}$/PP7	Indicates a bus cycle is in progress; multiplexed with PP7	O	Parallel port		17-10
Transfer type	TT[1:0]	Indicates transfer type: normal, CPU space, emulator mode, or DMA; multiplexed with PP[1:0]	O	Parallel port		17-10
Transfer modifier	TM[2:0]	Provides transfer modifier information; Multiplexed with PP[4:2].	O	Parallel port		17-10
Section 17.3, "Interrupt Control Signals"						17-12
Interrupt request	$\overline{IRQ7}$, $\overline{IRQ5}$, $\overline{IRQ3}$, $\overline{IRQ1}$	Four external interrupts are set to default levels 1,3,5,7; user-alterable.	I	—	Up	17-12
Section 17.4, "Bus Arbitration Signals"						17-12
Bus request	$\overline{BR}$	Indicates processor needs bus	O	High		17-12
Bus grant	$\overline{BG}$	Arbiter asserts to grant mastership.	I	—	Note [1]	17-12
Bus driven	$\overline{BD}$	Indicates processor is driving bus	O	High		17-13
Section 17.5, "Clock and Reset Signals"						17-13
Reset in	$\overline{RSTI}$	Processor reset input	I	—	Up	17-13
Clock input	CLKIN	Input used to clock internal logic	I	—		17-13
Bus clock out	BCLKO	Bus clock reference output	O	—		17-13
Reset out	$\overline{RSTO}$	Processor reset output	O	Low		17-13
Auto-acknowledge configuration [2]	AA_CONFIG	Controls auto acknowledge timing for $\overline{CS0}$ at reset	I	—		17-14
Port size configuration [2]	PS_CONFIG[1:0]	Controls port size for $\overline{CS0}$ at reset	I	—	User cfg	17-14
Address configuration [2]	ADDR_CONFIG	Programs parallel I/O ports	I	—	User cfg	17-14

Table 17-1. MCF5307 Signal Index (Continued)

Signal Name	Abbreviation	Function	I/O	Reset	Pull-Up	Page
Frequency control PLL	FREQ[1:0]	Indicates CLKIN frequency range.	I			17-15
Divide control PCLK to BCLKO	DIVIDE[1:0]	Indicates the BCLKO/PSTCLK ratio.	I			17-15
Section 17.6, "Chip-Select Module Signals"						17-15
Chip selects[7:0]	$\overline{CS}$[7:0]	Enables peripherals at programmed addresses; $\overline{CS0}$ provides boot ROM selection.	O	High		17-16
Byte enable[3:0]/ Byte write enable[3:0]	$\overline{BE}$[3:0]/ $\overline{BWE}$[3:0]	$\overline{BE}$[3:0] select bytes in memory.	O	High		17-16
Output enable	$\overline{OE}$	Output enable for chip select read cycles	O	High		17-16
Section 17.7, "DRAM Controller Signals"						17-16
Row address strobe	$\overline{RAS}$[1:0]	DRAM row address strobe	O	High		17-16
Column address strobe	$\overline{CAS}$[3:0]	DRAM column address strobe	O	High		17-16
DRAM write	$\overline{DRAMW}$	Asserted for DRAM write; negated for DRAM read	O	High		17-17
Synchronous column address strobe	$\overline{SCAS}$	SDRAM column address strobe	O	High		17-17
Synchronous row address strobe	$\overline{SRAS}$	SDRAM row address strobe	O	High		17-17
Synchronous clock enable	SCKE	Clock enable for external SDRAM	O	Low		17-17
Synchronous edge select	EDGESEL	Timing select for external SDRAM	I	—	User cfg	17-17
Section 17.8, "DMA Controller Module Signals"						17-17
DMA request	$\overline{DREQ}$[1:0]	External DMA transfer request; multiplexed with PP[6:5]	I	—		17-18
Section 17.9, "Serial Module Signals"						17-18
Receive data	RxD[1:0]	Receive serial data input for UART	I	—		17-18
Transmit data	TxD[1:0]	Transmit serial data output for UART	O	High		17-18
Request-to-send	$\overline{RTS}$[1:0]	UART asserts when ready to receive data query.	O	High		17-18
Clear-to-send	$\overline{CTS}$[1:0]	Signals UART that data can be sent to peripheral	I	—		17-18
Section 17.10, "Timer Module Signals"						17-18
Timer input	TIN[1:0]	Clock input to timer or trigger to timer value capture logic	I	—		17-19
Timer outputs	TOUT[1:0]	Outputs waveform or pulse.	O	High		17-19
Section 17.11, "Parallel I/O Port (PP[15:0])"						17-19

Table 17-1. MCF5307 Signal Index (Continued)

Signal Name	Abbreviation	Function	I/O	Reset	Pull-Up	Page
Parallel port	PP[15:0]	Interfaces with I/O; multiplexed with bus address and attribute signals.	I/O	Input		17-19
Section 17.12, "I²C Module Signals"						17-19
Serial clock line	SCL	Clock signal for I²C operation	I/O	Open drain	Up	17-19
Serial data line	SDA	Serial data port for I²C operation	I/O	Open drain	Up	17-19
Section 17.13, "Debug and Test Signals"						17-20
Motorola test mode	MTMOD0	Puts processor in functional or emulator mode	I	—	User cfg	17-20
Motorola test mode	MTMOD[3:1]	Reserved	I	—	Down	17-20
High impedance	$\overline{\text{HIZ}}$	Assertion three-states all outputs	I	—	Up	17-20
Processor clock out	PSTCLK	Output clock used for PSTDDATA	O	—		17-20
Processor status	PST[3:0]	Displays captured processor data .	O	Driven		17-20
Debug data	DDATA[3:0]	Displays captured processor data and breakpoint status.	O	Driven		17-20
Section 17.14, "Debug Module/JTAG Signals"						17-21
Test clock	TCK	Clock signal for IEEE 1149.1 JTAG	I	—	Low	17-23
Test reset/ Development serial clock	$\overline{\text{TRST}}$/DSCLK	Asynchronous reset for JTAG; debug module clock input	I	—	Up	17-21
Test mode select/ Breakpoint	TMS/$\overline{\text{BKPT}}$	TMS (JTAG)/hardware breakpoint (debug)	I	—	Up	17-22
Test data input/ Development serial input	TDI/DSI	Multiplexed serial input for the JTAG or background debug module	I	—	Up	17-22
Test data output/ Development serial output	TDO/DSO	Multiplexed serial output for the JTAG or background debug module	O	Driven		17-22

1 If there is no arbiter, $\overline{\text{BG}}$ should be tied low; otherwise, it should be negated.

2 These data pins are sampled at reset for configuration.

Table 17-2 lists signals in alphabetical order by abbreviated name.

Table 17-2. MCF507 Alphabetical Signal Index

Abbreviation	Signal Name	Function	I/O	Page
AA_CONFIG	Auto-acknowledge configuration	Clock/reset	I	17-14
ADDR_CONFIG	Address configuration	Clock/reset	I	17-14
$\overline{\text{AS}}$	Address strobe	Bus	I/O	17-9
A[31:0]	Address	Bus	I/O	17-7

Table 17-2. MCF507 Alphabetical Signal Index (Continued)

Abbreviation	Signal Name	Function	I/O	Page
BCLKO	Bus clock out	Clock/reset	O	17-13
$\overline{BD}$	Bus driven	Bus arbitration	O	17-13
$\overline{BE}$[3:0]/$\overline{BWE}$[3:0]	Byte enable[3:0]/Byte write enable[3:0]	Chip select	O	17-16
$\overline{BG}$	Bus grant	Bus arbitration	I	17-12
$\overline{BR}$	Bus request	Bus arbitration	O	17-12
$\overline{CAS}$[3:0]	Column address strobe	DRAM	O	17-16
CLKIN	Clock input	Clock/reset	I	17-13
$\overline{CS}$[7:0]	Chip selects[7:0]	UART	O	17-16
$\overline{CTS}$[1:0]	Clear-to-send	Serial module	I	17-18
DDATA[3:0]	Debug data	Debug	O	17-20
		Clock/Reset	I	17-15
$\overline{DRAMW}$	DRAM write	DRAM	O	17-17
$\overline{DREQ}$[1:0]	DMA request	DMA	I	17-18
D[31:0]	Data	Bus	I/O	17-8
EDGESEL	Sync edge select	DRAM	I	17-17
		Clock/Reset	I	17-15
$\overline{HIZ}$	High impedance	Debug	I	17-20
$\overline{IRQ7}$, $\overline{IRQ5}$, $\overline{IRQ3}$, $\overline{IRQ1}$	Interrupt request	Interrupt control	I	17-12
MTMOD[3:0]	Motorola test mode	Debug	I	17-20
$\overline{OE}$	Output enable	Chip select	O	17-16
PP[15:0]	Parallel port	Parallel port	I/O	17-19
PSTCLK	Processor clock out	Debug	O	17-20
PST[:0]	Processor status	Debug	O	17-20
PS_CONFIG[1:0]	Port size configuration	Clock/reset	I	17-14
R/$\overline{W}$	Read/Write	Bus	I/O	17-8
$\overline{RAS}$[1:0]	Row address strobe	DRAM	O	17-16
$\overline{RSTI}$	Reset In	Clock/reset	I	17-13
$\overline{RSTO}$	Reset Out	Clock/reset	O	17-13
$\overline{RTS}$[1:0]	Request-to-send	Serial module	O	17-18
RxD[1:0]	Receive data	Serial module	I	17-18
$\overline{SCAS}$	Synchronous column address strobe	DRAM	O	17-17
SCKE	Synchronous clock enable	DRAM	O	17-17
SCL	Serial clock line	I^2C	I/O	17-19
SDA	Serial data line	I^2C	I/O	17-19
SIZ[1:0]	Size	Bus	I/O	17-8

Table 17-2. MCF507 Alphabetical Signal Index (Continued)

Abbreviation	Signal Name	Function	I/O	Page
$\overline{\text{SRAS}}$	Synchronous row address strobe	DRAM	O	17-17
$\overline{\text{TA}}$	Transfer acknowledge	Bus	I/O	17-9
TCK	Test clock	JTAG	I	17-23
TDI/DSI	Test data input/Development serial input	JTAG	I	17-22
TDO/DSO	Test data output/Development serial output	JTAG	O	17-22
TIN[1:0]	Timer input	Timer	I	17-19
$\overline{\text{TIP}}$	Transfer in progress	Bus	O	17-10
TMS/$\overline{\text{BKPT}}$	Test mode select/Breakpoint	JTAG	I	17-22
TM[2:0]	Transfer modifier	Bus	O	17-10
TOUT[1:0]	Timer outputs	Timer	O	17-19
$\overline{\text{TRST}}$/DSCLK	Test reset/Development serial clock	JTAG	I	17-21
$\overline{\text{TS}}$	Transfer start	Bus	I/O	17-9
TT[1:0]	Transfer type	Bus	O	17-10
TxD[1:0]	Transmit data	Serial module	O	17-18

17.2 MCF5307 Bus Signals

The bus signals provide the external bus interface to the MCF5307.

17.2.1 Address Bus

The address bus provides the address of the byte or most-significant byte (MSB) of the word or longword being transferred. The address lines also serve as the DRAM addressing, providing multiplexed row and column address signals. When an external device has ownership of the MCF5307 bus, the device must drive the address bus and assert $\overline{\text{TS}}$ or $\overline{\text{AS}}$ to indicate the start of a bus cycle. During an interrupt acknowledge access, A[4:2] indicate the interrupt level being acknowledged.

17.2.1.1 Address Bus (A[23:0])

The lower 24 bits of the address bus become valid when $\overline{\text{TS}}$ is asserted. A[4:2] indicate the interrupt level during interrupt acknowledge cycles.

17.2.1.2 Address Bus (A[31:24]/PP[15:8])

These multiplexed pins can serve as the most-significant byte of the address bus, or as the most-significant byte of the parallel port. Programming the PAR in the system integration module (SIM) determines the function of each of these eight multiplexed pins. These pins are programmable on a bit-by-bit basis.

- A[31:24]—Pins are configured as address bits by setting corresponding PAR bits; they represent the most-significant address bus bits. As much as 4 Gbytes of memory are available when all of these pins are programmed as address signals.
- PP[15:8]—Pins are configured as parallel port signals by clearing corresponding PAR bits; these represent the most-significant parallel port bits.

17.2.2 Data Bus (D[31:0])

The data bus is bidirectional and non-multiplexed. Data is sampled by the MCF5307 on the rising BCLKO edge. The data bus port width, wait states, and internal termination are initially defined for the boot chip select by D[7:0] during reset. The port width for each chip select and DRAM bank are programmable. The data bus uses a default configuration if none of the chip selects or DRAM 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[7:0] are used during reset initialization as inputs to configure the functions as described in Table 17-3. They are defined in Section 17.5.5, “Data/Configuration Pins (D[7:0]).”

Table 17-3. Data Pin Configuration

Pin	Function	Section
D7	Auto-acknowledge configuration (AA_CONFIG)	Section 17.5.5.2, “D7—Auto Acknowledge Configuration (AA_CONFIG)”
D[6:5]	Port size configuration (PS_CONFIG[1:0])	Section 17.5.5.3, “D[6:5]—Port Size Configuration (PS_CONFIG[1:0])”
D4	Address configuration (ADDR_CONFIG/D4)	Section 17.5.6, “D4—Address Configuration (ADDR_CONFIG)”
D[3:2]	Frequency Control PLL (FREQ[1:0])	Section 17.5.7, “D[3:2]—Frequency Control PLL (FREQ[1:0])
D[1:0]	Divide Control (DIVIDE[1:0])	Section 17.5.8, “D[1:0]—Divide Control PCLK to BCLKO (DIVIDE[1:0])

17.2.3 Read/Write (R/$\overline{W}$)

When the MCF5307 is the bus master, it drives the R/$\overline{W}$ signal to indicate the direction of subsequent data transfers. It is driven high during read bus cycles and driven low during write bus cycles. This signal is an input during an external master access.

17.2.4 Size (SIZ[1:0])

When it is the bus master, the MCF5307 outputs these signals to indicate the requested data transfer size. Table 17-4 shows the definition of the bus request size encodings. When the MCF5307 device is not the bus master, these signals function as inputs.

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.

Table 17-4. Bus Cycle Size Encoding

SIZ[1:0]	Port Size
00	Longword
01	Byte
10	Word
11	Line

For burst-inhibited transfers, SIZ[1:0] changes with each $\overline{TS}$ 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.

17.2.5 Transfer Start ($\overline{TS}$)

The MCF5307 asserts $\overline{TS}$ during the first clock cycle when address and attributes (TM, TT, $\overline{TIP}$, R/$\overline{W}$, and SIZ) are valid. $\overline{TS}$ is negated in the following clock cycle. When the MCF5307 is not the bus master, $\overline{TS}$ is an input.

17.2.6 Address Strobe ($\overline{AS}$)

Address strobe ($\overline{AS}$) is asserted to indicate when the address is stable at the start of a bus cycle. The address and attributes are guaranteed to be valid during the entire period that $\overline{AS}$ is asserted. This signal is asserted and negated on the falling edge of the clock. When the MCF5307 is not the bus master, $\overline{AS}$ is an input.

17.2.7 Transfer Acknowledge ($\overline{TA}$)

When the MCF5307 is bus master, the external system drives this input to terminate the bus transfer. The bus continues to be driven until this synchronous signal is asserted. For write cycles, the processor continues to drive data one clock after $\overline{TA}$ is asserted. During read cycles, the peripheral must continue to drive data until $\overline{TA}$ is recognized.

If all bus cycles support fast termination, $\overline{TA}$ can be tied low. However, note that $\overline{TA}$ cannot be tied low if potential external bus masters are present. The MCF5307 drives $\overline{TA}$ for an

external master access. This condition is indicated by the AM bit in the chip-select mask register (CSMR) being cleared. See Chapter 10, “Chip-Select Module.”

17.2.8 Transfer In Progress ($\overline{\text{TIP}}$/PP7)

The $\overline{\text{TIP}}$/PP7 pin is programmed in the PAR to serve as the transfer-in-progress output or as a parallel port bits. The $\overline{\text{TIP}}$ output is asserted indicating a bus transfer is in progress. It is negated during idle bus cycles if the bus is still granted to the processor. It is three-stated for external master accesses. Note that $\overline{\text{TIP}}$ is held asserted on back-to-back bus cycles.

17.2.9 Transfer Type (TT[1:0]/PP[1:0])

The TT[1:0]/PP[1:0] pins are programmed in the PAR to serve as the transfer type outputs or as two parallel port bits. When the MCF5307 is bus master and TT[1:0] are enabled, these signals are driven as outputs only. If an external master owns the bus and TT[1:0] are enabled, these pins are three-stated by the MCF5307 and can be driven by the external master. Table 17-5 shows the definition of the encodings.

Table 17-5. Bus Cycle Transfer Type Encoding

TT[1:0]	Transfer Type
00	Normal access
01	DMA access
10	Emulator access
11	CPU space or interrupt acknowledge

17.2.10 Transfer Modifier (TM[2:0]/PP[4:2])

The TM[2:0]/PP[4:2] pins are programmed in the PAR to serve as the transfer modifier outputs or as three parallel port bits. These outputs provide supplemental information for each transfer type; see Table 17-6 through Table 17-10.

When the MCF5307 is the bus master and TM[2:0] are enabled, these signals are driven as outputs only. If an external device is bus master and TM[2:0] are enabled, these pins are three-stated by the MCF5307 and can be driven by the external master.

Table 17-6. TM[2:0] Encodings for TT = 00 (Normal Access)

TM[2:0]	Transfer Modifier
000	Cache push access
001	User data access
010	User code access
011–100	Reserved
101	Supervisor data access

Table 17-6. TM[2:0] Encodings for TT = 00 (Normal Access) (Continued)

TM[2:0]	Transfer Modifier
110	Supervisor code access
111	Reserved

As shown in Table 17-7, if the DMA is bus master (TT = 01), TM[2:0] indicate the type of DMA access and provide the DMA acknowledgement information for channels 0 and 1.

NOTE:

When TT= 01, the TM0 encoding is independent from TM[2:1] encoding.

Table 17-7. TM0 Encoding for DMA as Master (TT = 01)

TM0	Transfer Modifier Encoding
0	Single-address access negated
1	Single-address access

Table 17-8. TM[2:1] Encoding for DMA as Master (TT = 01)

TM[2:1]	Transfer Modifier Encoding
00	DMA acknowledges negated
01	DMA acknowledge, channel 0
10	DMA acknowledge, channel 1
11	Reserved

Table 17-9 shows TM[2:0] encodings for emulator mode accesses.

Table 17-9. TM[2:0] Encodings for TT = 10 (Emulator Access)

TM[2:0]	Transfer Modifier
000–100	Reserved
101	Emulator mode data access
110	Emulator mode code access
111	Reserved

The TM signals indicate user or data transfer types during emulation transfers, while for interrupt acknowledge transfers, the TM signals carry the interrupt level being acknowledged; see Table 17-10.

Table 17-10. TM[2:0] Encodings for TT = 11 (Interrupt Level)

TM[2:0]	Transfer Modifier
000	CPU Space
001	Interrupt level 1 acknowledge

Table 17-10. TM[2:0] Encodings for TT = 11 (Interrupt Level) (Continued)

TM[2:0]	Transfer Modifier
010	Interrupt level 2 acknowledge
011	Interrupt level 3 acknowledge
100	Interrupt level 4 acknowledge
101	Interrupt level 5 acknowledge
110	Interrupt level 6 acknowledge
111	Interrupt level 7 acknowledge

17.3 Interrupt Control Signals

The interrupt control signals supply the external interrupt level to the MCF5307 device.

17.3.1 Interrupt Request ($\overline{IRQ1}$/$\overline{IRQ2}$, $\overline{IRQ3}$/$\overline{IRQ6}$, $\overline{IRQ5}$/$\overline{IRQ4}$, and $\overline{IRQ7}$)

The $\overline{IRQ1}$, $\overline{IRQ3}$, $\overline{IRQ5}$, and $\overline{IRQ7}$ signals are the default interrupt request signals ($\overline{IRQn}$). However, by setting the appropriate bit in the interrupt port assignment register (IRQPAR), $\overline{IRQ1}$, $\overline{IRQ3}$, and $\overline{IRQ5}$ can be changed to function as $\overline{IRQ2}$, $\overline{IRQ6}$, and $\overline{IRQ4}$, respectively. See Section 9.2.4, “Interrupt Port Assignment Register (IRQPAR).”

17.4 Bus Arbitration Signals

The bus arbitration signals provide the external bus arbitration control for the MCF5307.

17.4.1 Bus Request ($\overline{BR}$)

The $\overline{BR}$ output indicates to an external arbiter that the processor is requesting to be bus master for one or more bus cycles. $\overline{BR}$ is negated when the MCF5307 begins an access to the external bus with no other internal accesses pending. $\overline{BR}$ remains negated until another internal request occurs.

17.4.2 Bus Grant ($\overline{BG}$)

An external arbiter asserts the $\overline{BG}$ input to indicate that the MCF5307 can take control of the bus on the next rising edge of BCLKO. When the arbiter negates $\overline{BG}$, the MCF5307 will release the bus as soon as the current transfer completes. The external arbiter must not grant the bus to any other master until both $\overline{BD}$ and $\overline{BG}$ are negated.

17.4.3 Bus Driven ($\overline{BD}$)

The MCF5307 asserts $\overline{BD}$ to indicate that it is the current master and is driving the bus. The MCF5307 behaves as follows:

- If the MCF5307 is the bus master but is not using the bus, $\overline{BD}$ is asserted.
- If the MCF5307 loses mastership during a transfer, it completes the last transfer of the access, negates $\overline{BD}$, and three-states all bus signals on the rising edge of BCLKO.
- If the MCF5307 loses bus mastership during an idle clock cycle, it three-states all bus signals on the rising edge of BCLKO.
- $\overline{BD}$ cannot be negated unless $\overline{BG}$ is negated.

17.5 Clock and Reset Signals

The clock and reset signals configure the MCF5307 and provide interface signals to the external system.

17.5.1 Reset In ($\overline{RSTI}$)

Asserting $\overline{RSTI}$ causes the MCF5307 to enter reset exception processing. When $\overline{RSTI}$ is recognized, $\overline{BR}$ and $\overline{BD}$ are negated and the address bus, data bus, TT, SIZ, R/$\overline{W}$, $\overline{AS}$, and $\overline{TS}$ are three-stated. $\overline{RSTO}$ is asserted automatically when $\overline{RSTI}$ is asserted.

17.5.2 Clock Input (CLKIN)

CLKIN is the MCF5307 input clock frequency to the on-board phase-locked-loop (PLL) clock generator. CLKIN is used to internally clock or sequence the MCF5307 internal bus interface at a selected multiple of the input frequency used for internal module logic.

17.5.3 Bus Clock Output (BCLKO)

The internal PLL generates BCLKO and can be programmed to be 1/2, 1/3, or 1/4 of the processor clock frequency. BCLKO should be used as the bus timing reference.

17.5.4 Reset Out ($\overline{RSTO}$)

After $\overline{RSTI}$ is asserted, the PLL temporarily loses its lock, during which time $\overline{RSTO}$ is asserted. When the PLL regains its lock, $\overline{RSTO}$ negates again. This signal can be used to reset external devices.

17.5.5 Data/Configuration Pins (D[7:0])

This section describes data pins, D[7:0], that are read at reset for configuration. Table 17-11 shows pin assignments.

Table 17-11. Data Pin Configuration

Pin	Function
D7	Auto-acknowledge configuration (AA_CONFIG)
D[6:5]	Port size configuration (PS_CONFIG[1:0])
D4	Address configuration (ADDR_CONFIG/D4)
D[3:2]	Frequency Control PLL (FREQ[1:0])
D[1:0]	Divide Control (DIVIDE[1:0])

17.5.5.1 D[7:5Boot Chip-Select ($\overline{CS0}$) Configuration

D[7:5] determine defaults for the global chip select ($\overline{CS0}$), the only chip select valid at reset. These signals correspond to bits in chip-select configuration register 0 (CSCR0).

17.5.5.2 D7—Auto Acknowledge Configuration (AA_CONFIG)

At reset, the enabling and disabling of auto acknowledge for boot $\overline{CS0}$ is determined by the logic level driven on D7 at the rising edge of $\overline{RSTI}$. AA_CONFIG is multiplexed with D7 and sampled only at reset. The D7 logic level is reflected as the reset value of CSCR[AA]. Table 17-12 shows how the D7 logic level corresponds to the auto acknowledge timing for $\overline{CS0}$ at reset. Note that auto acknowledge can be disabled by driving a logic 0 on D7 at reset.

Table 17-12. D7 Selection of $\overline{CS0}$ Automatic Acknowledge

D7 (CSCR0[AA])	Boot $\overline{CS0}$ AA
0	Disabled
1	Enabled with 15 wait states

17.5.5.3 D[6:5]—Port Size Configuration (PS_CONFIG[1:0])

The default port size value of the boot $\overline{CS0}$ is determined by the logic levels driven on D[6:5] at the rising edge of $\overline{RSTI}$, which are reflected as the reset value of CSCR[PS]. Table 17-13 shows how the logic levels of D[6:5] correspond to the $\overline{CS0}$ port size at reset.

Table 17-13. D6 and D5 Selection of $\overline{CS0}$ Port Size

D[6:5] (CSCR0[PS])	Boot $\overline{CS0}$ Port Size
00	32-bit port
01	8-bit port
1x	16-bit port

17.5.6 D4—Address Configuration (ADDR_CONFIG)

The address configuration signal (ADDR_CONFIG) programs the PAR of the parallel I/O port to be either parallel I/O or to be the upper address bus bits along with various attribute and control signals at reset to give the user the option to access a broader addressing range

of memory if desired. ADDR_CONFIG is multiplexed with D4 and its configuration is sampled at reset as shown in Table 17-14.

Table 17-14. D4/ADDR_CONFIG, Address Pin Assignment

D4/ADDR_CONFIG	PAR Configuration at Reset
0	PP[15:0], defaulted to inputs upon reset
1	A[31:24]/TIP/DREQ[1:0]/TM[2:0]/TT[1:0]

17.5.7 D[3:2]—Frequency Control PLL (FREQ[1:0])

The frequency control PLL input bus (FREQ[1:0]) indicates the CLKIN frequency range. These signals are multiplexed with D[3:2] and are sampled during the assertion of RESET. These signals indicate the operating frequency range to the PLL, as shown in Table 17-15. Note that these signals do not affect the PLL frequency but are required to set up the analog PLL.

Table 17-15. CLKIN Frequency

FREQ[1:0]/D[3:2]	CLKIN Frequency (MHz)
00	16.6–27.999
01	28–38.999
10	39–45
11	Reserved

17.5.8 D[1:0]—Divide Control PCLK to BCLKO (DIVIDE[1:0])

This 2-bit input bus indicates the BCLKO/PSTCLK ratio. These signals are sampled during the assertion of RESET and indicate the ratios shown in Table 17-16.

Table 17-16. BCLKO/PSTCLK Divide Ratios

DIVIDE[1:0]/D[1:0]	Ratio of BCLKO/PSTCLK
00	1/4
01	Reserved
10	1/2
11	1/3

17.6 Chip-Select Module Signals

The MCF5307 device provides eight programmable chip-select signals that can directly interface with SRAM, EPROM, EEPROM, and peripherals. These signals are asserted and negated on the falling edge of the clock.

17.6.1 Chip-Select ($\overline{CS}$[7:0])

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; $\overline{CS0}$ is the only chip select initialized out of reset. $\overline{CS0}$ 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. Port size and termination (internal vs. external) for boot $\overline{CS0}$ are configured by the levels on D[7:5] on the rising edge of $\overline{RSTI}$, as described in Section 17.5.5.1, “D[7:5Boot Chip-Select (CS0) Configuration.”

The chip-select implementation is described in Chapter 10, “Chip-Select Module.”

17.6.2 Byte Enables/Byte Write Enables ($\overline{BE}$[3:0]/$\overline{BWE}$[3:0])

The four byte enables are multiplexed with the MCF5307 byte-write-enable signals. Each pin can be individually programmed through the chip-select control registers (CSCRs). For each chip select, assertion of byte enables for reads and byte-write enables for write cycles can be programmed. Alternatively, users can program byte-write enables to assert on writes and no byte enable assertion for read transfers.

17.6.3 Output Enable ($\overline{OE}$)

The output enable ($\overline{OE}$) signal is sent to the interfacing memory and/or peripheral to enable a read transfer. $\overline{OE}$ is asserted only when a chip select matches the current address decode.

17.7 DRAM Controller Signals

The DRAM signals in the following sections interface to external DRAM. DRAM with widths of 8, 16, and 32 bits are supported and can access as much as 512 Mbytes of DRAM.

17.7.1 Row Address Strobes ($\overline{RAS}$[1:0])

The row address strobes ($\overline{RAS}$[1:0]) interface to $\overline{RAS}$ inputs on industry-standard ADRAMs. When SDRAMs are used, these signals interface to the chip-select lines of the SDRAMs within a memory block. Thus, there is one $\overline{RAS}$ line for each memory block.

17.7.2 Column Address Strobes ($\overline{CAS}$[3:0])

The column address strobes ($\overline{CAS}$[3:0]) interface to $\overline{CAS}$ inputs on industry-standard DRAMs. These provide $\overline{CAS}$ for a given ADRAM block. When SDRAMs are used, $\overline{CAS}$ signals control the byte enables for standard SDRAMs (referred to as DQM*x*). $\overline{CAS3}$ accesses the LSB and $\overline{CAS0}$ accesses the MSB of data.

17.7.3 DRAM Write ($\overline{\text{DRAMW}}$)

The DRAM write signal ($\overline{\text{DRAMW}}$) is asserted to signify that a DRAM write cycle is underway. A read bus cycle is indicated by the negation of $\overline{\text{DRAMW}}$.

17.7.4 Synchronous DRAM Column Address Strobe ($\overline{\text{SCAS}}$)

The synchronous DRAM column address strobe ($\overline{\text{SCAS}}$) is registered during synchronous mode to route directly to the $\overline{\text{SCAS}}$ signal of SDRAMs.

17.7.5 Synchronous DRAM Row Address Strobe ($\overline{\text{SRAS}}$)

The synchronous DRAM row address strobe output ($\overline{\text{SRAS}}$) is registered during synchronous mode to route directly to the $\overline{\text{SRAS}}$ signal of external SDRAMs.

17.7.6 Synchronous DRAM Clock Enable (SCKE)

The synchronous DRAM clock enable output (SCKE) is registered during synchronous mode to route directly to the SCKE signal of external SDRAMs. This signal provides the clock enable to the SDRAM.

17.7.7 Synchronous Edge Select (EDGESEL)

The synchronous edge select input (EDGESEL) helps select additional output hold times for signals that interface to external SDRAMs. It provides the following three modes of operation for SDRAM control signals:

- When EDGESEL is tied high, SDRAM control signals change on the rising edge of BCLKO.
- When EDGESEL is tied low, SDRAM control signals change on the falling edge of BCLKO.
- When EDGESEL is tied to the external clock (normally buffered BCLKO), which drives the SDRAM and other devices, SDRAM signals are generated within the MCF5307 make a transition on the rising edge of the SDRAM clock. See Figure 11-14 on page 11-19. This loop-back configuration provides additional output hold time for MCF5307 interface signals provided to the SDRAM. In this case, the SDRAM clock operates at the BCLKO frequency, with a possible slight phase delay.

17.8 DMA Controller Module Signals

The DMA controller module uses the signals in the following subsections to provide external request for either a source or destination.

17.8.1 DMA Request ($\overline{\text{DREQ}}$[1:0]/PP[6:5])

The DMA request pins ($\overline{\text{DREQ}}$[1:0]/PP[6:5]) can serve as the DMA request inputs or as two bits of the parallel port, as determined by individually programmable bits in the PAR.

These inputs are asserted by a peripheral device to request an operand transfer between that peripheral and memory by either channel 0 or 1 of the on-chip DMA.

Note that DMA acknowledge indication is displayed on TM[2:0], during DMA transfers of channel 0 and 1.

17.9 Serial Module Signals

The signals in the following sections are used to transfer serial data between the two UART modules and external peripherals.

17.9.1 Transmitter Serial Data Output (TxD)

TxD is held high (mark condition) when the transmitter is disabled, idle, or operating in the local loop-back mode. Data is shifted out least-significant bit (lsb) first on TxD on the falling edge of the clock source.

17.9.2 Receiver Serial Data Input (RxD)

Data received on RxD is sampled on the rising edge of the clock source, with the lsb received first.

17.9.3 Clear to Send ($\overline{\text{CTS}}$)

This input can generate an interrupt on a change of state.

17.9.4 Request to Send ($\overline{\text{RTS}}$)

This output can be programmed to be negated or asserted automatically by either the receiver or the transmitter. When connected to a transmitter's $\overline{\text{CTS}}$, $\overline{\text{RTS}}$ can control serial data flow.

17.10 Timer Module Signals

The signals in the following sections are external interfaces to the two general-purpose MCF5307 timers. These 16-bit timers can capture timer values, trigger external events or internal interrupts, or count external events.

17.10.1 Timer Inputs (TIN[1:0])

TIN[1:0] can be programmed as clocks that cause events in the counter and prescalers. They can also cause captures on the rising edge, falling edge, or both edges.

17.10.2 Timer Outputs (TOUT1, TOUT0)

The programmable timer outputs (TOUT1 and TOUT0) pulse or toggle on various timer events.

17.11 Parallel I/O Port (PP[15:0])

This 16-bit bus is dedicated for general-purpose I/O. The parallel port is multiplexed with the A[31:24], TT[1:0], TM[2:0], $\overline{\text{TIP}}$, and $\overline{\text{DREQ}}$[1:0]. These 16 bits are programmed for functionality with the PAR in the SIM.

The system designer controls the reset value of this register by driving D4 with a 1 or 0 on the rising edge of $\overline{\text{RSTI}}$ (reset input to MCF5307 device). At reset, the system is configured as PP[15:0] if D4 is 0; otherwise alternate pin functions selected by PAR = 1 are used. Motorola recommends that D4 be driven during reset to a logic level.

17.12 I^2C Module Signals

The I^2C module acts as a two-wire, bidirectional serial interface between the MCF5307 and peripherals with an I^2C interface (such as LED controller, A-to-D converter, or D-to-A converter). Devices connected to the I^2C must have open-drain or open-collector outputs.

17.12.1 I^2C Serial Clock (SCL)

The bidirectional, open-drain I^2C serial clock signal (SCL) is the clock signal for I^2C module operation. The I^2C module controls this signal when the bus is in master mode; all I^2C devices drive this signal to synchronize I^2C timing.

17.12.2 I^2C Serial Data (SDA)

The bidirectional, open-drain I^2C serial data signal (SDA) is the data input/output for the serial I^2C interface.

17.13 Debug and Test Signals

The signals in this section interface with external I/O to provide processor status signals.

17.13.1 Test Mode (MTMOD[3:0])

The test mode signals choose between multiplexed debug module and JTAG signals. If MTMOD0 is low, the part is in normal and background debug mode (BDM); if it is high, it is in normal and JTAG mode. All other MTMOD values are reserved; MTMOD[3:1] should be tied to ground and MTMOD[3:0] should not be changed while $\overline{\text{RSTI}}$ is negated.

17.13.2 High Impedance ($\overline{\text{HIZ}}$)

The assertion of $\overline{\text{HIZ}}$ forces all output drivers to high-impedance state. The timing on $\overline{\text{HIZ}}$ is independent of the clock. Note that $\overline{\text{HIZ}}$ does not override the JTAG operation; TDO/DSO can be forced to high impedance by asserting $\overline{\text{TRST}}$.

17.13.3 Processor Clock Output (PSTCLK)

The internal PLL generates this output signal, and is the processor clock output that is used as the timing reference for the debug bus timing (DDATA[3:0] and PST[3:0]). PSTCLK is at the same frequency as the core processor and cache memory. The frequency is 2x the CLKIN.

17.13.4 Debug Data (DDATA[3:0])

The debug data signals (DDATA[3:0]) display captured processor data and breakpoint status. See Chapter 5, “Debug Support,” for additional information on this bus.

17.13.5 Processor Status (PST[3:0])

The processor status pins indicate the MCF5307 processor status. During debug mode, the timing is synchronous with the processor clock (PSTCLK) and the status is not related to the current bus transfer. Table 2-11 shows the encodings of these signals.

Table 17-17. Processor Status Signal Encodings

PST[3:0]		Definition
Hex	Binary	
0x0	0000	Continue execution
0x1	0001	Begin execution of an instruction
0x2	0010	Reserved
0x3	0011	Entry into user-mode
0x4	0100	Begin execution of PULSE and WDDATA instructions
0x5	0101	Begin execution of taken branch or Synch_PC[1]
0x6	0110	Reserved
0x7	0111	Begin execution of RTE instruction
0x8	1000	Begin 1-byte data transfer on DDATA
0x9	1001	Begin 2-byte data transfer on DDATA
0xA	1010	Begin 3-byte data transfer on DDATA
0xB	1011	Begin 4-byte data transfer on DDATA
0xC	1100	Exception processing[2]
0xD	1101	Emulator mode entry exception processing[2]
0xE	1110	Processor is stopped, waiting for interrupt[2]
0xF	1111	Processor is halted[2]

1 Rev. B enhancement.

2 These encodings are asserted for multiple cycles.

17.14 Debug Module/JTAG Signals

The MCF5307 complies with the IEEE 1149.1a JTAG testing standard. JTAG test pins are multiplexed with background debug pins. Except for TCK, these signals are selected by the value of MTMOD0. If MTMOD0 is high, JTAG signals are chosen; if it is low, debug module signals are chosen. MTMOD0 should be changed only while $\overline{\text{RSTI}}$ is asserted.

17.14.1 Test Reset/Development Serial Clock ($\overline{\text{TRST}}$/DSCLK)

If MTMOD0 is high, $\overline{\text{TRST}}$ 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 MCF5307 functionality.

Although $\overline{\text{TRST}}$ is asynchronous, Motorola 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.

If MTMOD0 is low, 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. See Chapter 5, “Debug Support.”

17.14.2 Test Mode Select/Breakpoint (TMS/$\overline{\text{BKPT}}$)

If MTMOD0 is high, 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}.

If MTMOD0 is low, $\overline{\text{BKPT}}$ is selected. $\overline{\text{BKPT}}$ signals a hardware breakpoint to the processor in debug mode. See Chapter 5, “Debug Support.”

17.14.3 Test Data Input/Development Serial Input (TDI/DSI)

If MTMOD0 is high, 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}.

If MTMOD0 is low, DSI is selected. DSI provides the single-bit communication for debug module commands. See Chapter 5, “Debug Support.”

17.14.4 Test Data Output/Development Serial Output (TDO/DSO)

If MTMOD0 is high, 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.

If MTMOD0 is low, DSO is selected. DSO provides single-bit communication for debug module responses. See Chapter 5, “Debug Support.”

17.14.5 Test Clock (TCK)

TCK is the dedicated JTAG test logic clock independent of the MCF5307 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.

Chapter 18
Bus Operation

This chapter describes data-transfer operations, error conditions, bus arbitration, and reset operations. It describes transfers initiated by the MCF5307 and by an external bus master, and includes detailed timing diagrams showing the interaction of signals in supported bus operations. Chapter 11, "Synchronous/Asynchronous DRAM Controller Module," describes DRAM cycles.

18.1 Features

The following list summarizes bus operation features:

- Up to 32 bits of address and data
- 8-, 16-, and 32-bit port sizes
- Byte, word, longword, and line size transfers
- Bus arbitration for external devices
- Burst and burst-inhibited transfer support
- Internal termination for core and DMA bus cycles
- External termination of bus cycles controlled by an external bus master

Note that, throughout this manual, an overbar indicates an active-low signal.

18.2 Bus and Control Signals

Table 18-1 summarizes MCF5307 bus signals described in Chapter 17, "Signal Descriptions."

Table 18-1. ColdFire Bus Signal Summary

Signal Name	Description	MCF5307 Master	External Master	Edge
$\overline{AS}$	Address strobe	O	I	Falling
A[31:0]	Address bus	O	I	Rising
$\overline{BE}$/$\overline{BWE}$ [1]	Byte enable/Byte write enable	O	O	Falling
$\overline{CS}$[7:0] [1]	Chip selects	O	O	Falling
D[31:0]	Data bus	I/O	I/O	Rising

Table 18-1. ColdFire Bus Signal Summary (Continued)

Signal Name	Description	MCF5307 Master	External Master	Edge
$\overline{IRQ}$[7,5,3,1]	Interrupt request	I	I	Rising
$\overline{OE}$ [1]	Output enable	O	I	Falling
R/$\overline{W}$	Read/write	O	I	Rising
SIZ[1:0]	Transfer size	O	I	Rising
$\overline{TA}$	Transfer acknowledge	I	O	Rising
$\overline{TIP}$	Transfer in progress	O	Three-state	Rising
TM[2:0]	Transfer modifier	O	Three-state	Rising
$\overline{TS}$	Transfer start	O	I	Rising
TT[1:0]	Transfer type	O	Three-state	Rising

1 These signals change after the falling edge. In Chapter 20, “Electrical Specifications,” these signals are specified off the rising edge because CLKIN is squared up internally.

18.3 Bus Characteristics

The MCF5307 uses an input clock signal (CLKIN) to generate its internal clock. BCLKO is the bus clock rate, where all bus operations are synchronous to the rising edge of BCLKO. Some of the bus control signals ($\overline{BE}$/$\overline{BWE}$, $\overline{OE}$, $\overline{CSx}$, and $\overline{AS}$) are synchronous to the falling edge, shown in Figure 18-1. Bus characteristics may differ somewhat for interfacing with external DRAM.

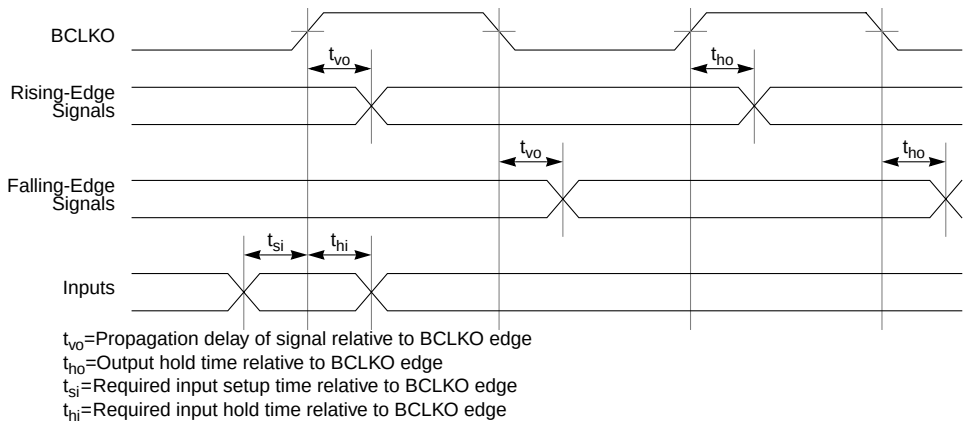


Figure 18-1. Signal Relationship to BCLKO for Non-DRAM Access

18.4 Data Transfer Operation

Data transfers between the MCF5307 and other devices involve the following signals:

- Address bus (A[31:0])
- Data bus (D[31:0])
- Control signals ($\overline{TS}$ and $\overline{TA}$)
- $\overline{AS}$, $\overline{CSx}$, $\overline{OE}$, $\overline{BE}/\overline{BWE}$
- Attribute signals (R/$\overline{W}$, SIZ, TT, TM, and $\overline{TIP}$)

The address bus, write data, $\overline{TS}$, and all attribute signals change on the rising edge of BCLKO. Read data is latched into the MCF5307 on the rising edge of BCLKO. $\overline{AS}$, $\overline{CSx}$, $\overline{OE}$, and $\overline{BE}/\overline{BWE}$ change on the falling edge.

The MCF5307 bus supports byte, word, and longword operand transfers and allows accesses to 8-, 16-, and 32-bit data ports. Transfer parameters such as 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 DRAM control registers (DACRs).

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.

Table 18-2 shows encoding for SIZ[1:0].

Table 18-2. Bus Cycle Size Encoding

SIZ[1:0]	Port Size
00	Longword
01	Byte
10	Word
11	Line

Figure 18-2 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] ($\overline{BE0}$). A longword transfer takes four transfers on D[31:24], starting with the MSB and going to the LSB.

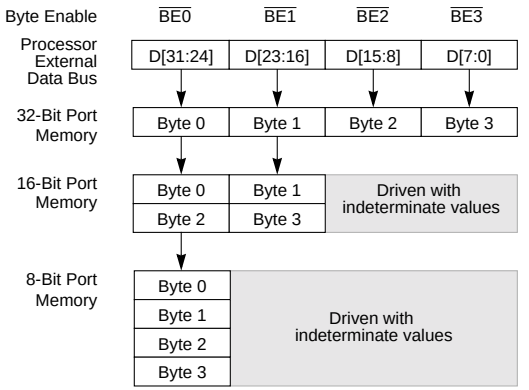


Figure 18-2. Connections for External Memory Port Sizes

The timing relationships between BCLKOchip select ($\overline{CS}$[7:0]), byte enable/byte write enables ($\overline{BE}$/$\overline{BWE}$[3:0]), and output enable ($\overline{OE}$) are similar to their relationships with address strobe ($\overline{AS}$) in that all transitions occur during the low phase of BCLKO. However, as shown in Figure 18-3, differences in on-chip signal routing and external loading may prevent signals from asserting simultaneously.

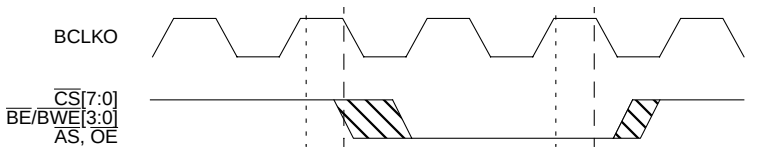


Figure 18-3. Chip-Select Module Output Timing Diagram

18.4.1 Bus Cycle Execution

When a bus cycle is initiated, the MCF5307 first compares its address with the base address and mask configurations programmed for chip selects 0–7 (CSCR0–CSCR7) and for DRAM blocks 0 and 1 address and control registers (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 CSCR or DACR, the MCF5307 runs an external burst-inhibited bus cycle with a default of external termination on a 32-bit port.
- If an address and attribute match in multiple CSCRs, the matching chip-select signals are driven; however, the MCF5307 runs an external burst-inhibited bus cycle with external termination on a 32-bit port.
- If an address and attribute match both DACRs or a DACR and a CSCR, the operation is undefined.

Table 18-3 shows the type of access as a function of match in the CSCRs and DACRs.

Table 18-3. 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 CSCRs
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 bus operations occur in three clocks, as follows:

1. During the first clock, the address, attributes, and $\overline{TS}$ are driven. $\overline{AS}$ is asserted at the falling edge of the clock to indicate that address and attributes are valid and stable.
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 clock edge.

   During a write, the MCF5307 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 DACRs and 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 18-6 and Figure 18-8 show the basic read and write operations.

18.4.2 Data Transfer Cycle States

The data transfer operation in the MCF5307 is controlled by an on-chip state machine. Each bus clock cycle is divided into two states. Even states occur when BCLKO is high and odd states occur when BCLKO is low. The state transition diagram for basic and fast-termination read and write cycles is shown in Figure 18-4.

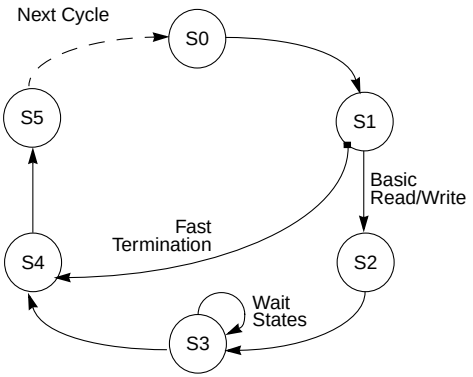


Figure 18-4. Data Transfer State Transition Diagram

Table 18-4 describes the states as they appear in subsequent timing diagrams. Note that the TT[1:0], TM[2:0], and $\overline{\text{TIP}}$ functions are chosen in the PAR, as described in Section 15.1.1, "Pin Assignment Register (PAR)."

Table 18-4. Bus Cycle States

State	Cycle	BCLKO	Description
S0	All	High	The read or write cycle is initiated. On the rising edge of BCLKO, the MCF5307 places a valid address on the address bus, asserts $\overline{\text{TIP}}$, and drives R/$\overline{\text{W}}$ high for a read and low for a write, if these signals are not already in the appropriate state. The MCF5307 asserts TT[1:0], TM[2:0], SIZ[1:0], and $\overline{\text{TS}}$ on the rising edge of BCLKO.
S1	All	Low	$\overline{\text{AS}}$ asserts on the falling edge of BCLKO, indicating that the address and attributes are stable. The appropriate $\overline{\text{CS}}$x, $\overline{\text{BE}}$/$\overline{\text{BWE}}$, and $\overline{\text{OE}}$ signals assert on the BCLKO falling edge.
	Fast termination		$\overline{\text{TA}}$ must be asserted during S1. Data is made available by the external device and is sampled on the rising edge of BCLKO with $\overline{\text{TA}}$ asserted.
S2	Read/write (skipped for fast termination)	High	$\overline{\text{TS}}$ is negated on the rising edge of BCLKO.
	Write		The data bus is driven out of high impedance as data is placed on the bus on the rising edge of BCLKO.
S3	Read/write (skipped for fast termination)	Low	The MCF5307 waits for $\overline{\text{TA}}$ assertion. If $\overline{\text{TA}}$ is not sampled as asserted before the rising edge of BCLKO at the end of the first clock cycle, the MCF5307 inserts wait states (full clock cycles) until $\overline{\text{TA}}$ is sampled as asserted.
	Read		Data is made available by the external device on the falling edge of BCLKO and is sampled on the rising edge of BCLKO with $\overline{\text{TA}}$ asserted.
S4	All	High	The external device should negate $\overline{\text{TA}}$.
	Read (including fast termination)		The external device can stop driving data after the rising edge of BCLKO. However, data could be driven up to S5.

Table 18-4. Bus Cycle States (Continued)

State	Cycle	BCLKO	Description
S5	S5	Low	$\overline{AS}$, $\overline{CS}$, $\overline{BE}$/$\overline{BWE}$, and $\overline{OE}$ are negated on the BCLKO falling edge. The MCF5307 stops driving address lines and R/$\overline{W}$ on the rising edge of BCLKO, terminating the read or write cycle. At the same time, the MCF5307 negates TT[1:0], TM[2:0], $\overline{TIP}$, and SIZ[1:0] on the rising edge of BCLKO. Note that the rising edge of BCLKO may be the start of S0 for the next access cycle; in this case, $\overline{TIP}$ remains asserted and R/$\overline{W}$ may not transition, depending on the nature of the back-to-back cycles.
	Read		The external device stops driving data between S4 and S5.
	Write		The data bus returns to high impedance on the rising edge of BCLKO. The rising edge of BCLKO may be the start of S0 for the next access.

NOTE:

An external device has at most two BCLKO cycles after the start of S4 to three-state the data bus after data is sampled in S3. This applies to basic read cycles, fast-termination cycles, and the last transfer of a burst.

18.4.3 Read Cycle

During a read cycle, the MCF5307 receives data from memory or from a peripheral device. Figure 18-5 is a read cycle flowchart.

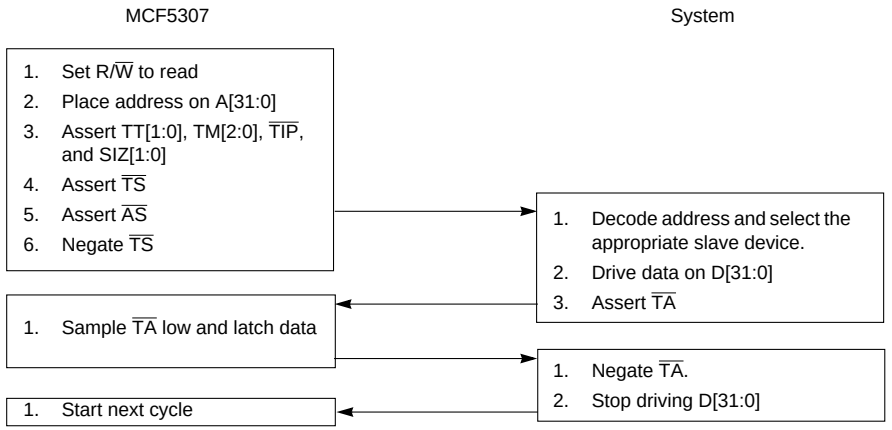


Figure 18-5. Read Cycle Flowchart

The read cycle timing diagram is shown in Figure 18-6.

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.

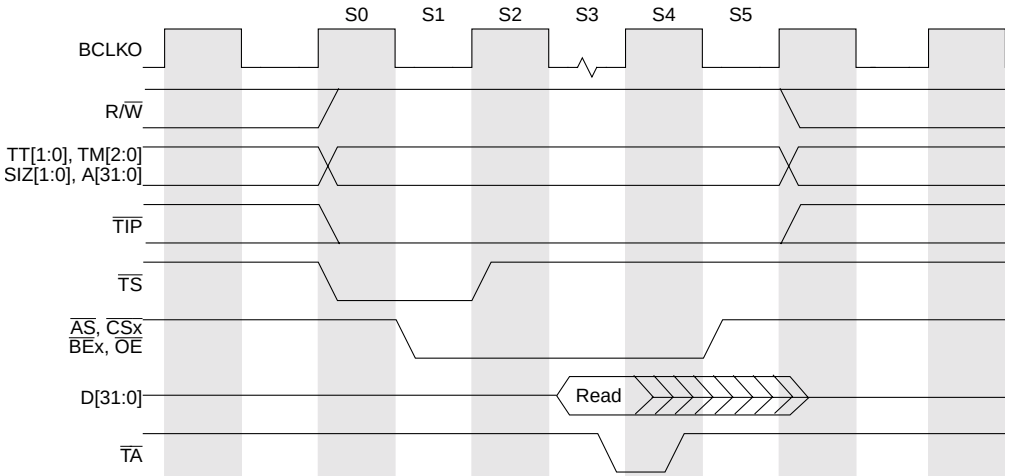


Figure 18-6. Basic Read Bus Cycle

Note the following characteristics of a basic read:

- In S3, data is made available by the external device on the falling edge of BCLKO and is sampled on the rising edge of BCLKO with TA asserted.
- In S4, the external device can stop driving data after the rising edge of BCLKO. 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 18-4.

18.4.4 Write Cycle

During a write cycle, the MCF5307 sends data to memory or to a peripheral device. The write cycle flowchart is shown in Figure 18-7.

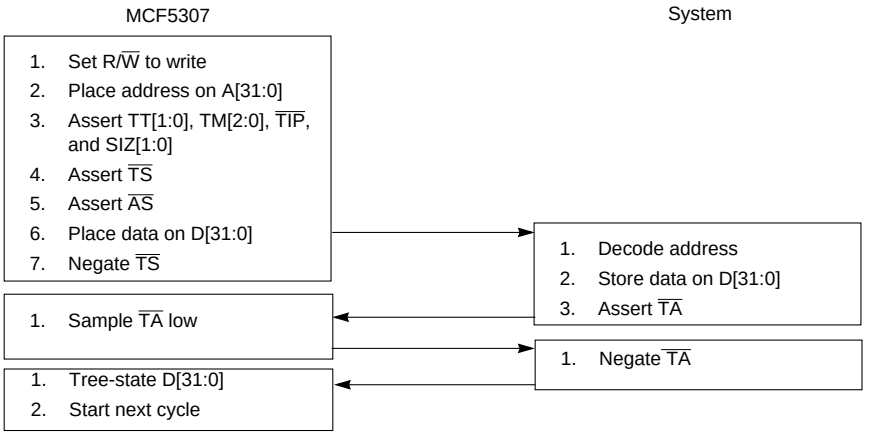


Figure 18-7. Write Cycle Flowchart

The write cycle timing diagram is shown in Figure 18-8.

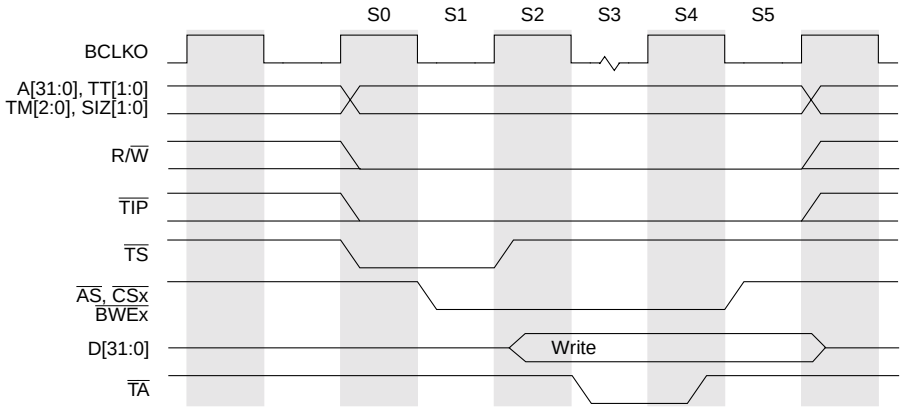


Figure 18-8. Basic Write Bus Cycle

Table 18-4 describes the six states of a basic write cycle.

18.4.5 Fast-Termination Cycles

Two clock-cycle transfers are supported on the MCF5307 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 AS is asserted. Because this is atypical, it is not referred to as the zero-wait-state case but is called the fast-termination case. A fast-termination cycle is one in which an external device or memory asserts TA as soon as TS is detected. This means that the MCF5307 samples TA on the rising edge of the second cycle of the bus transfer. Figure 18-9 shows a read cycle with fast termination. Note that fast termination cannot be used with internal termination.

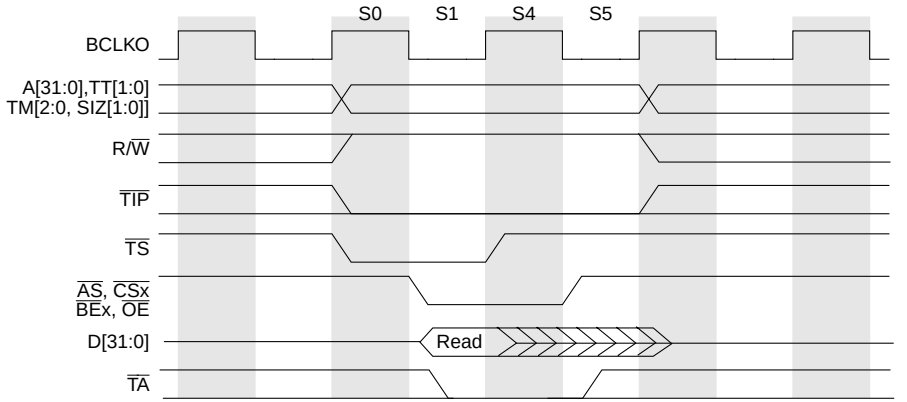


Figure 18-9. Read Cycle with Fast Termination

Figure 18-10 shows a write cycle with fast termination.

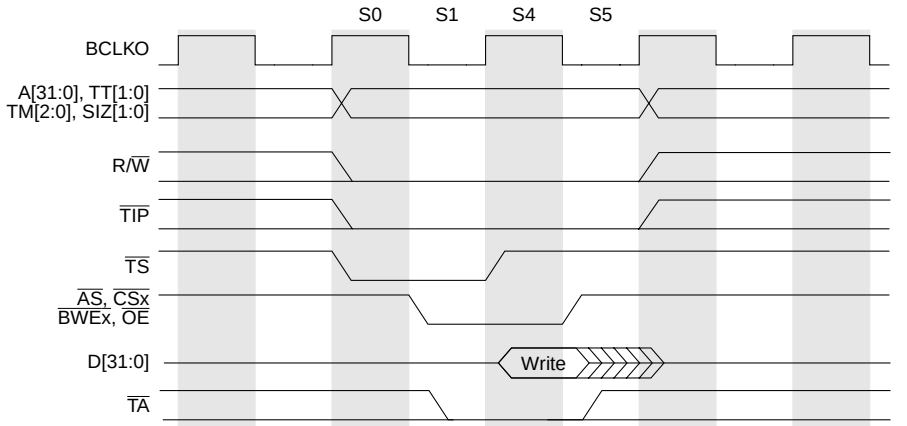


Figure 18-10. Write Cycle with Fast Termination

18.4.6 Back-to-Back Bus Cycles

The MCF5307 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 18-11 shows a read back-to-back with a write.

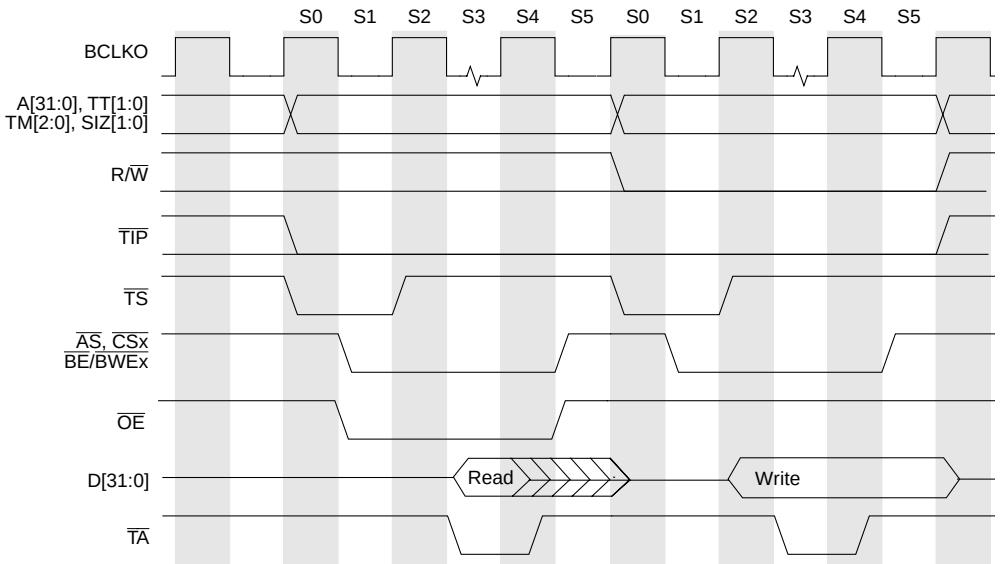


Figure 18-11. Back-to-Back Bus Cycles

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.

18.4.7 Burst Cycles

The MCF5307 can be programmed to initiate burst cycles if its transfer size exceeds the size of the port it is transferring to. For example, with bursting enabled, a word transfer to an 8-bit port would take a 2-byte burst cycle for which SIZ[1:0] = 10 throughout. A line transfer to a 32-bit port would take a 4-longword burst cycle, for which SIZ[1:0] = 11 throughout.

The MCF5307 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 MCF5307 writes a longword to an 8-bit port, SIZ[1:0] = 00 for the first byte transfer and does not change.

CSCRs are used to enable bursting for reads, writes, or both. MCF5307 memory space can be declared burst-inhibited for reads and writes by clearing the appropriate CSCR*x*[BSTR,BSTW]. A line access to a burst-inhibited region is broken into separate port-width accesses. Unlike a burst access, SIZ[1:0] = 11 only for the first port-width access; for the remaining accesses, SIZ[1:0] reflects the port width, with individual accesses separated by $\overline{AS}$ negations. The address changes if internal termination is used but does not change if external termination is used, as shown in Figure 18-12 and Figure 18-14.

18.4.7.1 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 18-5 shows allowable patterns for line accesses.

Table 18-5. 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

18.4.7.2 Line Read Bus Cycles

Figure 18-12 shows line 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 2 BCLKO cycles after $\overline{TA}$ is asserted. Note that $\overline{AS}$ and $\overline{CSx}$ are asserted throughout the burst transfer. This example shows the timing for external termination, which differs only from the internal termination example in Figure 18-13 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.

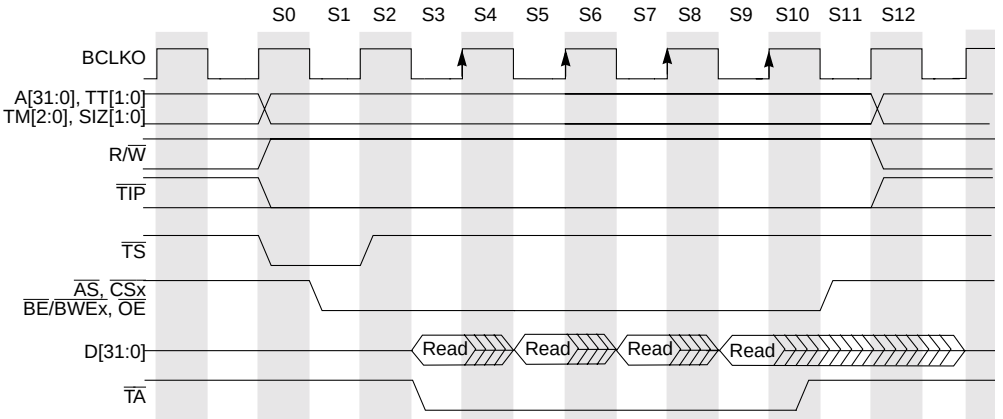


Figure 18-12. Line Read Burst (2-1-1-1), External Termination

Figure 18-13 shows timing when internal termination is used.

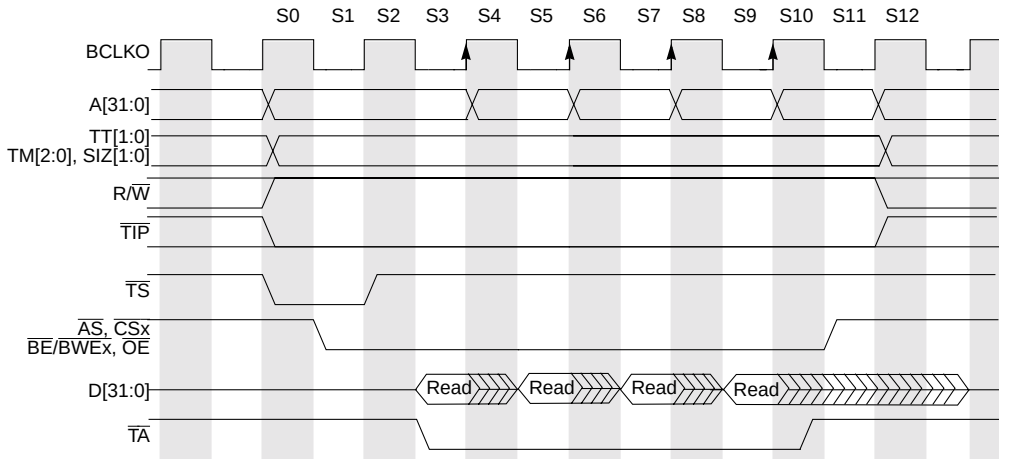


Figure 18-13. Line Read Burst (2-1-1-1), Internal Termination

Figure 18-14 shows a line access read with one wait state programmed in CSCR*x* 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.

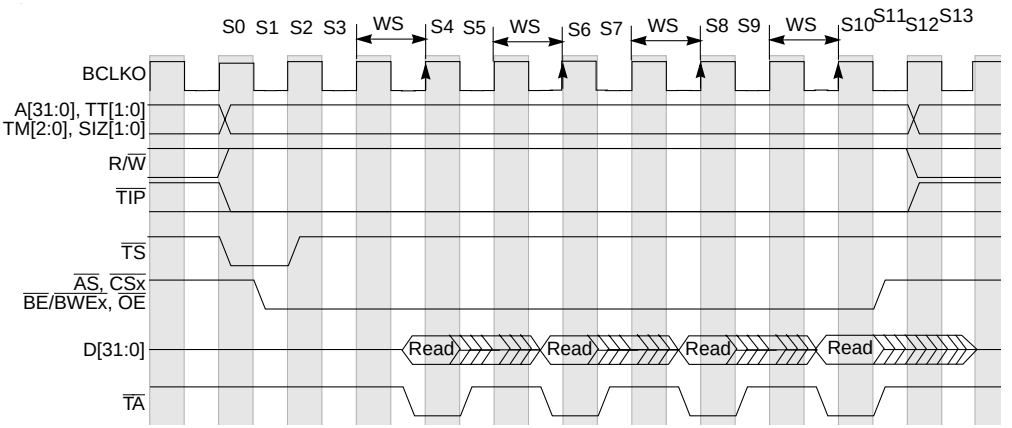


Figure 18-14. Line Read Burst (3-2-2-2), External Termination

Figure 18-15 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.

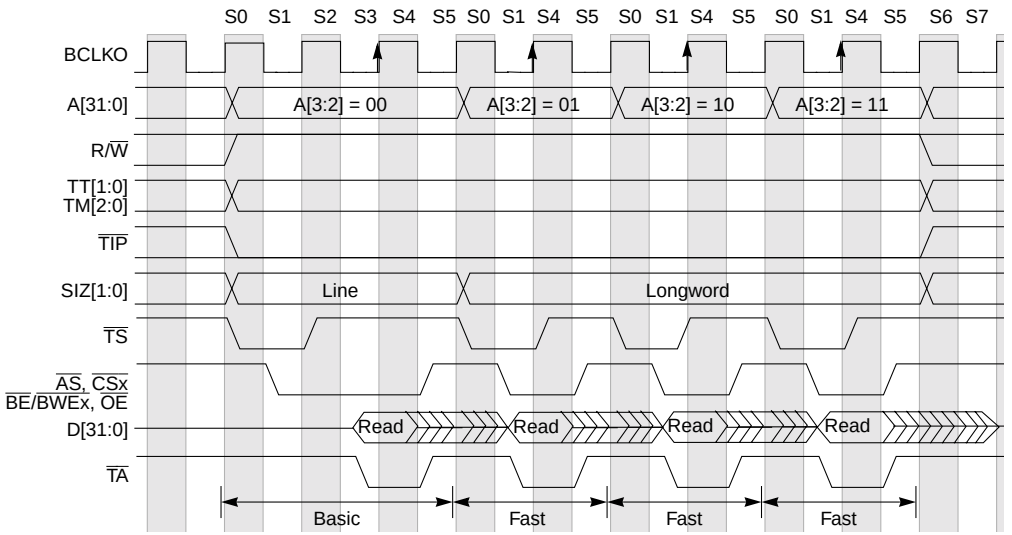


Figure 18-15. Line Read Burst-Inhibited, Fast, External Termination

18.4.7.3 Line Write Bus Cycles

Figure 18-16 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 18-12, AS and CS*x* remain asserted throughout the burst transfer. This example shows the behavior of the address lines for both internal and external termination. Note that with external termination, address lines, like SIZ, TT, and TM, hold the same value for the entire transfer.

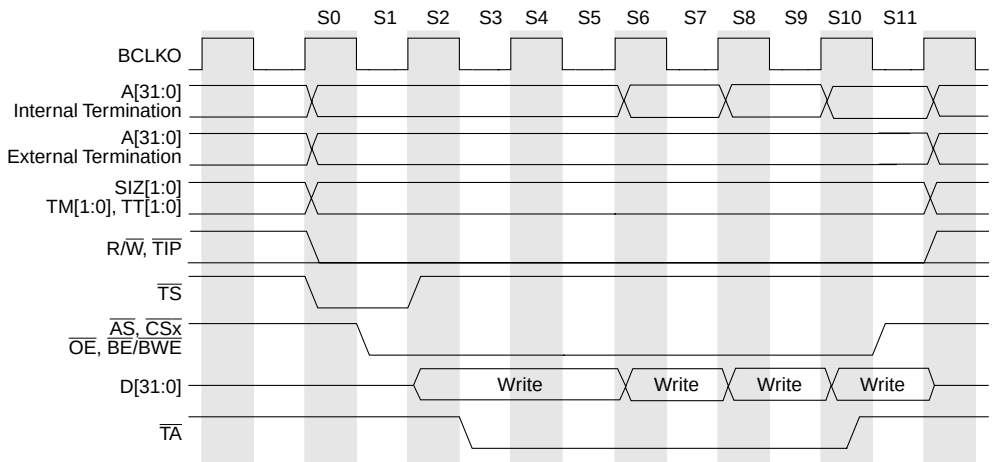


Figure 18-16. Line Write Burst (2-1-1-1), Internal/External Termination

Figure 18-17 shows a line burst write with one wait-state insertion.

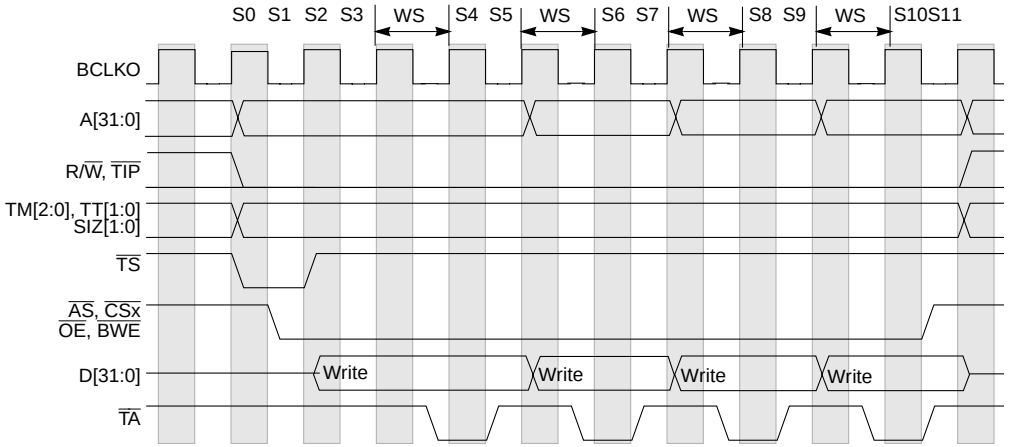


Figure 18-17. Line Write Burst (3-2-2-2) with One Wait State, Internal Termination

Figure 18-18 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.

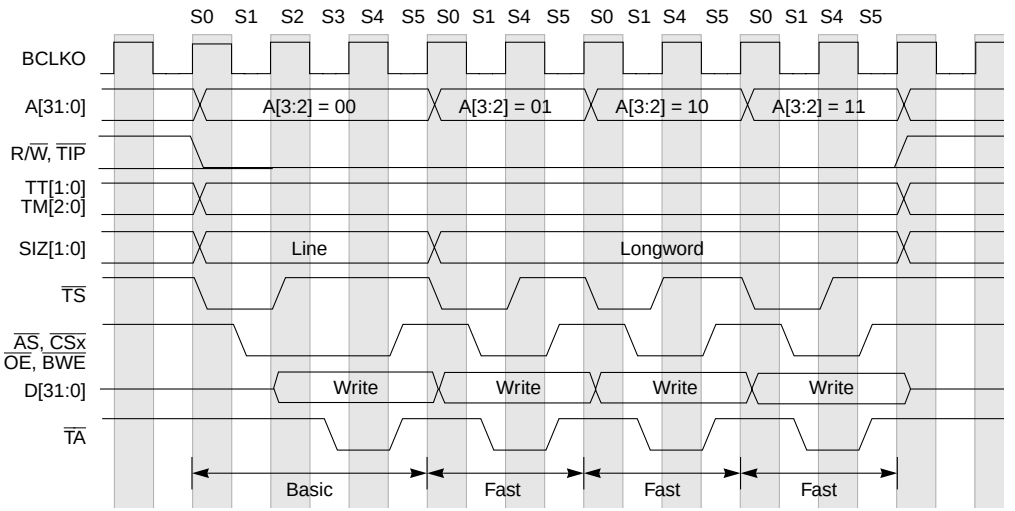


Figure 18-18. Line Write Burst-Inhibited, Internal Termination

18.4.7.4 Transfers Using Mixed Port Sizes

Figure 18-19 shows timing for a longword read from an 8-bit port using external termination. Figure 18-20 shows the same transfer with internal termination. For both, SIZ[1:0] change only at the start of a new transfer because this burst is implemented as one

transfer.

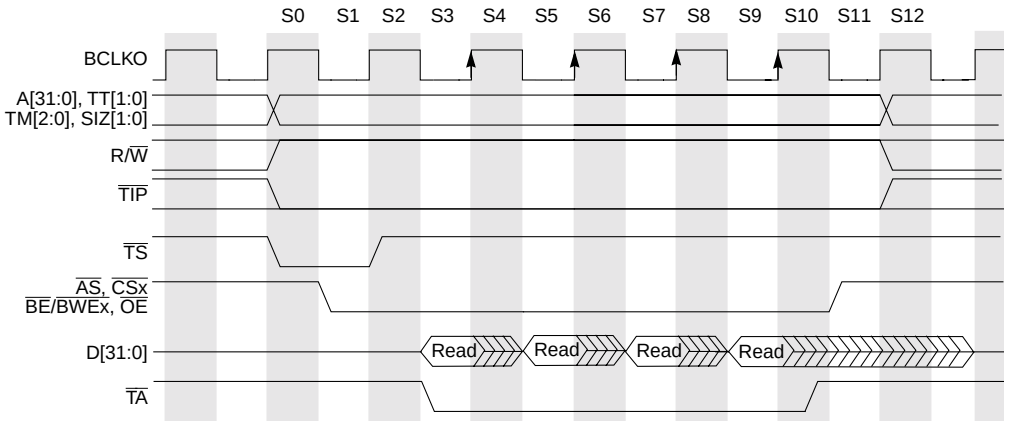


Figure 18-19. Longword Read from an 8-Bit Port, External Termination

Note that with external termination, address signals do not change. With internal termination, Figure 18-20, A[1:0] increment for the same longword transfer.

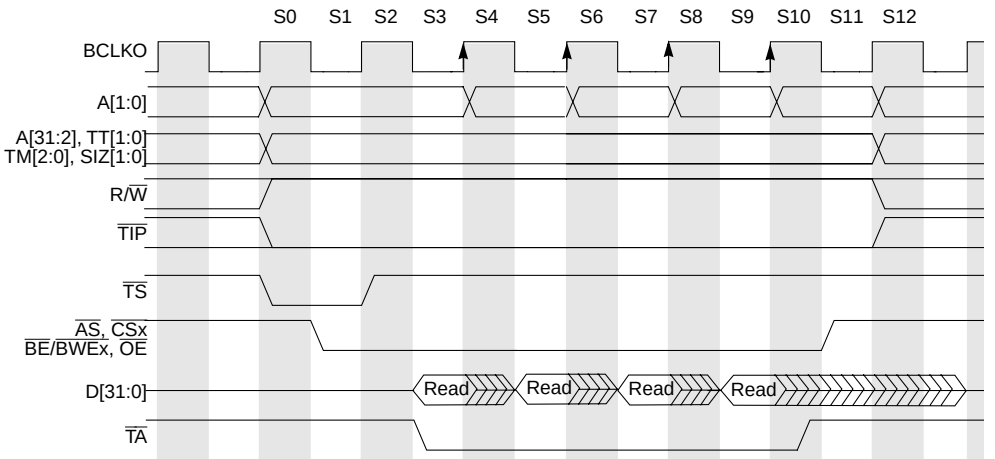


Figure 18-20. Longword Read from an 8-Bit Port, Internal Termination

18.5 Misaligned Operands

Because operands, unlike opcodes, can reside at any byte boundary, 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 MCF5307 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 MCF5307 converts misaligned, cache-inhibited operand accesses to multiple aligned accesses. Figure 18-21 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 MCF5307 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 18-21. 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 18-22 differs from the one in Figure 18-21 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 0	—	—	—	100

Figure 18-22. Example of a Misaligned Word Transfer (32-Bit Port)

NOTE:

External masters using internal MCF5307 chip selects and default memory control signals must initiate aligned transfers.

18.6 Bus Errors

The MCF5307 has no bus monitor. If the auto-acknowledge feature is not enabled for the address that generates the error, the bus cycle can be terminated by asserting $\overline{TA}$ or by using the software watchdog timer. If it is required that the MCF5307 handle a bus error differently, an interrupt handler can be invoked by asserting an interrupt to the core along with $\overline{TA}$ when the bus error occurs.

18.7 Interrupt Exceptions

A peripheral device uses the interrupt-request signals ($\overline{IRQx}$) to signal the core to take an interrupt exception when it needs the MCF5307 or is ready to send information to it. The interrupt transfers control to an appropriate routine.

The MCF5307 has the following two levels of interrupt masking:

- Interrupt mask registers in the SIM compare interrupt inputs with programmable interrupt mask levels. The SIM outputs only unmasked interrupts.
- The status register uses a 3-bit interrupt priority mask. The core recognizes only interrupt requests of higher priority than the value in the mask. See Section 2.2.2.1, “Status Register (SR).”

NOTE:

To mask a level 1–6 interrupt source, write a higher-level SR interrupt mask before setting IMR. Then restore the mask to its previous value. Do not mask a level 7 interrupt source.

The MCF5307 continuously samples and synchronizes external interrupt inputs. An interrupt request must be held for at least two consecutive BCLKO periods to be considered valid. To guarantee that the interrupt is recognized, the request level must be maintained until the MCF5307 acknowledges the interrupt with an interrupt-acknowledge cycle.

NOTE:

Interrupt levels 1–7 are level-sensitive. Level 7 is also edge-triggered. See Section 18.7.1, “Level 7 Interrupts.”

The MCF5307 takes an interrupt exception for a pending interrupt within one instruction boundary after processing any higher-priority pending exception. Thus, the MCF5307 executes at least one instruction in an interrupt exception handler before recognizing another interrupt request.

If autovector generation is used for internal interrupts (ICR*n*[AVEC] = 1), the interrupt acknowledge vector is generated internally and no interrupt acknowledge cycle is generated on the external bus.

If autovector generation is used for external interrupts, no interrupt acknowledge cycle is shown on the external bus ($\overline{AS}$ is not asserted) unless AVR[BLK] is 0. Consequently, the external interrupt must be cleared in the interrupt service routine. See Section 9.2.2, “Autovector Register (AVR).”

18.7.1 Level 7 Interrupts

Level 7 interrupts are nonmaskable and are handled differently than other interrupts. Level 7 interrupts are edge triggered by a transition from a lower priority request to the level 7 request. Interrupts at all other levels are level sensitive. Therefore, if $\overline{IRQ7}$ remains asserted, the MCF5307 recognizes only one level 7 interrupt because only one transition from a lower level request to a level 7evel 7 request occurred. For the processor to recognize two consecutive level 7 interrupts, one of the following must occur:

- The interrupt request on the interrupt control pins is raised to level 7 and stays there until an interrupt-acknowledge cycle begins. The level later drops but then returns to level 7, causing a second transition on the interrupt control lines.
- The interrupt request on the interrupt control pins is raised to level 7 and stays there. If the level 7 interrupt routine lowers the mask level, a second level 7 interrupt is recognized without a transition of the interrupt control pins. After the level 7 routine completes, the MCF5307 compares the mask level to the request level on the $\overline{\text{IRQ}}x$ signals. Because the mask level is lower than the requested level, the interrupt mask is set back to level 7. To ensure it is recognized, the level 7 request on $\overline{\text{IRQ7}}$ must be held until the second interrupt-acknowledge bus cycle begins.

18.7.2 Interrupt-Acknowledge Cycle

When the MCF5307 processes an interrupt exception, it performs an interrupt-acknowledge bus cycle to obtain the vector number that contains the starting location of the interrupt exception handler. The interrupt-acknowledge bus cycle is a read transfer that differs from normal read cycles in the following respects:

- TT[1:0] = 0x3 to indicate a CPU space or acknowledge bus cycle.
- TM[2:0] = the level of interrupt being acknowledged.
- A[31:5] = 0x7F_FFFF.
- A[4:2] = the interrupt request level being acknowledged (same as TM[2:0]).
- A[1:0] = 00.

During the interrupt-acknowledge bus cycle (a read cycle), the responding device places the vector number on D[31:24] and the cycle is terminated normally with $\overline{\text{TA}}$. Figure 18-23 is a flow diagram for an interrupt-acknowledge cycle terminated with $\overline{\text{TA}}$.

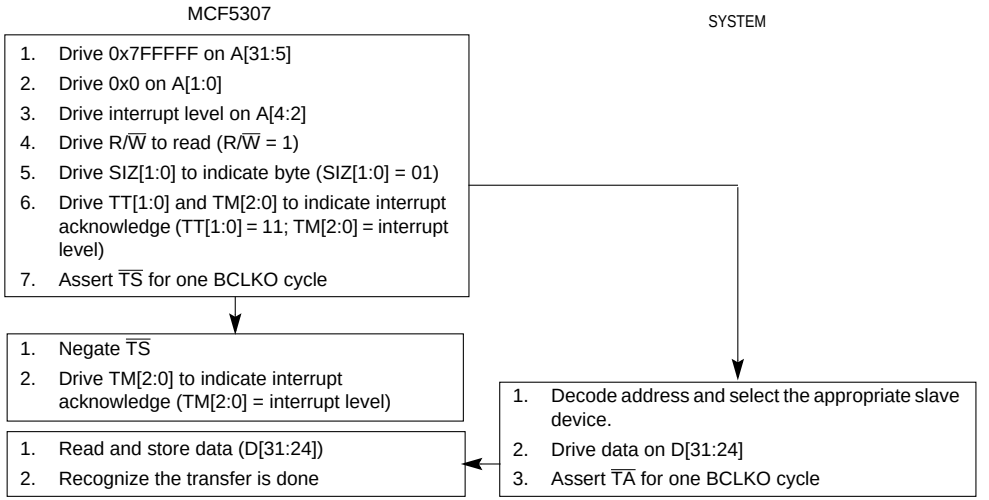


Figure 18-23. Interrupt-Acknowledge Cycle Flowchart

18.8 Bus Arbitration

The MCF5307 bus protocol gives either the MCF5307 or an external device access to the external bus. If more than one external device uses the bus, an external arbiter can prioritize requests and determine which device is bus master. When the MCF5307 is bus master, it uses the bus to fetch instructions and transfer data to and from external memory. When an external device is bus master, the MCF5307 can monitor the external master's transfers and interact through its chip-select, DRAM control, and transfer termination signals. See Section 10.4.1.3, "Chip-Select Control Registers (CSCR0–CSCR7)," and Chapter 11, "Synchronous/Asynchronous DRAM Controller Module."

Two-wire bus arbitration is used where the MCF5307 shares the bus with a single external device. This mode uses $\overline{BG}$ and $\overline{BD}$. The external device can ignore $\overline{BR}$. Three-wire mode is used where the MCF5307 shares the bus with multiple external devices. This requires an external bus arbiter and uses $\overline{BG}$, $\overline{BD}$, and $\overline{BR}$. In either mode, the MCF5307 bus arbiter operates synchronously and transitions between states on the rising edge of BCLKO.

Table 18-6 shows the four arbitration states the MCF5307 can be in during bus operation.

Table 18-6. MCF5307 Arbitration Protocol States

State	Master	Bus	$\overline{BD}$	Description
Reset	None	Not driven	Negated	The MCF5307 enters reset state from any other state when $\overline{RSTI}$ or software watchdog reset is asserted. If both are negated, the MCF5307 enters implicit or external device mastership state, depending on $\overline{BG}$.
Implicit master	MCF5307	Not driven	Negated	The MCF5307 is bus master ($\overline{BG}$ input is asserted) but is not ready to begin a bus cycle. It continues to three-state the bus until an internal bus request.

Table 18-6. MCF5307 Arbitration Protocol States (Continued)

State	Master	Bus	$\overline{BD}$	Description
Explicit master	MCF5307	Driven	Asserted	The MCF5307 is explicit bus master when $\overline{BG}$ is asserted and at least one bus cycle has been initiated. It asserts $\overline{BD}$ and retains explicit mastership until $\overline{BG}$ is negated even if no active bus cycles are executed. It releases the bus at the end of the current bus cycle, then negates $\overline{BD}$ and three-states the bus signals.
External master	External	Not driven	Negated	An external device is bus master ($\overline{BG}$ negated to MCF5307). The MCF5307 can assert $\overline{OE}$, $\overline{CS}$[7:0], $\overline{BE}$/$\overline{BWE}$[3:0], $\overline{TA}$, and all DRAM controller signals ($\overline{RAS}$[1:0], $\overline{CAS}$[3:0], $\overline{SRAS}$, $\overline{SCAS}$, $\overline{DRAMW}$, SCKE).

If the MCF5307 is the only possible master, $\overline{BG}$ can be tied to GND—no arbiter is needed.

18.8.1 Bus Arbitration Signals

Bus arbitration signal timings in Table 18-7 are referenced to the system clock, which is not considered a bus signal. Clock routing is expected to meet application requirements.

Table 18-7. ColdFire Bus Arbitration Signal Summary

Signal	I/O	Description
$\overline{BR}$	O	Bus request. Indicates to an external arbiter that the processor needs to become bus master. $\overline{BR}$ is negated when the MCF5307 begins an access to the external bus with no other internal accesses pending. $\overline{BR}$ remains negated until another internal request occurs.
$\overline{BG}$	I	Bus grant. An external arbiter asserts $\overline{BG}$ to indicate that the MCF5307 can control the bus at the next rising edge of BCLKO. When the arbiter negates $\overline{BG}$, the MCF5307 must release the bus as soon as the current transfer completes. The external arbiter must not grant the bus to any other device until both $\overline{BD}$ and $\overline{BG}$ are negated.
$\overline{BD}$	O	Bus driven. The MCF5307 asserts $\overline{BD}$ to indicate it is current master and is driving the bus. If it loses bus mastership during a transfer, it completes the last transfer of the current access, negates $\overline{BD}$, and three-states all bus signals on the rising edge of BCLKO. If it loses mastership during an idle clock cycle, it three-states all bus signals on the rising edge of BCLKO.

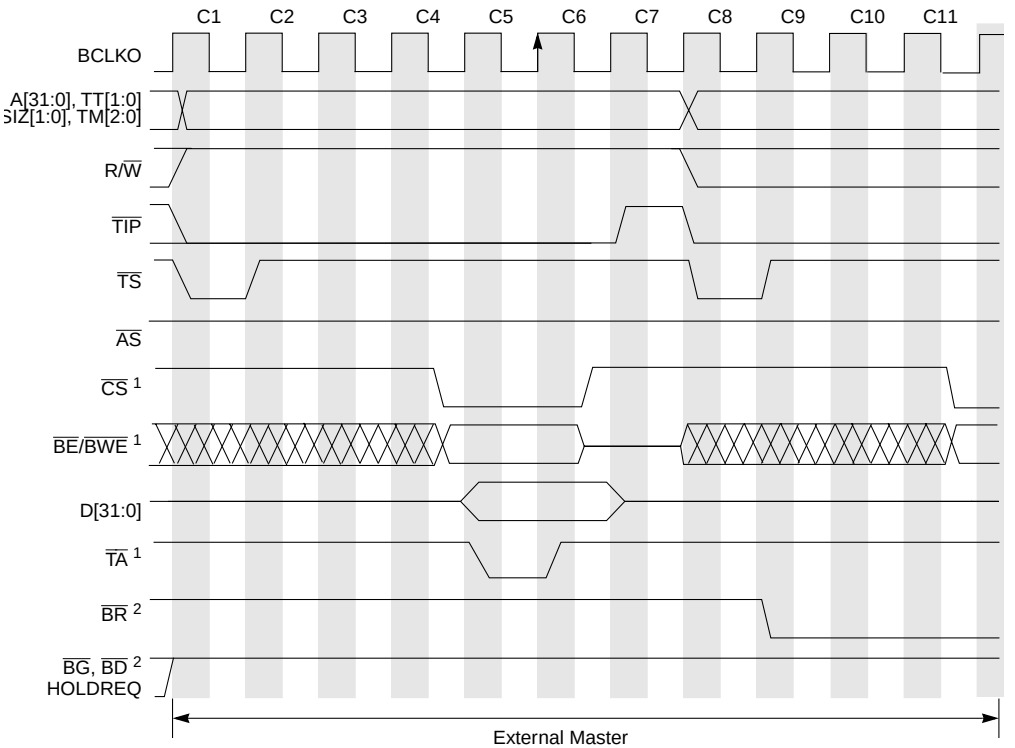
18.9 General Operation of External Master Transfers

An external master asserts its hold signal (such as HOLDREQ) when it executes a bus cycle, driving $\overline{BG}$ high and forcing the MCF5307 to hold all bus requests. During an external master cycle, the MCF5307 can provide memory control signals ($\overline{OE}$, $\overline{CS}$[7:0], $\overline{BE}$/$\overline{BWE}$[3:0], $\overline{RAS}$[1:0], $\overline{CAS}$[3:0]) and $\overline{TA}$ while the external master drives the address and data bus and other required bus control signals. When the external master asserts $\overline{TS}$ or $\overline{AS}$ to the MCF5307, the beginning of a bus cycle is identified and the MCF5307 starts decoding the address driven.

Note the following regarding external master accesses:

- For the MCF5307 to assert a $\overline{CSx}$ during external master accesses, CSMR*n*[AM] must be set. External master hits use the corresponding CSCR*n* settings for auto-acknowledge, byte enables, and wait states. See Section 10.4.1.3, “Chip-Select Control Registers (CSCR0–CSCR7).”
- To enable DRAM control signals during external master accesses, DCMR*n*[AM] must be set.
- During external master bus cycles, either $\overline{TS}$ or $\overline{AS}$ (but not both) should be driven to the MCF5307. Driving both during a bus cycle causes indeterminate results.

External master transfers that use the MCF5307 to drive memory control signals and $\overline{TA}$ are like normal MCF5307 transfers. Figure 18-24 shows timing for basic back-to-back bus cycles during an external master transfer.



1 Depending on programming, these signals may or may not be driven by the processor.
2 This signal is driven by the processor for an external master transfer.

Figure 18-24. Basic No-Wait-State External Master Access

R/$\overline{W}$ is asserted high for reads and low for writes; otherwise, the transfers are the same. In Figure 18-24, the MCF5307 chip select’s internal transfer acknowledge is enabled and the MCF5307 drives $\overline{TA}$ as an output after a programmed number of wait states.

NOTE:

Bus timing diagrams for external master transfers are not valid for on-chip internal four-channel DMA accesses on the MCF5307.

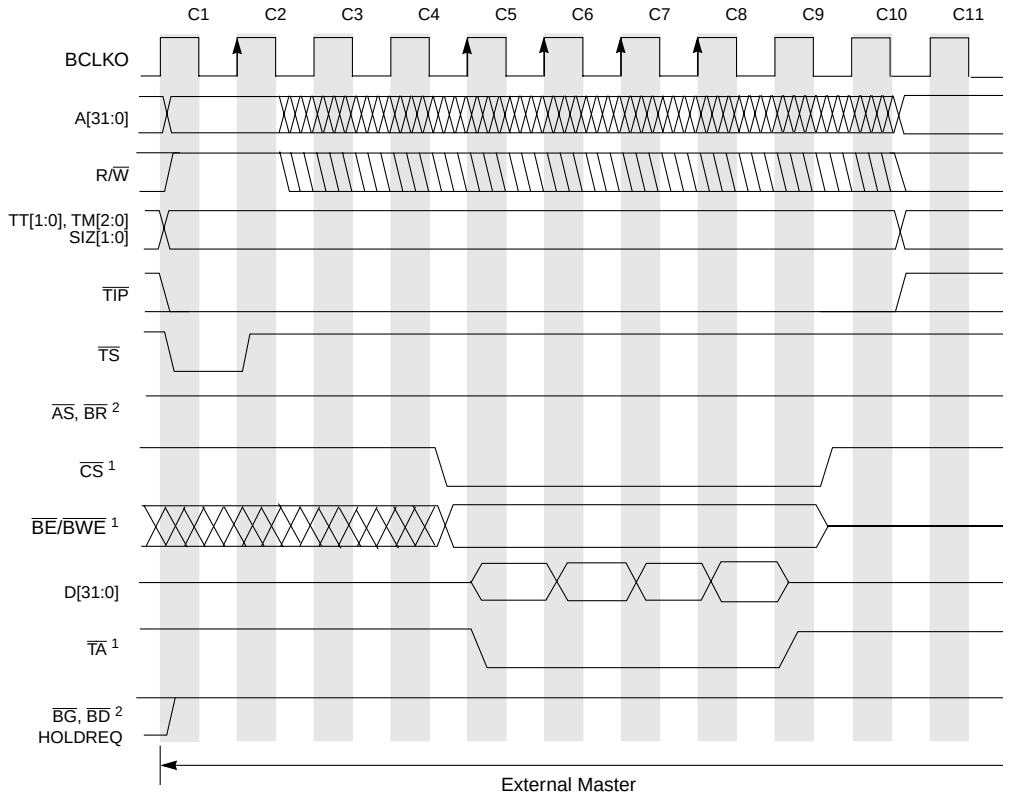
Timing diagrams describe transactions in general terms of bus cycles (C*n*) rather than the states (S*n*) used in the bus diagrams.

Table 18-8 defines the cycles for Figure 18-24.

Table 18-8. Cycles for Basic No-Wait-State External Master Access

Cycle	Definition
C1	The external master asserts HOLDREQ, signaling the MCF5307 to hold bus requests. $\overline{BD}$ should not be asserted. The external master drives address, $\overline{TS}$, R/$\overline{W}$, TT[1:0], TM[2:0], $\overline{TIP}$, and SIZ[1:0] as MCF5307 inputs.
C2–C3	The MCF5307 decodes the external master's address and control signals to identify the proper chip select and byte enable assertion. The external master negates $\overline{TS}$ in C2.
C4	On the falling edge of BCLKO, the MCF5307 asserts the appropriate chip select for the external master access along with the appropriate byte enables.
C5	On the rising edge of BCLKO, data is driven onto the bus by the device selected by $\overline{CS}$. On the rising edge, the MCF5307 asserts $\overline{TA}$ to indicate the cycle is complete.
C6	$\overline{TA}$ negates on the rising edge of BCLKO. On the falling edge, the MCF5307 negates the chip select and byte enables and the next cycle can begin.
C7	The external master negates $\overline{TIP}$ on the rising edge of BCLKO.
C8	The external device retains bus mastership and drives the address bus, $\overline{TS}$, R/$\overline{W}$, TT[1:0], TM[2:0], $\overline{TIP}$, and SIZ[1:0] as inputs to the MCF5307.
C9	The MCF5307 decodes the external master's address and control signals to identify the proper chip select and byte enable assertion. The external master negates $\overline{TS}$. The MCF5307 asserts $\overline{BR}$ on the rising edge of BCLKO, signalling that it wants to arbitrate for the bus when the current cycle completes.
C10	The MCF5307 continues to decode the external device's address and control signals to identify the proper chip select and byte enable assertion.
C11	On the falling edge of BCLKO, the MCF5307 asserts the appropriate chip select for the external master access along with the appropriate byte enables.

Figure 18-25 shows a burst line access for an external master transfer with the chip select set to no-wait states and with internal transfer-acknowledge assertion enabled.



1 Depending on programming, these signals may or may not be driven by the processor.
2 These signals are driven by the processor for an external master transfer.

Figure 18-25. External Master Burst Line Access to 32-Bit Port

Table 18-9 defines the cycles for Figure 18-25.

Table 18-9. Cycles for External Master Burst Line Access to 32-Bit Port

Cycle	Definition
C1	The external device is bus master and asserts HOLDREQ, indicating to the MCF5307 to hold all bus requests. In other words, BD should not be asserted. The external master drives address, TS, R/W, TT[1:0], TM[2:0], TIP, and SIZ[1:0] as inputs to the MCF5307. SIZ[1:0] inputs indicate a line transfer. The MCF5307 is not asserting BR.
C2–C3	The MCF5307 decodes the external device's address and control signals to identify the proper chip-select and byte-enable assertion. The external device negates TS in C2. Address and R/W are latched in the MCF5307 on the rising edge of BCLKO in C2. After C2, the address and R/W are ignored for the rest of the burst transfer.
C4	On the falling edge of BCLKO, the MCF5307 asserts the appropriate chip select for the external device access along with the appropriate byte enables.
C5	On the rising edge of BCLKO, data is driven onto the bus by the device selected by CS. The MCF5307 asserts TA on the rising edge of BCLKO, indicating the first data transfer is complete.

Table 18-9. Cycles for External Master Burst Line Access to 32-Bit Port (Continued)

Cycle	Definition
C6–C8	No-wait state data transfers 2–4 occur on the rising edges of BCLKO. $\overline{TA}$ continues to be asserted indicating completion of each transfer. $\overline{TIP}$, $\overline{CSx}$, and $\overline{BE}/\overline{BWE}$[3:0] are driven.
C9	$\overline{TA}$ negates on the rising edge of BCLKO along with external device's negation of $\overline{TIP}$. On the falling edge, the MCF5307 negates chip select and byte enables, creating an opportunity for another cycle to begin.

18.9.1 Two-Device Bus Arbitration Protocol (Two-Wire Mode)

Two-wire mode bus arbitration lets the MCF5307 share the external bus with a single external bus device without requiring an external bus arbiter. Figure 18-26 shows the MCF5307 connecting to an external device using the two-wire mode. The MCF5307 $\overline{BG}$ input is connected to the HOLDREQ output of the external device; the MCF5307 $\overline{BD}$ output is connected to the HOLDACK input of the external device. Because the external device controls the state of HOLDREQ, it controls when the MCF5307 is granted the bus, giving the MCF5307 lower priority.

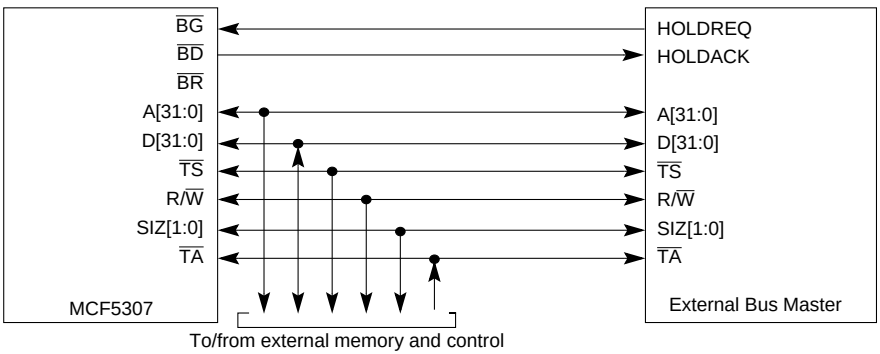


Figure 18-26. MCF5307 Two-Wire Mode Bus Arbitration Interface

When the external device is not using the bus, it negates HOLDREQ, driving $\overline{BG}$ low and granting the bus to the MCF5307. When the MCF5307 has an internal bus request pending and $\overline{BG}$ is low, the MCF5307 drives $\overline{BD}$ low, negating HOLDACK to the external device. When the external bus device needs the external bus, it asserts HOLDREQ, driving $\overline{BG}$ high, requesting the MCF5307 to release the bus. If $\overline{BG}$ is negated while a bus cycle is in progress, the MCF5307 releases the bus at the completion of the bus cycle. Note that the MCF5307 considers the individual transfers of a burst or burst-inhibited access to be a single bus cycle and does not release the bus until the last transfer of the series completes.

When the bus has been granted to the MCF5307, one of two situations can occur. In the first case, if the MCF5307 has an internal bus request pending, the MCF5307 asserts $\overline{BD}$ to indicate explicit bus mastership and begins the pending bus cycle by asserting $\overline{TS}$. As

shown in Figure 18-25, the MCF5307 continues to assert $\overline{BD}$ until the completion of the bus cycle. If $\overline{BG}$ is negated by the end of the bus cycle, the MCF5307 negates $\overline{BD}$. While $\overline{BG}$ is asserted, $\overline{BD}$ remains asserted to indicate the MCF5307 is master, and it continuously drives the address bus, attributes, and control signals.

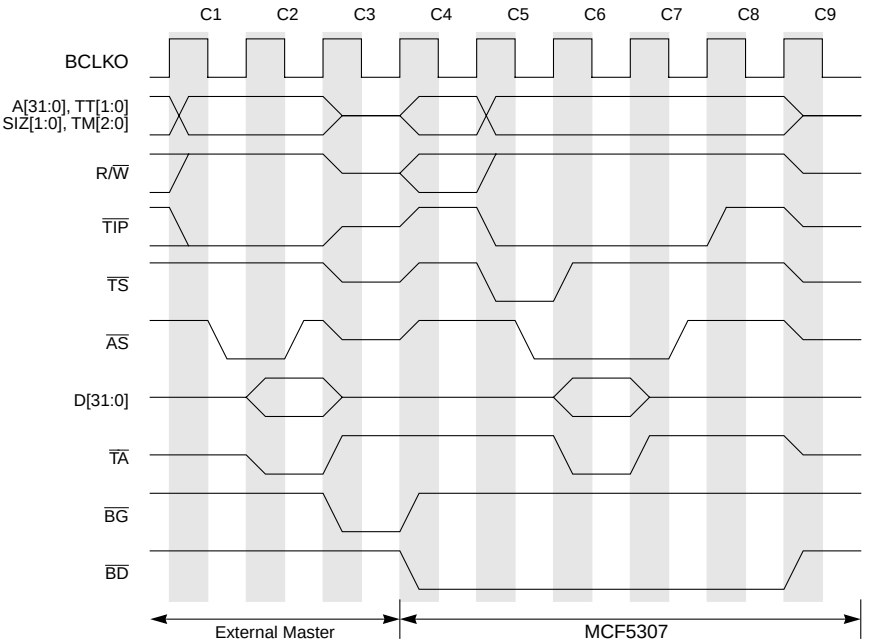


Figure 18-27. Two-Wire Bus Arbitration with Bus Request Asserted

In the second situation, the bus is granted to the MCF5307, but it does not have an internal bus request pending, so it takes implicit bus mastership. The MCF5307 does not drive the bus and does not assert $\overline{BD}$ if the bus has an implicit master. If an internal bus request is generated, the MCF5307 assumes explicit bus mastership. If explicit mastership was assumed because an internal request was generated, the MCF5307 immediately begins an access and asserts $\overline{BD}$.

In Figure 18-28, the external device is bus master during C1 and C2. During C3 the external device releases control of the bus by asserting $\overline{BG}$ to the MCF5307. At this point, there is an internal access pending so the MCF5307 asserts $\overline{BD}$ during C4 and begins the access. Thus, the MCF5307 becomes the explicit external bus master. Also during C4, the external device removes the grant from the MCF5307 by negating $\overline{BG}$. As the current bus master, the MCF5307 continues to assert $\overline{BD}$ until the current transfer completes. Because $\overline{BG}$ is negated, the MCF5307 negates $\overline{BD}$ during C9 and three-states the external bus, thereby returning external bus mastership to the external device.

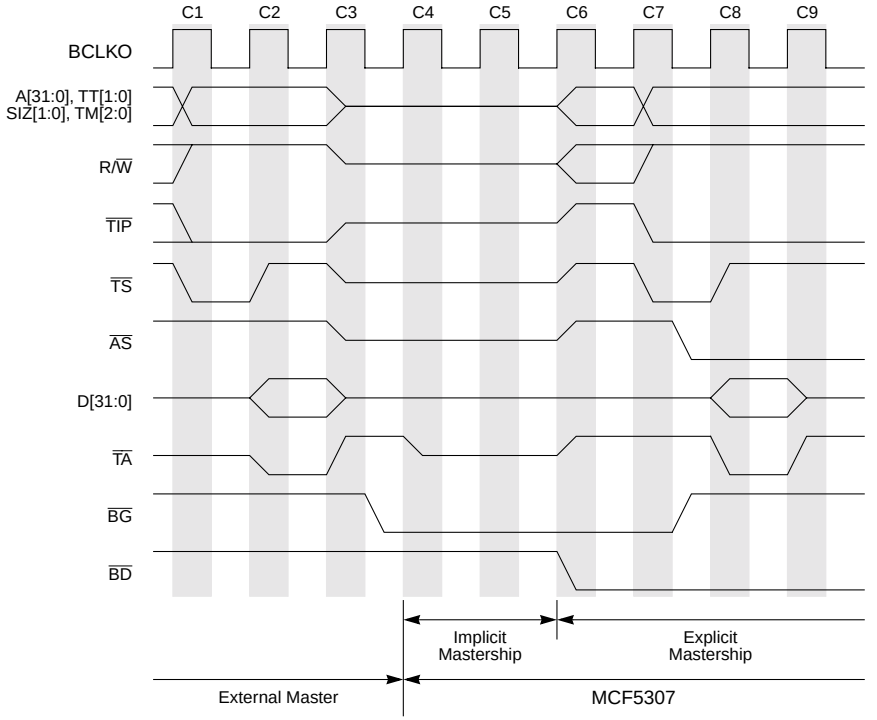


Figure 18-28. Two-Wire Implicit and Explicit Bus Mastership

In Figure 18-28, the external device is master during C1 and C2. It releases bus control in C3 by asserting $\overline{BG}$ to the MCF5307. During C4 and C5, the MCF5307 is implicit master because no internal access is pending. In C5, an internal bus request becomes pending, causing the MCF5307 to become explicit bus master in C6 by asserting $\overline{BD}$. In C7, the external device removes the bus grant to the MCF5307. The MCF5307 does not release the bus (the MCF5307 continues to assert $\overline{BD}$) until the transfer ends.

NOTE:

The MCF5307 can start a transfer in the clock cycle after $\overline{BG}$ is asserted. The external master must not assert $\overline{BG}$ to the MCF5307 while driving the bus or the part may be damaged.

Chapter 5, “Debug Support is a MCF5307 bus arbitration state diagram. States are described in Table 18-6.

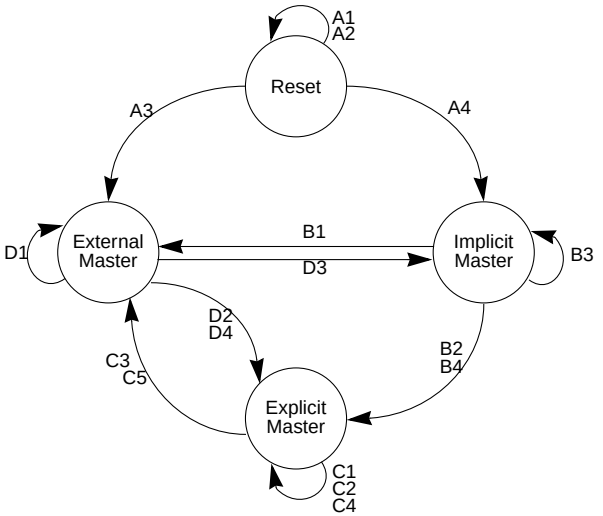


Figure 18-29. MCF5307 Two-Wire Bus Arbitration Protocol State Diagram

Table 18-10 describes the two-wire bus arbitration protocol transition conditions.

Table 18-10. MCF5307 Two-Wire Bus Arbitration Protocol Transition Conditions

Present State	Condition Label	RSTI	Software Watchdog Reset	BG	Bus Request	Transfer in Progress	End of Cycle[1]	Next State
Reset	A1	A[2]	—	—	—	—	—	Reset
	A2	N[3]	A	—	—	—	—	Reset
	A3	N	N	N	—	—	—	EM [4]
	A4	N	N	A	—	—	—	Implicit mas
Implicit Master	B1	N	N	N	—	—	—	EM
	B2	N	N	A	—	—	—	Explicit mas
	B3	N	N	A	N	—	—	Implicit mas
	B4	N	N	A	A	—	—	Explicit mas
Explicit Master	C1	N	N	A	—	—	—	Explicit mas
	C2	N	N	N	—	—	—	Explicit mas
	C3	N	N	N	—	N	—	EM
	C4	N	N	N	—	A	N	Explicit mas
	C5	N	N	N	—	A	A	EM
External Master	D1	N	N	N	—	—	—	EM mas
	D2	N	N	A	—	—	_	Explicit mas
	D3	N	N	A	N	—	—	Implicit mas
	D4	N	N	A	A	—	—	Explicit mas

1 Both normal terminations and terminations due to bus errors generate an end of cycle. Bus cycles resulting from a burst-inhibited transfer are considered part of that original transfer.
2 A means asserted.
3 N means negated.
4 EM means external master.

18.9.2 Multiple External Bus Device Arbitration Protocol (Three-Wire Mode)

Three-wire mode lets the MCF5307 share the external bus with multiple external devices. This mode requires an external arbiter to assign priorities to each potential master and to determine which device accesses the external bus. The arbiter uses the MCF5307 bus arbitration signals, $\overline{BR}$, $\overline{BD}$, and $\overline{BG}$, to control use of the external bus by the MCF5307.

The MCF5307 requests the bus from the external bus arbiter by asserting $\overline{BR}$ when the core requests an access. It continues to assert $\overline{BR}$ until after the transfer starts. It can negate $\overline{BR}$ at any time regardless of the $\overline{BG}$ status. If the MCF5307 is granted the bus when an internal bus request is generated, it asserts $\overline{BD}$ and the access begins immediately. The MCF5307 always drives $\overline{BR}$ and $\overline{BD}$, which cannot be directly wire-ORed with other devices.

The external arbiter asserts $\overline{BG}$ to grant the bus to MCF5307, which can begin a bus cycle after the next rising edge of BCLKO. If $\overline{BG}$ is negated during a bus cycle, the MCF5307 releases the bus when the cycle completes. To guarantee that the bus is released, $\overline{BG}$ must be negated before the rising edge of the BCLKO in which the last $\overline{TA}$ is asserted. Note that the MCF5307 treats any series of burst or a burst-inhibited transfers as a single bus cycle and does not release the bus until the last transfer of the series completes.

When the MCF5307 is granted the bus after it asserts $\overline{BR}$, one of two things can occur. If the MCF5307 has an internal bus request pending, it asserts $\overline{BD}$, indicating explicit bus mastership, and begins the pending bus cycle by asserting $\overline{TS}$. The MCF5307 continues to assert $\overline{BD}$ until the external bus arbiter negates $\overline{BG}$, after which $\overline{BD}$ is negated at the completion of the bus cycle. As long as $\overline{BG}$ is asserted, $\overline{BD}$ remains asserted to indicate that the MCF5307 is bus master, and the MCF5307 continuously drives the address bus, attributes, and control signals.

If no internal request is pending, the MCF5307 takes implicit bus mastership. It does not drive the bus and does not assert $\overline{BD}$ if the bus has an implicit master. If an internal bus request is generated, the MCF5307 assumes explicit bus mastership and immediately begins an access and asserts $\overline{BD}$. Figure 18-30 shows implicit and explicit bus mastership due to generation of an internal bus request.

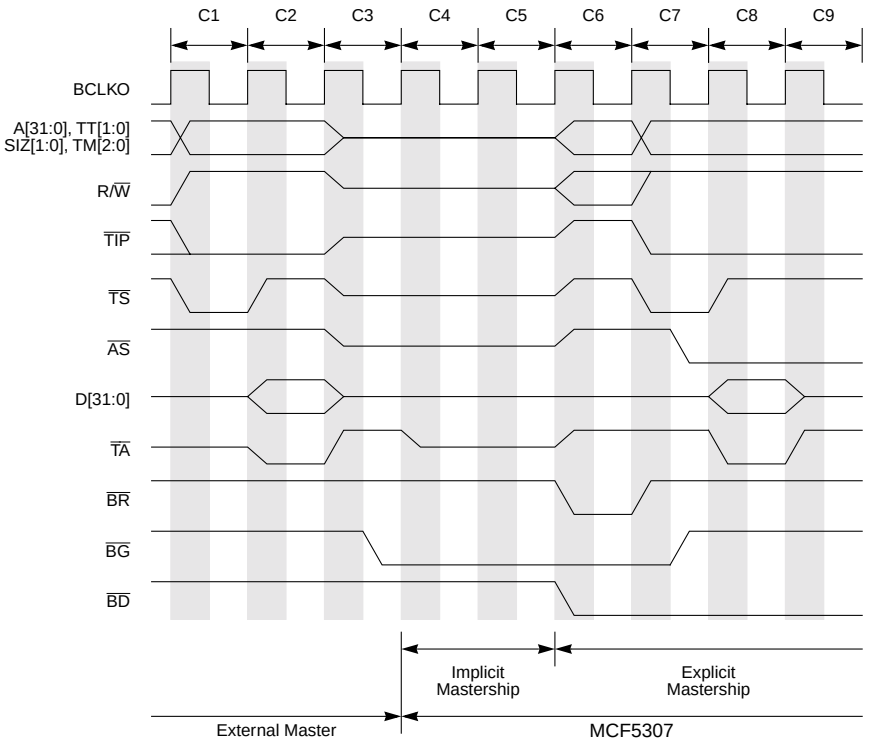


Figure 18-30. Three-Wire Implicit and Explicit Bus Mastership

In Figure 18-30, the external device is bus master during C1 and C2, releasing control in C3, at which time the external arbiter asserts $\overline{BG}$ to the MCF5307. During C4 and C5, the MCF5307 is implicit master because no internal access is pending. In C5, an internal bus request becomes pending, causing the MCF5307 to take explicit bus mastership in C6 by asserting $\overline{BR}$ and $\overline{BD}$. In C7, the external device removes the bus grant to the MCF5307. The MCF5307 does not release the bus (the MCF5307 asserts $\overline{BD}$) until the transfer ends.

NOTE:

The MCF5307 can start a transfer in the cycle after $\overline{BG}$ is asserted. The external arbiter should not assert $\overline{BG}$ to the MCF5307 until the previous external master stops driving the bus. Asserting $\overline{BG}$ during another external master's transfer may damage the part.

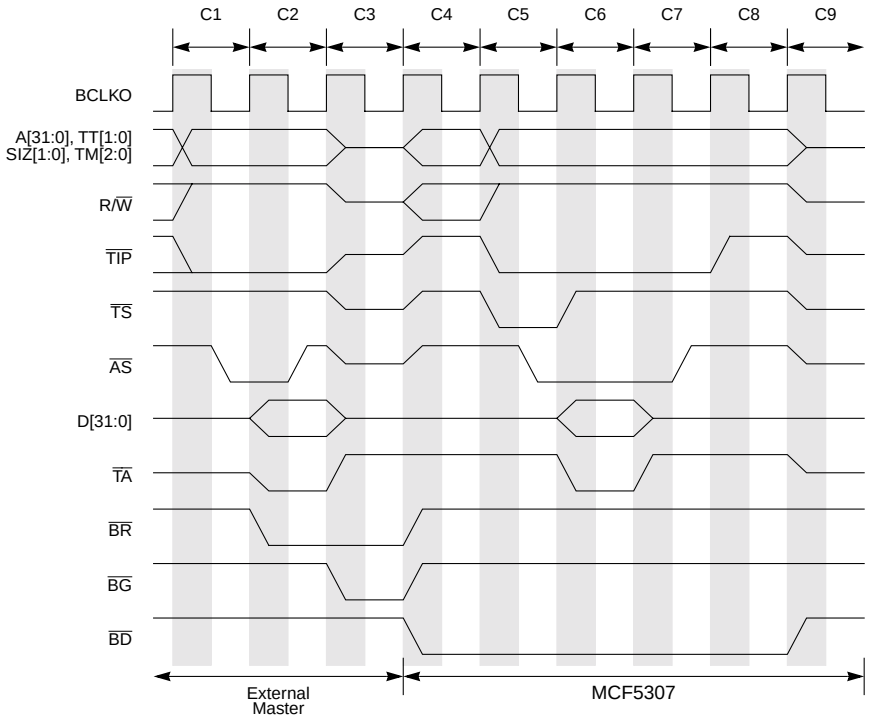


Figure 18-31. Three-Wire Bus Arbitration

In Figure 18-31, the external device is bus master during C1 and C2. During C2, the MCF5307 requests the external bus because of a pending internal transfer. On C3, the external releases mastership and the external arbiter grants the bus to the MCF5307 by asserting $\overline{BG}$. At this point, an internal is access pending so the MCF5307 asserts $\overline{BD}$ during C4 and begins the access. Thus, the MCF5307 becomes the explicit bus master. Also during C4, the external arbiter removes the grant from the MCF5307 by negating $\overline{BG}$. Because the MCF5307 is bus master, it continues to assert $\overline{BD}$ until the current transfer completes. Because $\overline{BG}$ is negated, the MCF5307 negates $\overline{BD}$ during C9 and three-states the external bus, thereby passing mastership to an external device.

The MCF5307 can assert $\overline{BR}$ to signal the external arbiter that it needs the bus. However, there is no guarantee that when the bus is granted to the MCF5307 that a bus cycle will be performed. At best, $\overline{BR}$ must be used as a status output that indicates when the MCF5307 needs the bus, but not as an indication that the MCF5307 is in a certain bus arbitration state.

Figure 18-32 is a high-level state diagram for MCF5307 bus arbitration protocol. Table 18-6 describes the four states shown in Figure 18-32.

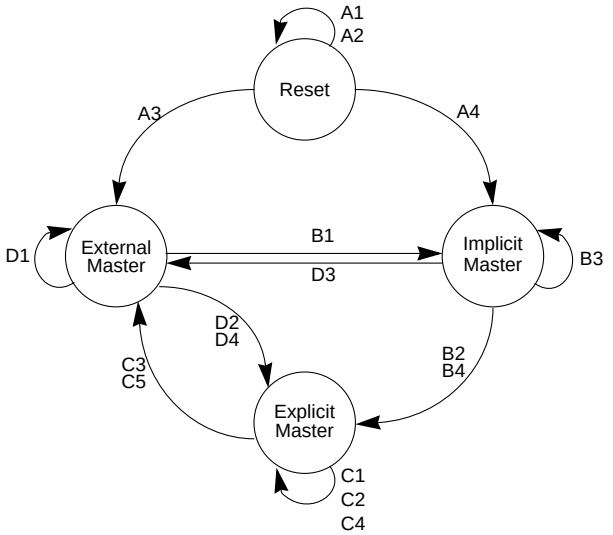


Figure 18-32. Three-Wire Bus Arbitration Protocol State Diagram

Table 18-11 lists conditions that cause state transitions.

Table 18-11. Three-Wire Bus Arbitration Protocol Transition Conditions

Current State	Condition Label	$\overline{\text{RSTI}}$	Software Watchdog Reset	$\overline{\text{BG}}$	Bus Request	Transfer in Progress	End of Cycle [1]	Next State
Reset	A1	Asserted	—	—	—	—	—	Reset
	A2	Negated	Asserted	—	—	—	—	Reset
	A3	Negated	Negated	Negated	—	—	—	EM
	A4	Negated	Negated	Asserted	—	—	—	Implicit master
Implicit master	B1	Negated	Negated	Negated	—	—	—	External device master
	B2	Negated	Negated	Asserted	—	—	—	Explicit master
	B3	Negated	Negated	Asserted	Negated	—	—	Implicit master
	B4	Negated	Negated	Asserted	Asserted	—	—	Explicit master

Table 18-11. Three-Wire Bus Arbitration Protocol Transition Conditions (Continued)

Current State	Condition Label	$\overline{\text{RSTI}}$	Software Watchdog Reset	$\overline{\text{BG}}$	Bus Request	Transfer in Progress	End of Cycle [1]	Next State
Explicit master	C1	Negated	Negated	Asserted	—	—	—	Explicit master
	C2	Negated	Negated	Negated	—	—	—	Explicit master
	C3	Negated	Negated	Negated	—	Negated	—	External device master
	C4	Negated	Negated	Negated	—	Yes	Negated	Explicit master
	C5	Negated	Negated	Negated	—	Yes	Yes	External device master
External master	D1	Negated	Negated	Negated[1]	—	—	—	External device master
	D2	Negated	Negated	Asserted	—	—	—	Explicit master
	D3	Negated	Negated	Asserted	Negated	—	—	Implicit master
	D4	Negated	Negated	Asserted	Asserted	—	—	Explicit master

[1] Both normal terminations and terminations due to bus errors generate an end of cycle. Bus cycles resulting from a burst-inhibited transfer are considered part of that original transfer.

The bus arbitration state diagram can be used for the MCF5307 three-wire bus arbitration protocol to approximate the high-level behavior of the MCF5307. It is assumed that all $\overline{\text{TS}}$ or $\overline{\text{AS}}$ signals in a system are tied together and each bus device's $\overline{\text{BD}}$ and $\overline{\text{BR}}$ signals are connected individually to the external arbiter. The external arbiter must ensure that any external masters will have released the bus after the next rising edge of before asserting $\overline{\text{BG}}$ to the MCF5307. The MCF5307 does not monitor external bus master operation regarding bus arbitration.

NOTE:

The MCF5307 can start a transfer on the rising edge of the cycle after $\overline{\text{BG}}$ is asserted. The external arbiter should not assert $\overline{\text{BG}}$ to the MCF5307 until the previous external master stops driving the bus or the part may be damaged.

18.10 Reset Operation

The MCF5307 supports two types of reset. Asserting $\overline{\text{RSTI}}$ resets the entire MCF5307. A software watchdog reset resets everything but the internal PLL module.

18.10.1 Master Reset

To perform a master reset, an external device asserts $\overline{\text{RSTI}}$. When power is applied to the system, external circuitry should assert $\overline{\text{RSTI}}$ for a minimum of 80 CLKIN cycles after Vcc is within tolerance. Figure 18-33 is a functional timing diagram of the master reset operation, showing relationships among Vcc, $\overline{\text{RSTI}}$, mode selects, and bus signals. CLKIN must be stable by the time Vcc reaches the minimum operating specification. CLKIN should start oscillating as Vcc is ramped up to clear out contention internal to the MCF5307 caused by the random states of internal flip-flops on power up. $\overline{\text{RSTI}}$ is internally synchronized for two CLKIN cycles before being used and must meet the specified setup and hold times in relationship to CLKIN to be recognized.

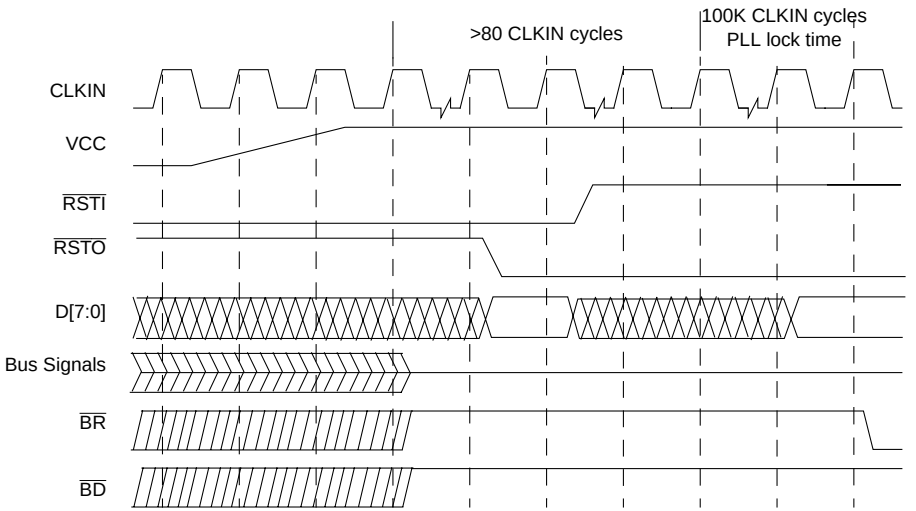


Figure 18-33. Master Reset Timing

During the master reset period, all signals capable of being three-stated are driven to a high-impedance; all others are negated. When $\overline{\text{RSTI}}$ negates, all bus signals remain in a high-impedance state until the MCF5307 is granted the bus and the core begins the first bus cycle for reset exception processing. A master reset causes any bus cycle (including DRAM refresh cycle) to terminate and initializes registers appropriately for a reset exception.

Note that during reset D[7:0] are sampled on the negating edge of $\overline{\text{RSTI}}$ for initial MCF5307 configurations listed in Table 18-12.

Table 18-12. Data Pin Configuration

Pin	Function
D7	Auto-Acknowledge Configuration (AA_CONFIG)
D[6:5]	Port Size Configuration (PS_CONFIG[1:0])
D4	Address Configuration (ADDR_CONFIG/D4)
D[3:2]	Frequency of CLKIN (FREQ[1:0])
D[1:0]	Ratio of BCLKO/Processor Clock {DIVIDE[1:0])

See Section 17.5.5, “Data/Configuration Pins (D[7:0]).” Motorola recommends that the data pins be driven rather than using a weak pull-up or pull-down resistor. Table 17-1 lists the encoding of these pins sampled at reset.

18.10.2 Software Watchdog Reset

A software watchdog reset is performed if the executing software does not provide the correct write data sequence with the enable-control bit set. This reset helps prevent runaway software or unterminated bus cycles. Figure 18-34 is a functional timing diagram of the software watchdog reset operation, showing $\overline{\text{RSTO}}$ and bus signal relationships.

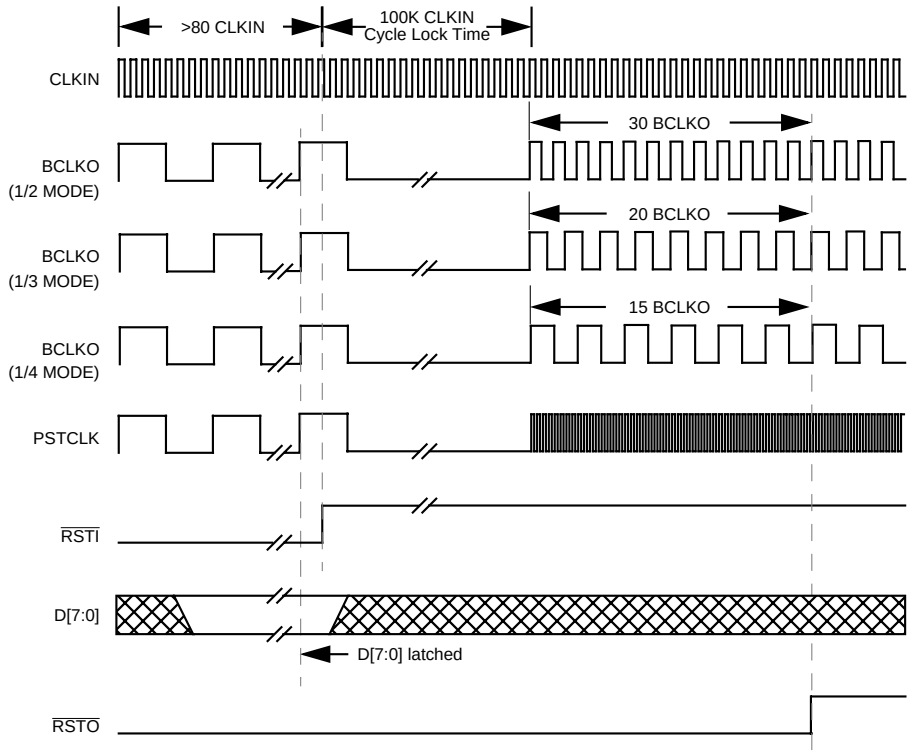


Figure 18-34. Software Watchdog Reset Timing

During the software watchdog reset period, all signals that can be are driven to a high-impedance state; all those that cannot be are negated. When $\overline{\text{RSTO}}$ negates, bus signals remain in a high-impedance state until the MCF5307 is granted the bus and the ColdFire core begins the first bus cycle for reset exception processing.

Chapter 19
IEEE 1149.1 Test Access Port (JTAG)

This chapter describes configuration and operation of the MCF5307 JTAG test implementation. It describes the use of JTAG instructions and provides information on how to disable JTAG functionality.

19.1 Overview

The MCF5307 dedicated user-accessible test logic is fully compliant with the publication *Standard Test Access Port and Boundary-Scan Architecture*, IEEE Standard 1149.1. Use the following description in conjunction with the supporting IEEE document listed above. This section includes the description of those chip-specific items that the IEEE standard requires as well as those items specific to the MCF5307 implementation.

The MCF5307 JTAG test architecture supports circuit board test strategies based on the IEEE standard. This architecture provides access to all data and chip control pins from the board-edge connector through the standard four-pin test access port (TAP) and the JTAG reset pin, $\overline{\text{TRST}}$. Test logic design is static and is independent of the system logic except where the JTAG is subordinate to other complimentary test modes, as described in Chapter 5, “Debug Support.” When in subordinate mode, JTAG test logic is placed in reset and the TAP pins can be used for other purposes, as described in Table 19-1.

The MCF5307 JTAG implementation can do the following:

- Perform boundary-scan operations to test circuit board electrical continuity
- Bypass the MCF5307 by reducing the shift register path to a single cell
- Set MCF5307 output drive pins to fixed logic values while reducing the shift register path to a single cell
- Sample MCF5307 system pins during operation and transparently shift out the result
- Protect MCF5307 system output and input pins from backdriving and random toggling (such as during in-circuit testing) by placing all system pins in high-impedance state

NOTE:

IEEE Standard 1149.1 may interfere with system designs that do not incorporate JTAG capability. Section 19.6, “Disabling IEEE Standard 1149.1 Operation,” describes precautions for ensuring that this logic does not affect system or debug operation.

Figure 19-1 is a block diagram of the MCF5307 implementation of the 1149.1 IEEE standard. The test logic includes several test data registers, an instruction register, instruction register control decode, and a 16-state dedicated TAP controller.

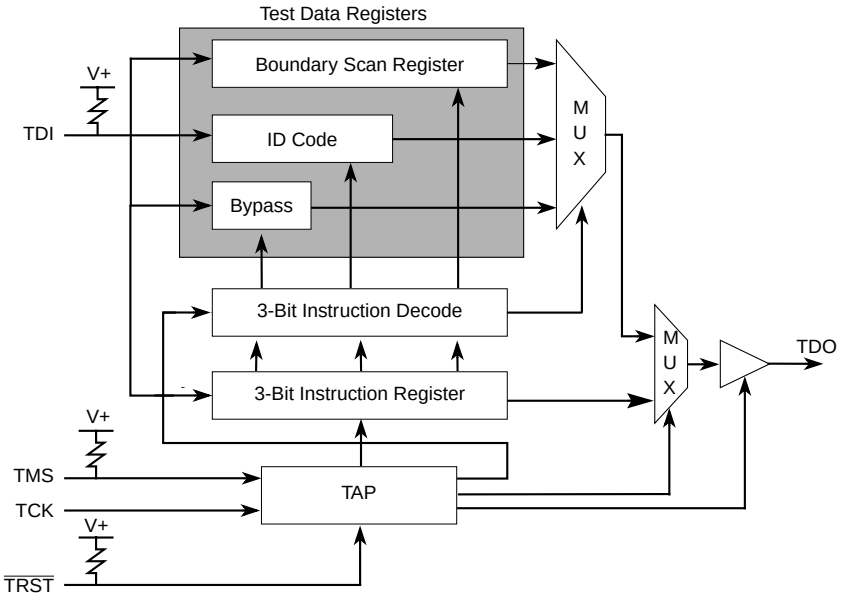


Figure 19-1. JTAG Test Logic Block Diagram

19.2 JTAG Signal Descriptions

JTAG operation on the MCF5307 is enabled when MTMOD0 is high (logic 1), as described in Table 19-1. Otherwise, JTAG TAP signals, TCK, TMS, TDI, TDO, and $\overline{\text{TRST}}$, are interpreted as the debug port pins. MTMOD0 should not be changed while $\overline{\text{RSTI}}$ is asserted.

Table 19-1. JTAG Pin Descriptions

Pin	Description
TCK	Test clock. The dedicated JTAG test logic clock is independent of the MCF5307 processor clock. Various JTAG operations occur on the rising or falling edge of TCK. Internal JTAG controller logic is designed such that holding TCK high or low indefinitely does cause the JTAG test logic to lose state information. If TCK is not used, it should be tied to ground.
TMS/ $\overline{\text{BKPT}}$	Test mode select (MTMOD0 high)/breakpoint (MTMOD0 low). TMS provides the JTAG controller with information to determine the test operation mode. The states of TMS and of 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, so if it is not driven low, its value defaults to a logic level of 1. If TMS is not used, it should be tied to VDD. $\overline{\text{BKPT}}$ signals a hardware breakpoint to the processor in debug mode. See Chapter 5, “Debug Support.”

Table 19-1. JTAG Pin Descriptions

Pin	Description
TDI/DSI	Test data input (MTMOD0 high)/development serial input (MTMOD0 low). TDI provides the serial data port for loading the JTAG boundary-scan, bypass, and instruction registers. Shifting in of data depends on the state of the JTAG controller state machine and the instruction in the instruction register. This shift occurs on the rising edge of TCK. TDI has an internal pull-up so if it is not driven low its value defaults to a logical 1. If TDI is not used, it should be tied to VDD. DSI provides single-bit communication for debug module commands. See Chapter 5, “Debug Support.”
TDO/ DSO	Test data output (MTMOD0 high)/development serial output (MTMOD0 low). TDO is the serial data port for outputting data from JTAG logic. Shifting data out depends on the state of the JTAG controller state machine and the instruction currently in the instruction register. This shift occurs on the falling edge of TCK. When not outputting test data, TDO is three-stated. It can also be placed in three-state mode to allow bussed or parallel connections to other devices having JTAG. DSO provides single-bit communication for debug module commands. See Chapter 5, “Debug Support.”
$\overline{\text{TRST}}$/ DSCLK	Test reset (MTMOD0 high)/development serial clock (MTMOD0 low). As $\overline{\text{TRST}}$, this pin asynchronously resets the internal JTAG controller to the test logic reset state, causing the JTAG instruction register to choose the IDCODE instruction. When this occurs, all JTAG logic is benign and does not interfere with normal MCF5307 functionality. Although this signal is asynchronous, Motorola recommends that $\overline{\text{TRST}}$ make only an asserted-to-negated transition while TMS is held at a logic 1 value. $\overline{\text{TRST}}$ has an internal pull-up; if it is not driven low its value defaults to a logic level of 1. However, if $\overline{\text{TRST}}$ is not used, it can either be tied to ground or, if TCK is clocked, to VDD. The former connection places the JTAG controller in the test logic reset state immediately; the latter connection eventually puts the JTAG controller (if TMS is a logic 1) into the test logic reset state after 5 TCK cycles. DSCLK is the development serial clock for the serial interface to the debug module.The maximum DSCLK frequency is 1/2 the BCLKO frequency. See Chapter 5, “Debug Support.”

19.3 TAP Controller

The state of TMS at the rising edge of TCK determines the current state of the TAP controller. The TAP controller can follow two basic two paths, one for executing JTAG instructions and the other for manipulating JTAG data based on JTAG instructions. The various states of the TAP controller are shown in Figure 19-2. For more detail on each state, see the IEEE Standard 1149.1 JTAG document. Note that regardless of the TAP controller state, test-logic-reset can be entered if TMS is held high for at least five rising edges of TCK. Figure 19-2 shows the JTAG TAP controller state machine.

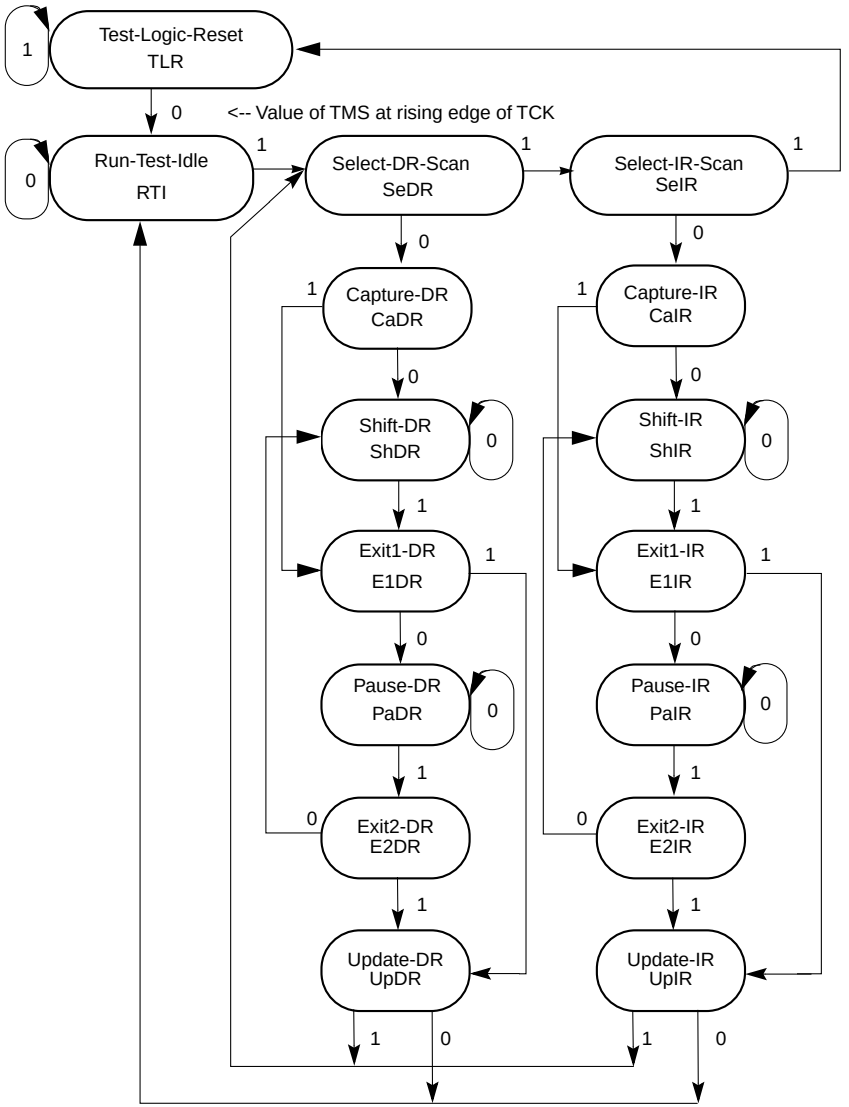


Figure 19-2. JTAG TAP Controller State Machine

19.4 JTAG Register Descriptions

The following sections describe the JTAG registers implemented on the MCF5307.

19.4.1 JTAG Instruction Shift Register

The MCF5307 IEEE Standard 1149.1 implementation uses a 3-bit instruction-shift register (IR) without parity. This register transfers its value to a parallel hold register and applies one of six instructions on the falling edge of TCK when the TAP state machine is in Update-IR state. To load instructions into the shift portion of the register, place the serial data on TDI before each rising edge of TCK. The msb of the instruction shift register is the bit closest to the TDI pin, and the lsb is the bit closest to TDO.

Table 19-2 describes customer-usable instructions.

Table 19-2. JTAG Instructions

Instruction	Class	IR	Description
EXTEST (EXT)	Required	000	Selects the boundary-scan register. Forces all output pins and bidirectional pins configured as outputs to the preloaded fixed values (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 becomes active on the falling edge of TCK in the Update-IR state when the data held in the instruction-shift register is equivalent to octal 0.
IDCODE (IDC)	Optional	001	Selects the IDCODE register for connection as a shift path between TDI and TDO. Interrogates the MCF5307 for version number and other part identification. The IDCODE register is implemented in accordance with IEEE Standard 1149.1 so the lsb of the shift register stage is set to logic 1 on the rising edge of TCK following entry into the capture-DR state. Therefore, the first bit shifted out after selecting the IDCODE register is always a logic 1. The remaining 31-bits are also set to fixed values. See Section 19.4.2, “IDCODE Register.” IDCODE is the default value in the IR when a JTAG reset occurs by either asserting $\overline{\text{TRST}}$ or holding TMS high while clocking TCK through at least five rising edges and the falling edge after the fifth rising edge. A JTAG reset causes the TAP state machine to enter test-logic-reset state (normal operation of the TAP state machine into the test-logic-reset state also places the default value of octal 1 into the instruction register). The shift register portion of the instruction register is loaded with the default value of octal 1 in Capture-IR state and a TCK rising edge occurs.
SAMPLE/ PRELOAD (SMP)	Required	100	Provides two separate functions. It obtains a sample of the system data and control signals at the MCF5307 input pins and before the boundary-scan cell at the output pins. This sampling occurs on the rising edge of TCK in the capture-DR state when an instruction encoding of octal 4 is in the instruction register. Sampled data is observed by shifting it through the boundary-scan register to TDO by using shift-DR state. The data capture and shift are transparent to system operation. The users must provide external synchronization to achieve meaningful results because there is no internal synchronization between TCK and CLK. SAMPLE/PRELOAD also initializes the boundary-scan register update cells before selecting EXTEST or CLAMP. This is done by ignoring data shifted out of TDO while shifting in initialization data. The Update-DR state in conjunction with the falling edge of TCK can then transfer this data to the update cells. This data is applied to external outputs when an instruction listed above is applied.
HIGHZ (HIZ)	Optional	101	Anticipates the need to backdrive outputs and protects inputs from random toggling during board testing. Selects the bypass register, forcing all output and bidirectional pins into high-impedance. HIGHZ goes active on the falling edge of TCK in the Update-IR state when instruction shift register data held is equivalent to octal 5.

Table 19-2. JTAG Instructions (Continued)

Instruction	Class	IR	Description
CLAMP (CMP)	Optional	110	Selects the bypass register and asserts functional reset while forcing all output and bidirectional pins configured as outputs to fixed, preloaded values in the boundary-scan update registers. Enhances test efficiency by reducing the overall shift path to one bit (the bypass register) while conducting an EXTEST type of instruction through the boundary-scan register. CLAMP becomes active on the falling edge of TCK in the Update-IR state when instruction-shift register data is equivalent to octal 6.
BYPASS (BYP)	Required	111	Selects the single-bit bypass register, creating a single-bit shift register path from TDI to the bypass register to TDO. Enhances test efficiency by reducing the overall shift path when a device other than the MCF5307 is under test on a board design with multiple chips on the overall 1149.1 defined boundary-scan chain. The bypass register is implemented in accordance with 1149.1 so the shift register stage is set to logic 0 on the rising edge of TCK following entry into the capture-DR state. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0 (to differentiate a part that supports an IDCODE register from a part that supports only the bypass register). BYPASS goes active on the falling edge of TCK in the Update-IR state when instruction shift register data is equivalent to octal 7.

The IEEE Standard 1149.1 requires the EXTEST, SAMPLE/PRELOAD, and BYPASS instructions. IDCODE, CLAMP, and HIGHZ are optional standard instructions that the MCF5307 implementation supports and are described in the IEEE Standard 1149.1.

19.4.2 IDCODE Register

The MCF5307 includes an IEEE Standard 1149.1-compliant JTAG identification register, IDCODE, which is read by the MCF5307 JTAG instruction encoded as octal 1.

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
Version Number (0000 forH55J, 0001 for J20C)				0	1	0	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	1	0	1

Table 19-3 describes IDCODE bit assignments.

Table 19-3. IDCODE Bit Assignments

Bits	Description
31–28	Version number. Indicates the revision number of the MCF5307
27–22	Design center. Indicates the ColdFire design center
21–12	Device number. Indicates an MCF5307
11–1	Indicates the reduced JEDEC ID for Motorola. Joint Electron Device Engineering Council (JEDEC) Publication 106-A and Chapter 11 of the IEEE Standard 1149.1 give more information on this field.
0	Identifies this as the JTAG IDCODE register (and not the bypass register) according to the IEEE Standard 1149.1

19.4.3 JTAG Boundary-Scan Register

The MCF5307 model includes an IEEE Standard 1149.1-compliant boundary-scan register connected between TDI and TDO when the EXTEST or SAMPLE/PRELOAD instructions are selected. This register captures signal data on the input pins, forces fixed values on the output pins, and selects the direction and drive characteristics (a logic value or high impedance) of the bidirectional and three-state pins. Table 19-4 shows MCF5307 boundary-scan register bits.

Table 19-4. Boundary-Scan Bit Definitions

Bit	Cell Type	Pin Cell	Pin Type	Bit	Cell Type	Pin Cell	Pin Type
0	O.Ctl	PP0 enable	—	120	O.Pin	$\overline{BE0}$	O
1	O.Pin	PP0	I/O	121	O.Pin	SCKE	O
2	I.Pin	PP0	I/O	122	O.Pin	$\overline{SCAS}$	O
3	IO.Ctl	PP1 enable	—	123	O.Pin	$\overline{SRAS}$	O
4	O.Pin	PP1	I/O	124	O.Pin	$\overline{DRAMW}$	O
5	I.Pin	PP1	I/O	125	O.Pin	$\overline{CAS3}$	O
6	IO.Ctl	PP2 enable	—	126	O.Pin	$\overline{CAS2}$	O
7	O.Pin	PP2	I/O	127	O.Pin	$\overline{CAS1}$	O
8	I.Pin	PP2	I/O	128	O.Pin	$\overline{CAS0}$	O
9	IO.Ctl	PP3 enable	—	129	O.Pin	$\overline{RAS1}$	O
10	O.Pin	PP3	I/O	130	O.Pin	$\overline{RAS0}$	O
11	I.Pin	PP3	I/O	131	I.Pin	TIN1	I
12	IO.Ctl	PP4 enable	—	132	I.Pin	TIN0	I
13	O.Pin	PP4	I/O	133	O.Pin	TOUT0	O
14	I.Pin	PP4	I/O	134	O.Pin	TOUT1	O
15	IO.Ctl	PP5 enable	—	135	I.Pin	$\overline{BG}$	I
16	O.Pin	PP5	I/O	136	O.Pin	$\overline{BD}$	O
17	I.Pin	PP5	I/O	137	O.Pin	$\overline{BR}$	O
18	IO.Ctl	PP6 enable	—	138	I.Pin	$\overline{IRQ1}$	I
19	O.Pin	PP6	I/O	139	I.Pin	$\overline{IRQ3}$	I
20	I.Pin	PP6	I/O	140	I.Pin	$\overline{IRQ5}$	I
21	IO.Ctl	PP7 enable	—	141	I.Pin	$\overline{IRQ7}$	I
22	O.Pin	PP7	I/O	142	I.Pin	$\overline{RSTI}$	I
23	I.Pin	PP7	I/O	143	O.Pin	$\overline{TS}$	I/O
24	O.Pin	PST3	O	144	I.Pin	$\overline{TS}$	I/O
25	O.Pin	PST2	O	145	IO.Ctl	$\overline{TA}$ enable	—
26	O.Pin	PST1	O	146	O.Pin	$\overline{TA}$	I/O
27	O.Pin	PST0	O	147	I.Pin	$\overline{TA}$	I/O
28	O.Pin	DDATA3	O	148	O.Pin	R/$\overline{W}$	I/O

Table 19-4. Boundary-Scan Bit Definitions

Bit	Cell Type	Pin Cell	Pin Type	Bit	Cell Type	Pin Cell	Pin Type
29	O.Pin	DDATA2	O	149	I.Pin	R/$\overline{W}$	I/O
30	O.Pin	DDATA1	O	150	O.Pin	$\overline{AS}$	I/O
31	O.Pin	DDATA0	O	151	I.Pin	$\overline{AS}$	I/O
32	O.Pin	PSTCLK	O	152	O.Pin	$\overline{CS7}$	O
33	I.Pin	CLKIN	I	153	O.Pin	$\overline{CS6}$	O
34	IO.Ctl	$\overline{RSTO}$ enable	—	154	O.Pin	$\overline{CS5}$	O
35	O.Pin	$\overline{RSTO}$	I/O	155	O.Pin	$\overline{CS4}$	O
36	I.Pin	$\overline{RSTO}$	I/O	156	O.Pin	$\overline{CS3}$	O
37	O.Pin	BCLKO	O	157	O.Pin	$\overline{CS2}$	O
38	I.Pin	EDGESEL	I	158	O.Pin	$\overline{CS1}$	O
39	O.Pin	TXD0	O	159	O.Pin	$\overline{CS0}$	O
40	I.Pin	RXD0	I	160	O.Pin	$\overline{OE}$	O
41	O.Pin	RTS0	O	161	O.Pin	SIZ1	I/O
42	I.Pin	CTS0	I	162	I.Pin	SIZ1	I/O
43	O.Pin	TXD1	O	163	O.Pin	SIZ0	I/O
44	I.Pin	RXD1	I	164	I.Pin	SIZ0	I/O
45	O.Pin	RTS1	O	165	IO.Ctl	PP15 enable	—
46	I.Pin	CTS1	I	166	I.Pin	PP15	I/O
47	I.Pin	$\overline{HIZ}$	I	167	O.Pin	PP15	I/O
48	IO.Ctl	Data enable	—	168	IO.Ctl	PP14 enable	—
49	O.Pin	D0	I/O	169	I.Pin	PP14	I/O
50	I.Pin	D0	I/O	170	O.Pin	PP14	I/O
51	O.Pin	D1	I/O	171	IO.Ctl	PP13 enable	—
52	I.Pin	D1	I/O	172	I.Pin	PP13	I/O
53	O.Pin	D2	I/O	173	O.Pin	PP13	I/O
54	I.Pin	D2	I/O	174	IO.Ctl	PP12 enable	—
55	O.Pin	D3	I/O	175	I.Pin	PP12	I/O
56	I.Pin	D3	I/O	176	O.Pin	PP12	I/O
57	O.Pin	D4	I/O	177	IO.Ctl	PP11 enable	—
58	I.Pin	D4	I/O	178	I.Pin	PP11	I/O
59	O.Pin	D5	I/O	179	O.Pin	PP11	I/O
60	I.Pin	D5	I/O	180	IO.Ctl	PP10 enable	—
61	O.Pin	D6	I/O	181	I.Pin	PP10	I/O
62	I.Pin	D6	I/O	182	O.Pin	PP10	I/O
63	O.Pin	D7	I/O	183	IO.Ctl	PP9 enable	—
64	I.Pin	D7	I/O	184	I.Pin	PP9	I/O

Table 19-4. Boundary-Scan Bit Definitions

Bit	Cell Type	Pin Cell	Pin Type	Bit	Cell Type	Pin Cell	Pin Type
65	O.Pin	D8	I/O	185	O.Pin	PP9	I/O
66	I.Pin	D8	I/O	186	IO.Ctl	PP8 enable	—
67	O.Pin	D9	I/O	187	I.Pin	PP8	I/O
68	I.Pin	D9	I/O	188	O.Pin	PP8	I/O
69	O.Pin	D10	I/O	189	IO.Ctl	$\overline{TS}$/R/$\overline{W}$/SIZ enable	—
70	I.Pin	D10	I/O	190	IO.Ctl	Address enable	—
71	O.Pin	D11	I/O	191	O.Pin	A23	I/O
72	I.Pin	D11	I/O	192	I.Pin	A23	I/O
73	O.Pin	D12	I/O	193	O.Pin	A22	I/O
74	I.Pin	D12	I/O	194	I.Pin	A22	I/O
75	O.Pin	D13	I/O	195	O.Pin	A21	I/O
76	I.Pin	D13	I/O	196	I.Pin	A21	I/O
77	O.Pin	D14	I/O	197	O.Pin	A20	I/O
78	I.Pin	D14	I/O	198	I.Pin	A20	I/O
79	O.Pin	D15	I/O	199	O.Pin	A19	I/O
80	I.Pin	D15	I/O	200	I.Pin	A19	I/O
81	O.Pin	D16	I/O	201	O.Pin	A18	I/O
82	I.Pin	D16	I/O	202	I.Pin	A18	I/O
83	O.Pin	D17	I/O	203	O.Pin	A17	I/O
84	I.Pin	D17	I/O	204	I.Pin	A17	I/O
85	O.Pin	D18	I/O	205	O.Pin	A16	I/O
86	I.Pin	D18	I/O	206	I.Pin	A16	I/O
87	O.Pin	D19	I/O	207	O.Pin	A15	I/O
88	I.Pin	D19	I/O	208	I.Pin	A15	I/O
89	O.Pin	D20	I/O	209	O.Pin	A14	I/O
90	I.Pin	D20	I/O	210	I.Pin	A14	I/O
91	O.Pin	D21	I/O	211	O.Pin	A13	I/O
92	I.Pin	D21	I/O	212	I.Pin	A13	I/O
93	O.Pin	D22	I/O	213	O.Pin	A12	I/O
94	I.Pin	D22	I/O	214	I.Pin	A12	I/O
95	O.Pin	D23	I/O	215	O.Pin	A11	I/O
96	I.Pin	D23	I/O	216	I.Pin	A11	I/O
97	O.Pin	D24	I/O	217	O.Pin	A10	I/O
98	I.Pin	D24	I/O	218	I.Pin	A10	I/O
99	O.Pin	D25	I/O	219	O.Pin	A9	I/O
100	I.Pin	D25	I/O	220	I.Pin	A9	I/O

Table 19-4. Boundary-Scan Bit Definitions

Bit	Cell Type	Pin Cell	Pin Type
101	O.Pin	D26	I/O
102	I.Pin	D26	I/O
103	O.Pin	D27	I/O
104	I.Pin	D27	I/O
105	O.Pin	D28	I/O
106	I.Pin	D28	I/O
107	O.Pin	D29	I/O
108	I.Pin	D29	I/O
109	O.Pin	D30	I/O
110	I.Pin	D30	I/O
111	O.Pin	D31	I/O
112	I.Pin	D31	I/O
113	O.Pin	SDA	OD
114	I.Pin	SDA	I
115	O.Pin	SCL	OD
116	I.Pin	SCL	I
117	O.Pin	$\overline{BE3}$	O
118	O.Pin	$\overline{BE2}$	O
119	O.Pin	$\overline{BE1}$	O
221	O.Pin	A8	I/O
222	I.Pin	A8	I/O
223	O.Pin	A7	I/O
224	I.Pin	A7	I/O
225	O.Pin	A6	I/O
226	I.Pin	A6	I/O
227	O.Pin	A5	I/O
228	I.Pin	A5	I/O
229	O.Pin	A4	I/O
230	I.Pin	A4	I/O
231	O.Pin	A3	I/O
232	I.Pin	A3	I/O
233	O.Pin	A2	I/O
234	I.Pin	A2	I/O
235	O.Pin	A1	I/O
236	I.Pin	A1	I/O
237	O.Pin	A0	I/O
238	I.Pin	A0	I/O

19.4.4 JTAG Bypass Register

The IEEE Standard 1149.1-compliant bypass register creates a single-bit shift register path from TDI to the bypass register to TDO when the BYPASS instruction is selected.

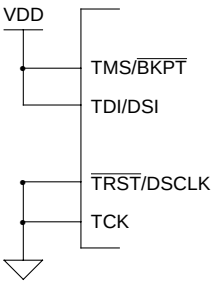
19.5 Restrictions

Test logic design is static, so TCK can be stopped in high or low state with no data loss. However, system logic uses a different system clock not internally synchronized to TCK. Operation mixing 1149.1 test logic with system functional logic that uses both clocks must coordinate and synchronize these clocks externally to the MCF5307.

19.6 Disabling IEEE Standard 1149.1 Operation

There are two ways to use the MCF5307 without IEEE Standard 1149.1 test logic being active:

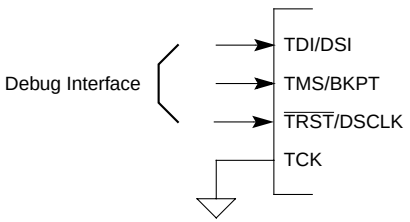
- Nonuse of JTAG test logic by either nontermination (disconnection) or intentionally fixing TAP logic values. The following issues must be addressed if IEEE Standard 1149.1 logic is not to be used when the MCF5307 is assembled onto a board.
  - IEEE Standard 1149.1 test logic must remain transparent and benign to the system logic during functional operation. To ensure that the part enters the test-logic-reset state requires either connecting $\overline{\text{TRST}}$ to logic 0 or connecting TCK to a source that supplies five rising edges and a falling edge after the fifth rising edge. The recommended solution is to connect $\overline{\text{TRST}}$ to logic 0.
  - TCK has no internal pull-up as is required on TMS, TDI, and $\overline{\text{TRST}}$; therefore, it must be terminated to preclude mid-level input values. Figure 19-4 shows pin values recommended for disabling JTAG with the MCF5307 in JTAG mode.



Note: MTMOD0 high allows JTAG mode.

Figure 19-4. Disabling JTAG in JTAG Mode

- Disabling JTAG test logic by holding MTMOD0 low during reset (debug mode). This allows the IEEE Standard 1149.1 test controller to enter test-logic-reset state when $\overline{\text{TRST}}$ is internally asserted to the controller. TAP pins function as debug mode pins. In JTAG mode, inputs TDI/DSI, TMS/$\overline{\text{BKPT}}$, and $\overline{\text{TRST}}$/DSCLK have internal pull-ups enabled. Figure 19-5 shows pin values recommended for disabling JTAG in debug mode.



Note: MTMOD0 low prohibits JTAG.

Figure 19-5. Disabling JTAG in Debug Mode

19.7 Obtaining the IEEE Standard 1149.1

The IEEE Standard 1149.1 JTAG specification is a copyrighted document and must be obtained directly from the IEEE:

IEEE Standards Department
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

http://stdsbbs.ieee.org/

FAX: 908-981-9667
Information: 908-981-0060 or 1-800-678-4333

Chapter 20
Electrical Specifications

This chapter describes the AC and DC electrical specifications and thermal characteristics for the MCF5307. Note that this information was correct at the time this book was published. As process technologies improve, there is a likelihood that this information may change. To confirm that this is the latest information, see Motorola's ColdFire webpage, http://www.motorola.com/coldfire.

20.1 General Parameters

Table 20-1 lists maximum and minimum ratings for supply and operating voltages and storage temperature. Operating outside of these ranges may cause erratic behavior or damage to the processor.

Table 20-1. Absolute Maximum Ratings

Rating	Symbol	Value	Units
Supply voltage	V_{cc}	-0.3 to +4.0	V
Maximum operating voltage	V_{cc}	+3.6	V
Minimum operating voltage	V_{cc}	+3.0	V
Input voltage	V_{in}	-0.5 to +5.5	V
Storage temperature range	T_{stg}	-55 to +150	^{o}C

Table 20-2 lists junction and ambient operating temperatures.

Table 20-2. Operating Temperatures

Characteristic	Symbol	Value	Units
Maximum operating junction temperature	T_j	105	^{o}C
Maximum operating ambient temperature	T_{Amax}	70 [1]	^{o}C
Minimum operating ambient temperature	T_{Amin}	0	^{o}C

[1] This published maximum operating ambient temperature should be used only as a system design guideline. All device operating parameters are guaranteed only when the junction temperature lies within the specified range.

Table 20-3 lists DC electrical operating temperatures. This table is based on an operating

voltage of Vcc = 3.3 Vdc ± 0.3 Vdc.

Table 20-3. DC Electrical Specifications

Characteristic	Symbol	Min	Max	Units
Operation voltage range	V_{cc}	3.0	3.6	V
Input high voltage	V_{IH}	2.0	3.6	V
Input low voltage	V_{IL}	-0.5	0.8	V
Input signal undershoot	—	—	0.8	V
Input signal overshoot	—	—	0.8	V
Input leakage current @ 0.5/2.4 V during normal operation	I_{in}	—	20	µA
High impedance (three-state) leakage current @ 0.5/2.4 V during normal operation	I_{TSI}	—	20	µA
Signal low input current, V_{IL} = 0.8 V [1]	I_{IL}	0	1	mA
Signal high input current, V_{IH} = 2.0 V [1]	I_{IH}	0	1	mA
Output high voltage I_{OH} = 6 mA [2], 12 mA [3]	V_{OH}	2.4	—	V
Output low voltage I_{OL} = 6 mA [2], 12 mA [3]	V_{OL}	—	0.5	V
Load capacitance (all outputs)	C_L	—	50	pF
Capacitance [4], V_{in} = 0 V, f = 1 MHz	C_{IN}	—	10	pF

1 $\overline{\text{BKPT}}$/TMS, DSI/TDI, DSCLK/$\overline{\text{TRST}}$

2 D[31:0], A[23:0], PP[15:0],$\overline{\text{TS}}$, $\overline{\text{TA}}$, SIZ[1:0], R/$\overline{\text{W}}$, $\overline{\text{BR}}$, $\overline{\text{BD}}$, $\overline{\text{RSTO}}$, $\overline{\text{AS}}$, $\overline{\text{CS}}$[7:0], $\overline{\text{BE}}$[3:0], $\overline{\text{OE}}$, PSTCLK, PST[3:0], DDATA[3:0], DSO, TOUT[1:0], SCL, SDA, $\overline{\text{RTS}}$[1:0], TXD[1:0]

3 BCLKO, $\overline{\text{RAS}}$[1:0], $\overline{\text{CAS}}$[3:0], $\overline{\text{DRAMW}}$, SCKE, $\overline{\text{SRAS}}$, $\overline{\text{SCAS}}$

4 Capacitance C_{IN} is periodically sampled rather than 100% tested.

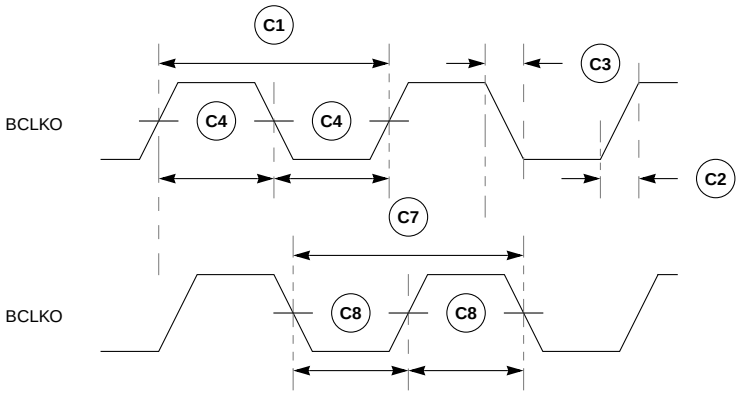
20.2 Clock Timing Specifications

Table 20-4 lists specifications for the clock timing parameters shown in Figure 20-1 and Figure 20-2.

Table 20-4. Clock Timing Specification

Num	Characteristic	66 MHz		90 MHz		Units
		Min	Max	Min	Max	
C1	CLKIN cycle time	30	—	22	—	nS
C2	CLKIN rise time (0.5V to 2.4 V)	—	5	—	5	nS
C3	CLKIN fall time (2.4V to 0.5 V)	—	5	—	5	nS
C4	CLKIN duty cycle (at 1.5 V)	40	60	40	60	%
C5	PSTCLK cycle time	15	—	11	—	nS
C6	PSTCLK duty cycle (at 1.5 V)	40	60	40	60	%
C7	BCLKO cycle time	30	—	22	—	nS
C8	BCLKO duty cycle (at 1.5 V)	45	55	45	55	%

Figure 20-1 shows timings for the parameters listed in Table 20-4.



Note: Input and output AC timing specifications are measured to BCLKO with a 50-pF load capacitance (not including pin capacitance).

Figure 20-1. Clock Timing

Figure 20-2 shows PSTCLK timings for parameters listed in Table 20-4.

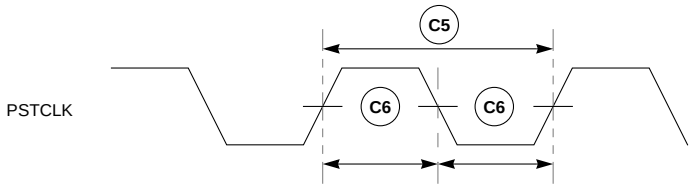


Figure 20-2. PSTCLK Timing

20.3 Input/Output AC Timing Specifications

Table 20-5 lists specifications for parameters shown in Figure 20-3 and Figure 20-4. Note that inputs $\overline{\text{IRQ}}$[7,5,3,1], $\overline{\text{BKPT}}$, and $\overline{\text{AS}}$ are synchronized internally; that is, the logic level is validated if the value does not change for two consecutive rising BCLKO edges. Setup and hold times must be met only if recognition on a particular clock edge is required.

Table 20-5. Input AC Timing Specification

Num	Characteristic	66 MHz		90 MHz		Units
		Min	Max	Min	Max	
B1 [1]	Valid to BCLKO rising (setup)	7.5	—	5.5	—	nS
B2 [1]	BCLKO rising to invalid (hold)	3	—	2	—	nS
B3 [2]	Valid to BCLKO falling (setup)	7.5	—	5.5	—	nS
B4 [2]	BCLKO falling to invalid (hold)	3	—	2	—	nS

Table 20-5. Input AC Timing Specification

Num	Characteristic	66 MHz		90 MHz		Units
		Min	Max	Min	Max	
B5 [3]	BCLKO to input high impedance	—	2	—	2	Bus clock
B6	BCLKO to EDGESEL delay	0	7.5	0	5.5	nS

1 Inputs: $\overline{BG}$, $\overline{TA}$, A[23:0], PP[15:0], SIZ[1:0], R/$\overline{W}$, $\overline{TS}$, EDGESEL, D[31:0], $\overline{IRQ}$[7,5,3,1], and $\overline{BKPT}$

2 Inputs: $\overline{AS}$

3 Inputs: D[31:0]

Table 20-6 lists specifications for timings in Figure 20-3, Figure 20-4, and Figure 20-10. Although output signals that share a specification number have approximately the same timing, due to loading differences, they do not necessarily change at the same time. However, they have similar timings; that is, minimum and maximum times are not mixed.

Table 20-6. Output AC Timing Specification

Num	Characteristic	66 MHz		90 MHz		Units
		Min	Max	Min	Max	
B10 [1,2,3]	BCLKO rising to valid	—	15	—	11	nS
B11 [1,2,3,4]	BCLKO rising to invalid (hold)	1	—	1	—	nS
B11a [1,2,3,5]	BCLKO rising to invalid (hold)	0.5	—	0.5	—	
B12 [6,7]	BCLKO to high impedance (three-state)	—	15	—	11	nS
B13 [8,2,3]	BCLKO rising to valid	—	15	—	11	nS
B14 [8,2,3]	BCLKO rising to invalid (hold)	3	—	2	—	nS
B15 [2,3]	EDGESEL to valid	—	18.5	—	13.5	nS
B16 [2,3]	EDGESEL to invalid (hold)	3	—	2	—	nS
H1	$\overline{HIZ}$ to high impedance	—	60	—	60	nS
H2	$\overline{HIZ}$ to low Impedance	—	60	—	60	nS

1 Outputs that only change on rising edge of BCLKO: $\overline{RSTO}$, $\overline{TS}$, $\overline{BR}$, $\overline{BD}$, $\overline{TA}$, R/$\overline{W}$, SIZ[1:0], PP[7:0] (and PP[15:8] when configured as parallel port outputs).

2 Outputs that can change on either BCLKO edge depending only upon EDGESEL: D[31:0], A[23:0], SCKE, $\overline{SRAS}$, $\overline{SCAS}$, $\overline{DRAMW}$ (and PP[15:8] when individually configured as address outputs).

3 Outputs that can change on either BCLKO edge depending only upon EDGESEL: D[31:0], A[23:0], SCKE, $\overline{SRAS}$, $\overline{SCAS}$, $\overline{DRAMW}$ (and PP[15:8] when individually configured as address outputs).

4 Applies to D[31:0], A[23:0], $\overline{RSTO}$, $\overline{TS}$, $\overline{BR}$, $\overline{BD}$, $\overline{TA}$, R/$\overline{W}$, SIZ[1:0], PP[7:0] (and PP[15:8] when configured as parallel port outputs).

5 Applies to $\overline{RAS}$[1:0], $\overline{CAS}$[1:0], SCKE, $\overline{SRAS}$, $\overline{SCAS}$, $\overline{DRAMW}$

6 High Impedance (three-state): D[31:0]

7 Outputs that transition to high-impedance due to bus arbitration: A[23:0], R/$\overline{W}$, SIZ[1:0], $\overline{TS}$, $\overline{AS}$, $\overline{TA}$, (and PP[15:0] when individually configured as address outputs)

8 Outputs that only change on falling edge of BCLKO: $\overline{AS}$, $\overline{CS}$[7:0], $\overline{BE}$[3:0], $\overline{OE}$

Note that these figures show two representative bus operations and do not attempt to show all cases. For explanations of the states, S0–S5, see Section 18.4, “Data Transfer

Operation." Note that Figure 20-4 does not show all signals that apply to each timing specification. See the previous tables for a complete listing.

Figure 20-3 shows AC timings for normal read and write bus cycles.

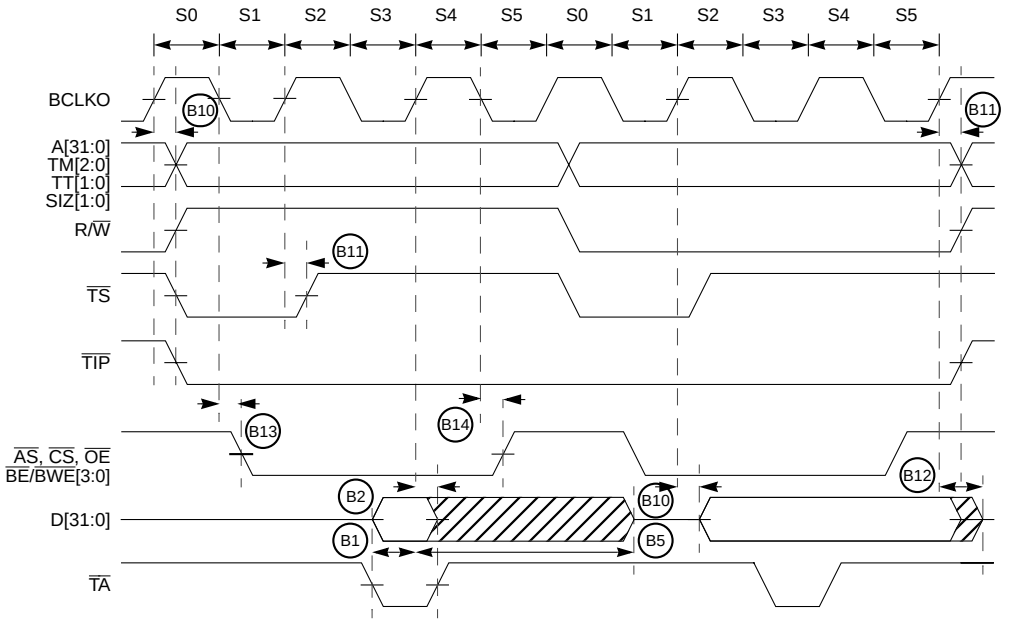


Figure 20-3. AC Timings—Normal Read and Write Bus Cycles

Figure 20-4 shows timings for a read cycle with EDGESEL tied to buffered BCLKO.

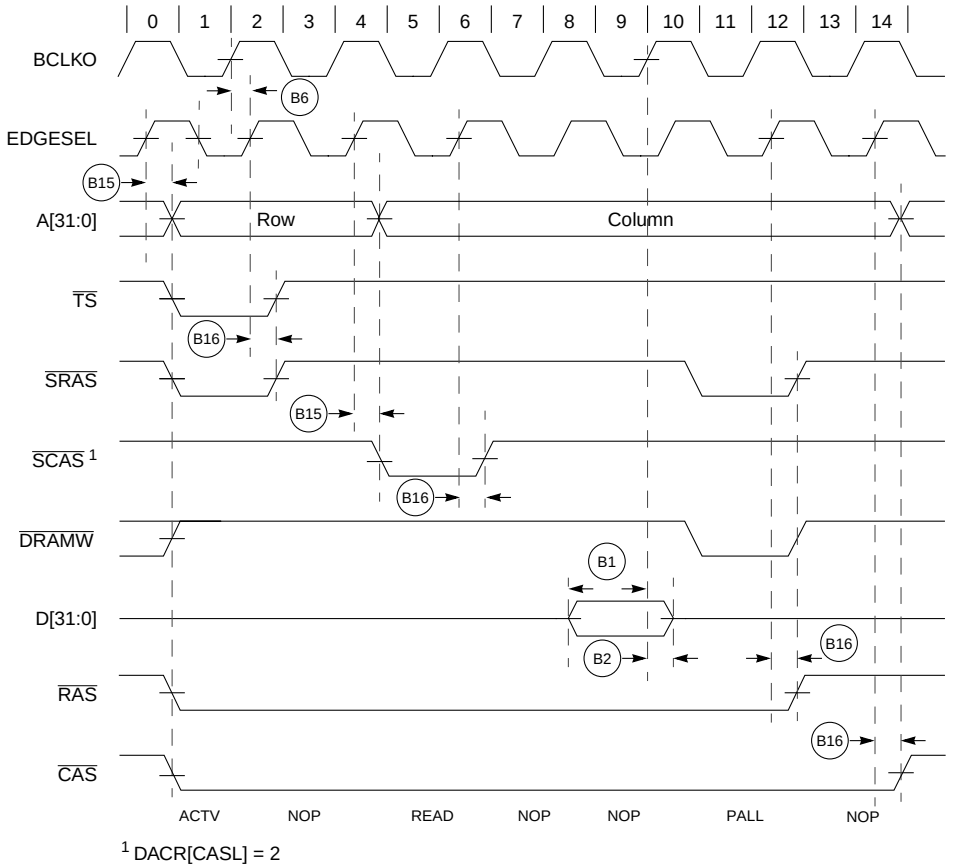


Figure 20-4. SDRAM Read Cycle with EDGESEL Tied to Buffered BCLKO

Figure 20-5 shows an SDRAM write cycle with EDGESEL tied to buffered BCLKO.

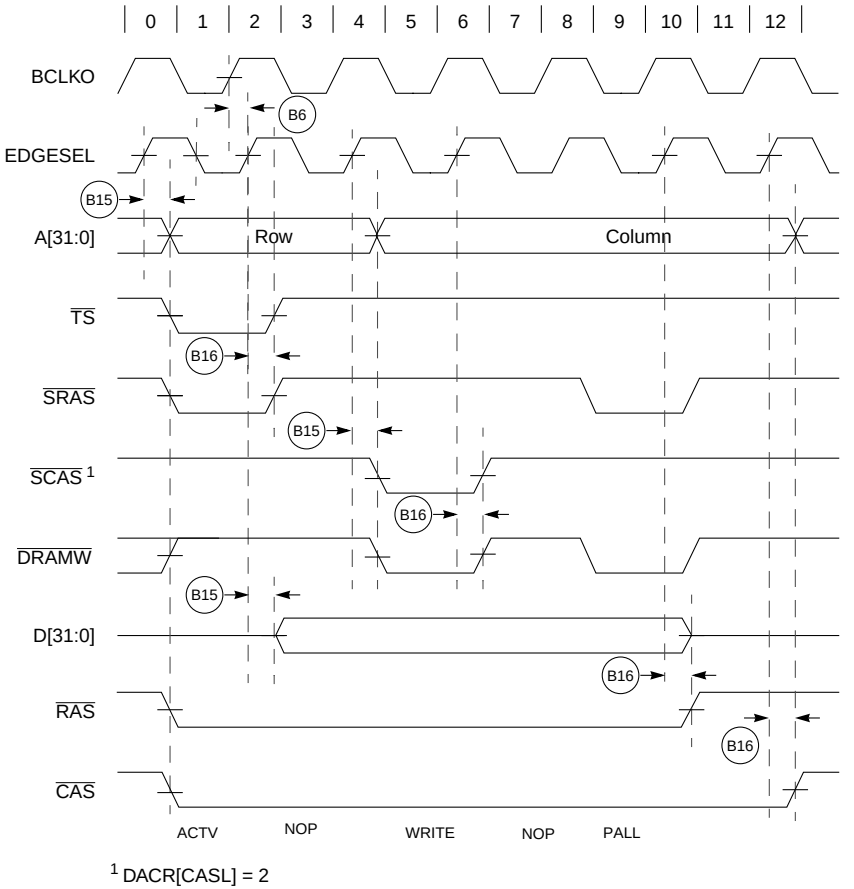


Figure 20-5. SDRAM Write Cycle with EDGESEL Tied to Buffered BCLKO

Figure 20-6 shows an SDRAM read cycle with EDGESEL tied high.

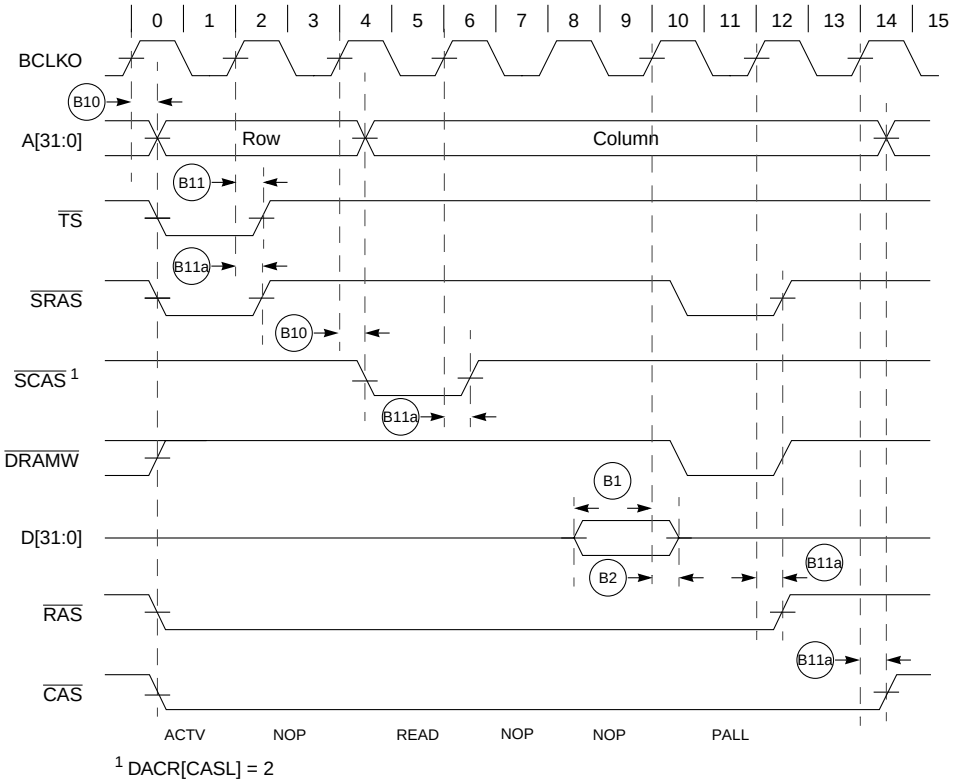


Figure 20-6. SDRAM Read Cycle with EDGESEL Tied High

Figure 20-7 shows an SDRAM write cycle with EDGESEL tied high.

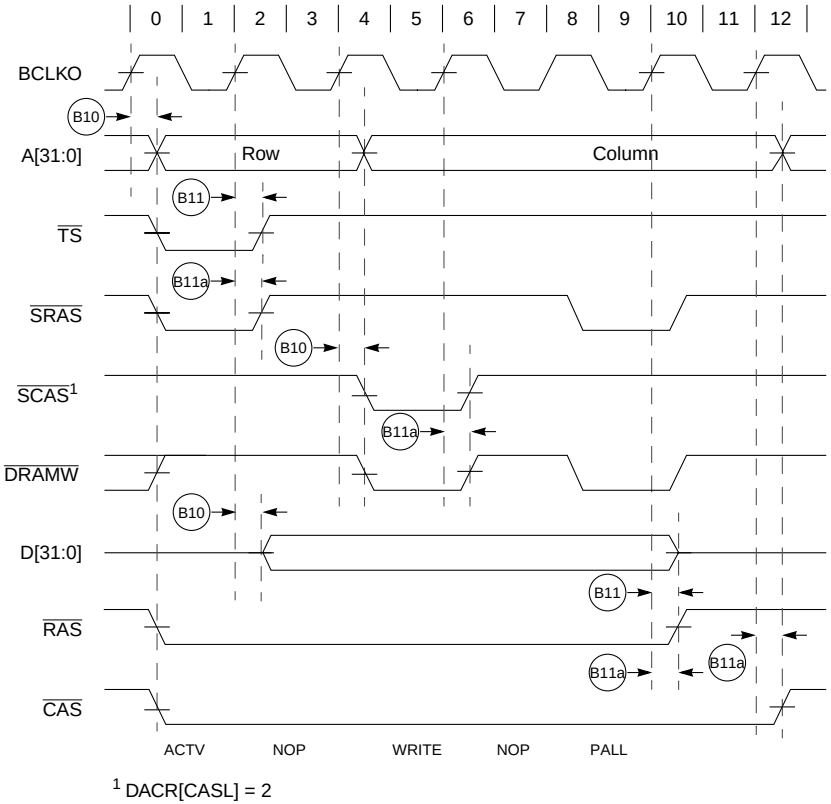


Figure 20-7. SDRAM Write Cycle with EDGESEL Tied High

Figure 20-8 shows an SDRAM read cycle with EDGESEL tied low.

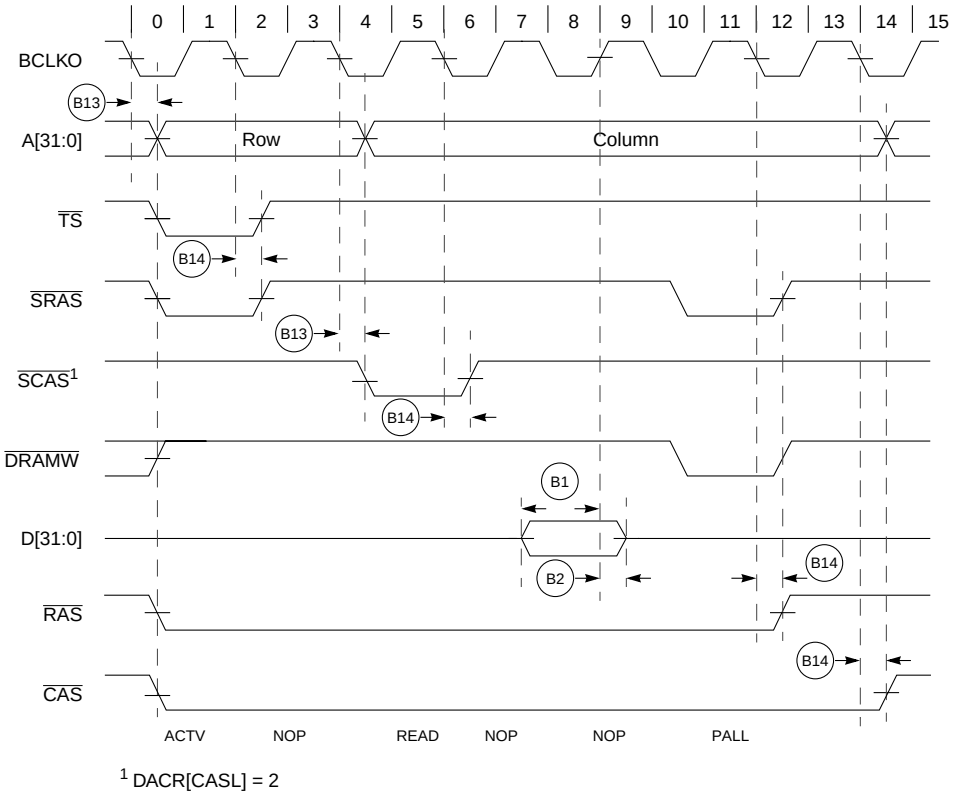


Figure 20-8. SDRAM Read Cycle with EDGESEL Tied Low

Figure 20-9 shows an SDRAM write cycle with EDGESEL tied low.

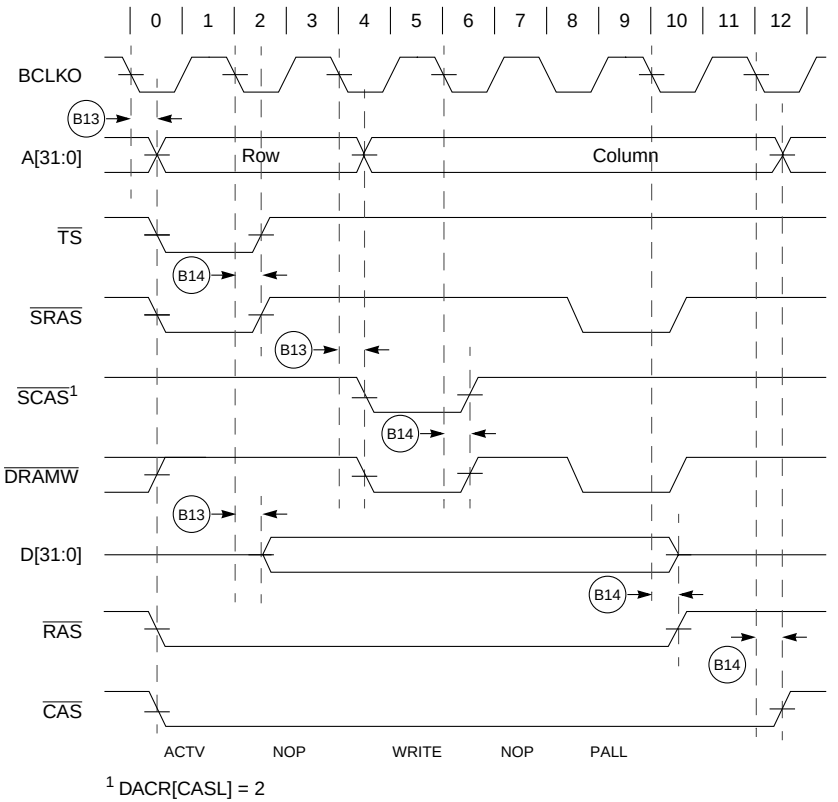


Figure 20-9. SDRAM Write Cycle with EDGESEL Tied Low

Figure 20-10 shows AC timing showing high impedance.

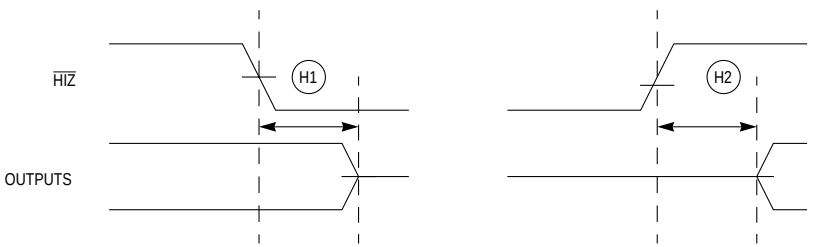


Figure 20-10. AC Output Timing—High Impedance

20.4 Reset Timing Specifications

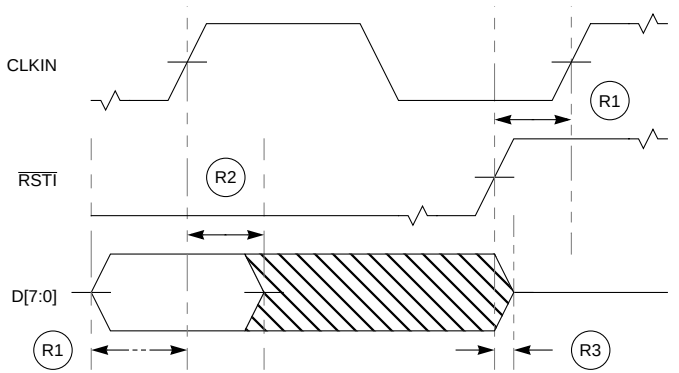
Table 20-7 lists specifications for the reset timing parameters shown in Figure 20-11.

Table 20-7. Reset Timing Specification

Num	Characteristic	66 MHz		90 MHz		Units
		Min	Max	Min	Max	
R1 [1]	Valid to CLKIN (setup)	7.5	—	5.5	—	nS
R2	CLKIN to invalid (hold)	3	—	2	—	nS
R3	$\overline{\text{RSTI}}$ to invalid (hold)	3	—	2	—	nS

1 $\overline{\text{RSTI}}$ and D[7:0] are synchronized internally. Setup and hold times must be met only if recognition on a particular clock is required.

Figure 20-11 shows reset timing for the values in Table 20-7.



Note: Mode selects are registered on the rising CLKIN edge before the cycle in which $\overline{\text{RSTI}}$ is recognized as being negated.

Figure 20-11. Reset Timing

20.5 Debug AC Timing Specifications

Table 20-8 lists specifications for the debug AC timing parameters shown in Figure 20-13.

Table 20-8. Debug AC Timing Specification

Num	Characteristic	66 MHz		90 MHz		Units
		Min	Max	Min	Max	
D1	PST, DDATA to PSTCLK setup	7.5		5.5		nS
D2	PSTCLK to PST, DDATA hold	7.5		5.5		nS
D3	DSI-to-DSCLK setup	1		1		PSTCLKs

Table 20-8. Debug AC Timing Specification

Num	Characteristic	66 MHz		90 MHz		Units
		Min	Max	Min	Max	
D4 [1]	DSCLK-to-DSO hold	4		4		PSTCLKs
D5	DSCLK cycle time	5		5		PSTCLKs

[1] DSCLK and DSI are synchronized internally. D4 is measured from the synchronized DSCLK input relative to the rising edge of PSTCLK.

Figure 20-12 shows real-time trace timing for the values in Table 20-8.

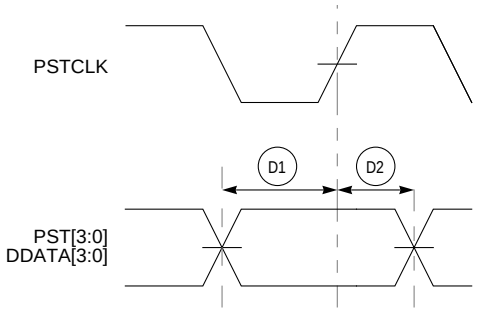


Figure 20-12. Real-Time Trace AC Timing

Figure 20-13 shows BDM serial port AC timing for the values in Table 20-8.

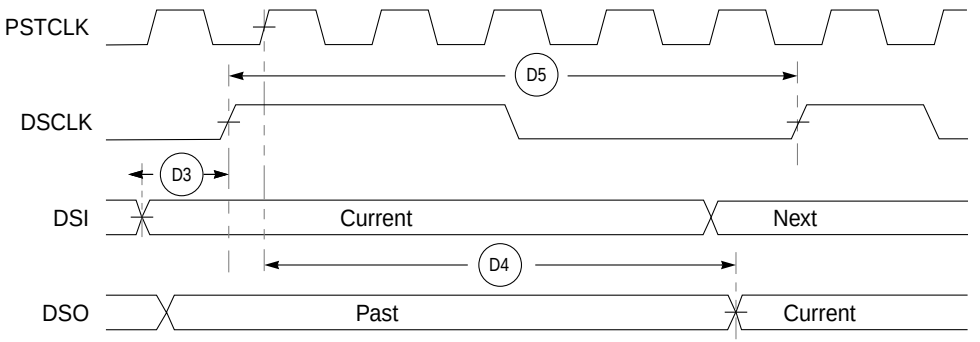


Figure 20-13. BDM Serial Port AC Timing

20.6 Timer Module AC Timing Specifications

Table 20-9 lists specifications for timer module AC timing parameters shown in Figure 20-14.

Figure 20-14 shows timings for Table 20-9.

Table 20-9. Timer Module AC Timing Specification

Num	Characteristic	66 MHz		90 MHz		Units
		Min	Max	Min	Max	
T1	TIN cycle time	3	—	3	—	Bus clocks
T2	TIN valid to BCLKO (input setup)	7.5	—	5.5	—	nS
T3	BCLKO to TIN invalid (input hold)	3	—	2	—	nS
T4	BCLKO to TOUT valid (output valid)	—	15	—	11	nS
T5	BCLKO to TOUT invalid (output hold)	1.5	—	1.5	—	nS
T6	TIN pulse width	1	—	1	—	Bus clocks
T7	TOUT pulse width	1	—	1	—	Bus clocks

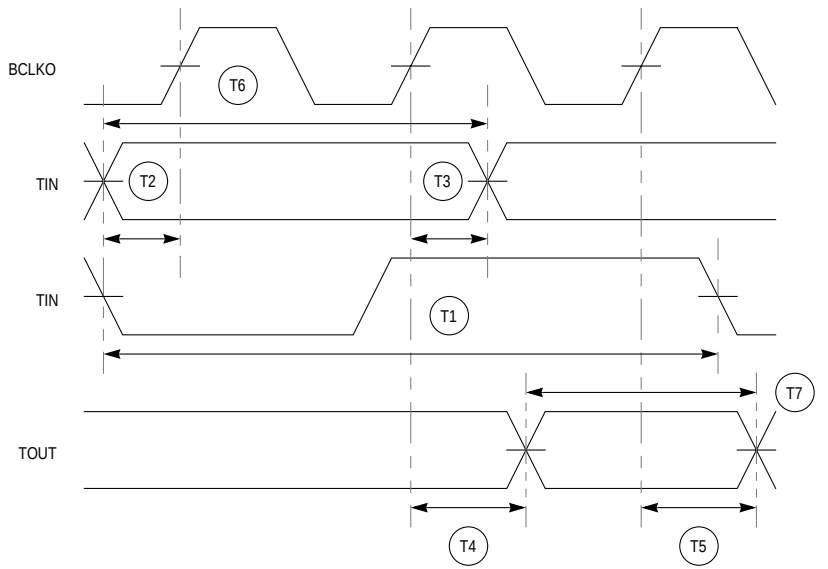


Figure 20-14. Timer Module AC Timing

20.7 I²C Input/Output Timing Specifications

Table 20-10 lists specifications for the I²C input timing parameters shown in Figure 20.8.

Table 20-10. I²C Input Timing Specifications between SCL and SDA

Num	Characteristic	66 MHz		90 MHz		Units
		Min	Max	Min	Max	
I1	Start condition hold time	2	—	2	—	Bus clocks
I2	Clock low period	8	—	8	—	Bus clocks
I3	SCL/SDA rise time (V_{IL} = 0.5 V to V_{IH} = 2.4 V)	—	1	—	1	mS
I4	Data hold time	0	—	0	—	nS
I5	SCL/SDA fall time (V_{IH} = 2.4 V to V_{IL} = 0.5 V)	—	1	—	1	mS
I6	Clock high time	4	—	4	—	Bus clocks
I7	Data setup time	0	—	0	—	nS
I8	Start condition setup time (for repeated start condition only)	2	—	2	—	Bus clocks
I9	Stop condition setup time	2	—	2	—	Bus clocks

Table 20-11 lists specifications for the I²C output timing parameters shown in Figure 20.8.

Table 20-11. I²C Output Timing Specifications between SCL and SDA

Num	Characteristic	66 MHz		90 MHz		Units
		Min	Max	Min	Max	
I1[1]	Start condition hold time	6	—	6	—	Bus clocks
I2 [1]	Clock low period	10	—	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	—	7	—	Bus clocks
I5 [3]	SCL/SDA fall time (V_{IH} = 2.4 V to V_{IL} = 0.5 V)	—	3	—	3	nS
I6 [1]	Clock high time	10	—	10	—	Bus clocks
I7 [1]	Data setup time	2	—	2	—	Bus clocks
I8 [1]	Start condition setup time (for repeated start condition only)	20	—	20	—	Bus clocks
I9 [1]	Stop condition setup time	10	—	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 20-11. 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 20-11 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 20.8 shows timing for the values in Table 20-10 and Table 20-11.

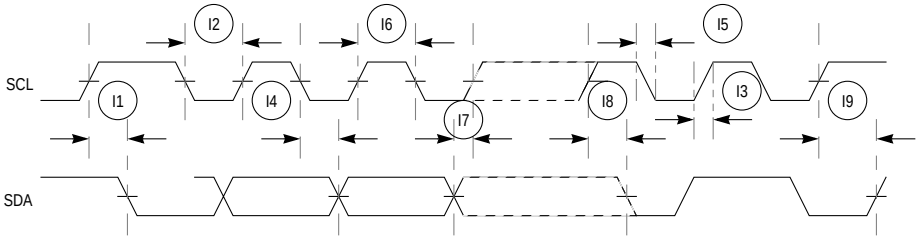


Figure 20-15. I²C Input/Output Timings

20.8 UART Module AC Timing Specifications

Table 20-12 lists specifications for UART module AC timing parameters in Figure 20-16.

Table 20-12. UART Module AC Timing Specifications

Num	Characteristic	66 MHz		90 MHz		Units
		Min	Max	Min	Max	
U1	RXD valid to BCLKO (input setup)	7.5	—	5.5	—	nS
U2	BCLKO to RXD invalid (input hold)	3	—	2	—	nS
U3	$\overline{\text{CTS}}$ valid to BCLKO (input setup)	7.5	—	5.5	—	nS
U4	BCLKO to $\overline{\text{CTS}}$ invalid (input hold)	3	—	2	—	nS
U5	BCLKO to TXD valid (output valid)	—	15	—	11	nS
U6	BCLKO to TXD invalid (output hold)	1.5	—	1.5	—	nS
U7	BCLKO to $\overline{\text{RTS}}$ valid (output valid)	—	15	—	11	nS
U8	BCLKO to $\overline{\text{RTS}}$ invalid (output hold)	1.5	—	1.5	—	nS

Figure 20-16 shows UART timing for the values in Table 20-12.

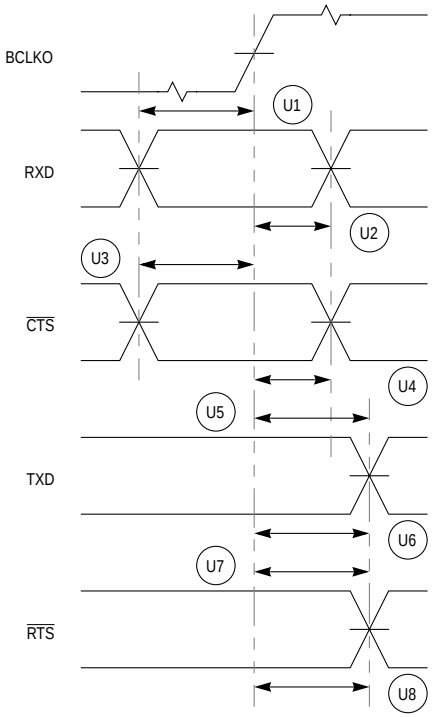


Figure 20-16. UART0/1 Module AC Timing—UART Mode

20.9 Parallel Port (General-Purpose I/O) Timing Specifications

Table 20-13 lists specifications for general-purpose I/O timing parameters in Figure 20-17.

Table 20-13. General-Purpose I/O Port AC Timing Specifications

Num	Characteristic	66 MHz		90 MHz		Units
		Min	Max	Min	Max	
P1	PP valid to BCLKO (input setup)	7.5	—	5.5	—	nS
P2	BCLKO to PP invalid (input hold)	3	—	2	—	nS
P3	BCLKO to PP valid (output valid)	—	15	—	11	nS
P4	BCLKO to PP invalid (output hold)	1	—	1	—	nS

Figure 20-17 shows general-purpose I/O timing.

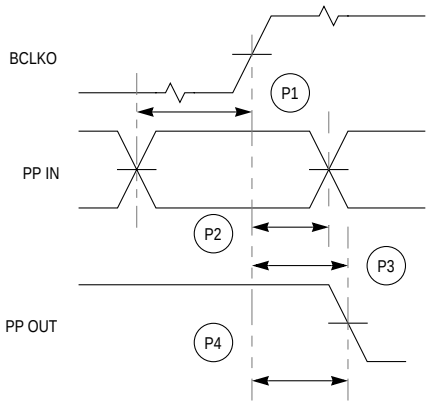


Figure 20-17. General-Purpose I/O Timing

20.10 DMA Timing Specifications

Table 20-14 lists specifications for DMA timing parameters shown in Figure 20-17.

Table 20-14. DMA AC Timing Specifications

Num	Characteristic	66 MHz		90 MHz		Units
		Min	Max	Min	Max	
M1	DREQ valid to BCLKO (input setup)	7.5	—	5.5	—	nS
M2	BCLKO to DREQ invalid (input hold)	3	—	2	—	nS

Figure 20-18 shows DMA AC timing.

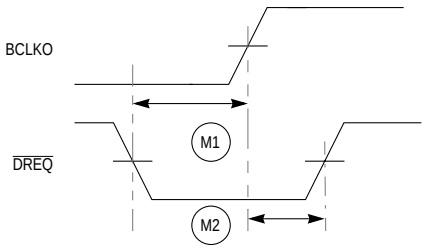


Figure 20-18. DMA Timing

20.11 IEEE 1149.1 (JTAG) AC Timing Specifications

Table 20-15 lists specifications for JTAG AC timing parameters shown in Figure 20-19.

Table 20-15. IEEE 1149.1 (JTAG) AC Timing Specifications

Num	Characteristic	All Frequencies		Units
		Min	Max	
—	TCK frequency of operation	0	10	MHz
J1	TCK cycle time	100	—	nS
J2a	TCK clock pulse high width (measured at 1.5 V)	40	—	nS
J2b	TCK clock pulse low width (measured at 1.5 V)	40	—	nS
J3a	TCK fall time (V_{IH} = 2.4 V to V_{IL} = 0.5V)	—	5	nS
J3b	TCK rise time (V_{IL} = 0.5v to V_{IH} = 2.4V)	—	5	nS
J4	TDI, TMS to TCK rising (input setup)	10	—	nS
J5	TCK rising to TDI, TMS invalid (hold)	15	—	nS
J6	Boundary scan data valid to TCK (setup)	10	—	nS
J7	TCK to boundary-scan data invalid (hold)	15	—	nS
J8	$\overline{\text{TRST}}$ pulse width (asynchronous to clock edges)	15	—	—
J9	TCK falling to TDO valid (signal from driven or three-state)	—	30	nS
J10	TCK falling to TDO high impedance	—	30	nS
J11	TCK falling to boundary scan data valid (signal from driven or three-state)	—	30	nS
J12	TCK falling to boundary scan data high impedance	—	30	nS

Figure 20-19 shows JTAG timing.

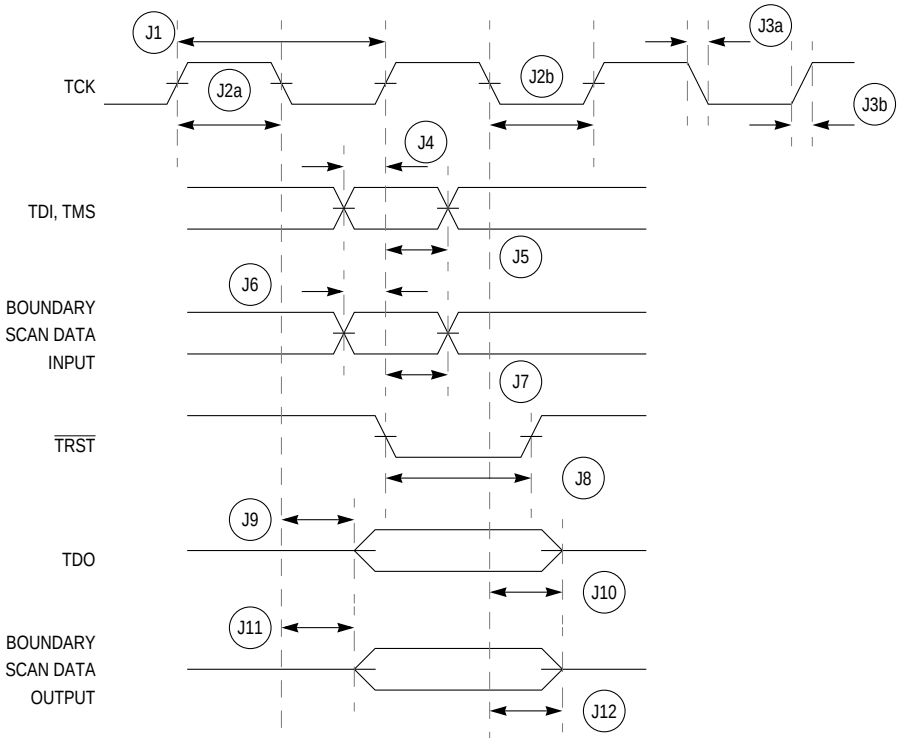


Figure 20-19. IEEE 1149.1 (JTAG) AC Timing

Appendix A
List of Memory Maps

Table A-1. SIM Registers

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x000	Reset status register (RSR) [p. 6-5]	System protection control register (SYPCR) [p. 6-8]	Software watchdog interrupt vector register (SWIVR) [p. 6-9]	Software watchdog service register (SWSR) [p. 6-9]
0x004	Pin assignment register (PAR) [p. 6-10]		Interrupt port assignment register (IRQPAR) [p. 9-7]	Reserved
0x008	PLL control (PLLCR) [p. 7-3]	Reserved		
0x00C	Default bus master park register (MPARK) [p. 6-11]	Reserved		
0x010–0x03C	Reserved			

Table A-2. Interrupt Controller Registers

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
Interrupt Registers [p. 9-3]				
0x040	Interrupt pending register (IPR) [p. 9-6]			
0x044	Interrupt mask register (IMR) [p. 9-6]			
0x048	Reserved			Autovector register (AVR) [p. 9-5]
Interrupt Control Registers (ICRs) [p. 9-3]				
0x04C	Software watchdog timer (ICR0) [p. 6-6]	Timer0 (ICR1) [p. 9-2]	Timer1 (ICR2) [p. 9-3]	I^2C (ICR3) [p. 9-3]
0x050	UART0 (ICR4) [p. 9-3]	UART1 (ICR5) [p. 9-3]	DMA0 (ICR6) [p. 9-3]	DMA1 (ICR7) [p. 9-3]
0x054	DMA2 (ICR8) [p. 9-3]	DMA3 (ICR9) [p. 9-3]	Reserved	

Table A-3. Chip-Select Registers

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x080	Chip-select address register—bank 0 (CSAR0) [p. 10-6]		Reserved[1]	
0x084	Chip-select mask register—bank 0 (CSMR0) [p. 10-6]			
0x088	Reserved[1]		Chip-select control register—bank 0 (CSCR0) [p. 10-8]	
0x08C	Chip-select address register—bank 1 (CSAR1) [p. 10-6]		Reserved[1]	
0x090	Chip-select mask register—bank 1 (CSMR1) [p. 10-6]			
0x094	Reserved[1]		Chip-select control register—bank 1 (CSCR1) [p. 10-8]	
0x098	Chip-select address register—bank 2 (CSAR2) [p. 10-6]		Reserved[1]	
0x09C	Chip-select mask register—bank 2 (CSMR2) [p. 10-6]			
0x0A0	Reserved[1]		Chip-select control register—bank 2 (CSCR2) [p. 10-8	
0x0A4	Chip-select address register—bank 3 (CSAR3) [p. 10-6]		Reserved[1]	
0x0A8	Chip-select mask register—bank 3 (CSMR3) [p. 10-6]			
0x0AC	Reserved[1]		Chip-select control register—bank 3 (CSCR3) [p. 10-8]	
0x0B0	Chip-select address register—bank 4 (CSAR4) [p. 10-6]		Reserved[1]	
0x0B4	Chip-select mask register—bank 4 (CSMR4) [p. 10-6]			
0x0B8	Reserved[1]		Chip-select control register—bank 4 (CSCR4) [p. 10-8]	
0x0BC	Chip-select address register—bank 5 (CSAR5) [p. 10-6]		Reserved[1]	
0x0C0	Chip-select mask register—bank 5 (CSMR5) [p. 10-6]			
0x0C4	Reserved		Chip-select control register—bank 5 (CSCR5) [p. 10-8]	
0x0C8	Chip-select address register—bank 6 (CSAR6) [p. 10-6]		Reserved[1]	
0x0CC	Chip-select mask register—bank 6 (CSMR6) [p. 10-6]			
0x0D0	Reserved[1]		Chip-select control register—bank 6 (CSCR6) [p. 10-8]	
0x0D4	Chip-select address register—bank 7 (CSAR7) [p. 10-6]		Reserved[1]	
0x0D8	Chip-select mask register—bank 7 (CSMR7) [p. 10-6]			
0x0DC	Reserved[1]		Chip-select control register—bank 7 (CSCR7) [p. 10-8	

Table A-3. Chip-Select Registers (Continued)

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x080	Chip-select address register—bank 0 (CSAR0) [p. 10-6]		Reserved[1]	
0x084	Chip-select mask register—bank 0 (CSMR0) [p. 10-6]			
0x088	Reserved[1]		Chip-select control register—bank 0 (CSCR0) [p. 10-8]	
0x08C	Chip-select address register—bank 1 (CSAR1) [p. 10-6]		Reserved[1]	
0x090	Chip-select mask register—bank 1 (CSMR1) [p. 10-6]			
0x094	Reserved[1]		Chip-select control register—bank 1 (CSCR1) [p. 10-8]	
0x098	Chip-select address register—bank 2 (CSAR2) [p. 10-6]		Reserved[1]	
0x09C	Chip-select mask register—bank 2 (CSMR2) [p. 10-6]			
0x0A0	Reserved[1]		Chip-select control register—bank 2 (CSCR2) [p. 10-8]	
0x0A4	Chip-select address register—bank 3 (CSAR3) [p. 10-6]		Reserved[1]	
0x0A8	Chip-select mask register—bank 3 (CSMR3) [p. 10-6]			
0x0AC	Reserved[1]		Chip-select control register—bank 3 (CSCR3) [p. 10-8]	
0x0B0	Chip-select address register—bank 4 (CSAR4) [p. 10-6]		Reserved[1]	
0x0B4	Chip-select mask register—bank 4 (CSMR4) [p. 10-6]			
0x0B8	Reserved[1]		Chip-select control register—bank 4 (CSCR4) [p. 10-8]	

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.

Table A-4. DRAM Controller Registers

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x100	DRAM control register (DCR) [p. 11-3]		Reserved	
0x104	Reserved			
0x108	DRAM address and control register 0 (DACR0) [p. 11-3]			
0x10C	DRAM mask register block 0 (DMR0) [p. 11-3]			
0x110	DRAM address and control register 1 (DACR1) [p. 11-3]			
0x114	DRAM mask register block 1 (DMR1) [p. 11-3]			

Table A-5. General-Purpose Timer Registers

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x140	Timer 0 mode register (TMR0) [p. 13-3]		Reserved	
0x144	Timer 0 reference register (TRR0) [p. 13-4]		Reserved	
0x148	Timer 0 capture register (TCR0) [p. 13-4]		Reserved	
0x14C	Timer 0 counter (TCN0) [p. 13-5]		Reserved	
0x150	Reserved	Timer 0 event register (TER0) [p. 13-5]	Reserved	
0x180	Timer 1 mode register (TMR1) [p. 13-3]		Reserved	
0x184	Timer 1 reference register (TRR1) [p. 13-4]		Reserved	
0x188	Timer 1 capture register (TCR1) [p. 13-4]		Reserved	
0x18C	Timer 1 counter (TCN1) [p. 13-5]		Reserved	
0x190	Reserved	Timer 1 event register (TER1) [p. 13-5]	Reserved	

Table A-6. UART0 Module Programming Model

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x1C0	UART mode registers[1]—(UMR1n) [p. 14-4], (UMR2n) [p. 14-6]	—		
0x1C4	(Read) UART status registers—(USRn) [p. 14-7]	—		
	(Write) UART clock-select register[1]—(UCSRn) [p. 14-8]	—		
0x1C8	(Read) Do not access[2]	—		
	(Write) UART command registers—(UCRn) [p. 14-9]	—		
0x1CC	(UART/Read) UART receiver buffers—(URBn) [p. 14-11]	—		
	(UART/Write) UART transmitter buffers—(UTBn) [p. 14-11]	—		
0x1D0	(Read) UART input port change registers—(UIPCRn) [p. 14-12]	—		

Table A-6. UART0 Module Programming Model (Continued)

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
	(Write) UART auxiliary control registers[1]—(UACRn) [p. 14-12]	—		
0x1D4	(Read) UART interrupt status registers—(UISRn) [p. 14-13]	—		
	(Write) UART interrupt mask registers—(UIMRn) [p. 14-13]	—		
0x1D8	UART divider upper registers—(UDUn) [p. 14-14]	—		
0x1DC	UART divider lower registers—(UDLn) [p. 14-14]	—		
0x1E0–0x1EC	Do not access[2]	—		
0x1F0	UART interrupt vector register—(UIVRn) [p. 14-15]	—		
0x1F4	(Read) UART input port registers—(UIPn) [p. 14-15]	—		
	(Write) Do not access[2]	—		
0x1F8	(Read) Do not access[2]	—		
	(Write) UART output port bit set command registers—(UOP1n[3]) [p. 14-15]	—		
0x1FC	(Read) Do not access[2]	—		
	(Write) UART output port bit reset command registers—(UOP0n[3]) [p. 14-15]	—		

1 UMR1n, UMR2n, and UCSRn should be changed only after the receiver/transmitter is issued a software reset command. That is, if channel operation is not disabled, undesirable results may occur.

2 This address is for factory testing. Reading this location results in undesired effects and possible incorrect transmission or reception of characters. Register contents may also be changed.

3 Address-triggered commands

Table A-7. UART1 Module Programming Model

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x200	UART mode registers[1]—(UMR1n) [p. 14-4], (UMR2n) [p. 14-6]	—		
0x204	(Read) UART status registers—(USRn) [p. 14-7]	—		
	(Write) UART clock-select register[1]—(UCSRn) [p. 14-8]	—		
0x208	(Read) Do not access[2]	—		
	(Write) UART command registers—(UCRn) [p. 14-9]	—		
0x20C	(UART/Read) UART receiver buffers—(URBn) [p. 14-11]	—		
	(UART/Write) UART transmitter buffers—(UTBn) [p. 14-11]	—		
0x210	(Read) UART input port change registers—(UIPCRn) [p. 14-12]	—		
	(Write) UART auxiliary control registers[1]—(UACRn) [p. 14-12]	—		
0x214	(Read) UART interrupt status registers—(UISRn) [p. 14-13]	—		
	(Write) UART interrupt mask registers—(UIMRn) [p. 14-13]	—		
0x218	UART divider upper registers—(UDUn) [p. 14-14]	—		
0x21C	UART divider lower registers—(UDLn) [p. 14-14]	—		

Table A-7. UART1 Module Programming Model (Continued)

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x220–0x22C	Do not access[2]	—		
0x230	UART interrupt vector register—(UIVRn) [p. 14-15]	—		
0x234	(Read) UART input port registers—(UIPn) [p. 14-15]	—		
	(Write) Do not access[2]	—		
0x238	(Read) Do not access[2]	—		
	(Write) UART output port bit set command registers—(UOP1n[3]) [p. 14-15]	—		
0x23C	(Read) Do not access[2]	—		
	(Write) UART output port bit reset command registers—(UOP0n[3]) [p. 14-15]	—		

1 UMR1n, UMR2n, and UCSRn should be changed only after the receiver/transmitter is issued a software reset command. That is, if channel operation is not disabled, undesirable results may occur.

2 This address is for factory testing. Reading this location results in undesired effects and possible incorrect transmission or reception of characters. Register contents may also be changed.

3 Address-triggered commands

Table A-8. Parallel Port Memory Map

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x244	Parallel port data direction register (PADDR) [p. 15-2]		Reserved	
0x248	Parallel port data register (PADAT) [p. 15-2]		Reserved	

Table A-9. I^2C Interface Memory Map

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x280	I^2C address register (IADR) [p. 8-6]	Reserved		
0x284	I^2C frequency divider register (IFDR) [p. 8-7]	Reserved		

Table A-9. I²C Interface Memory Map

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x288	I²C control register (I2CR) [p. 8-8]	Reserved		
0x28C	I²C status register (I2SR) [p. 8-9]	Reserved		
0x290	I²C data I/O register (I2DR) [p. 8-10]	Reserved		

Table A-10. DMA Controller Registers

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x300	Source address register 0 (SAR0) [p. 12-6]			
0x304	Destination address register 0 (DAR0) [p. 12-7]			
0x308	DMA control register 0 (DCR0) [p. 12-8]			
0x30C	Byte count register 0 (BCR24BIT = 0) [1]		Reserved	
0x30C	Reserved	Byte count register 0 (BCR24BIT = 1) [1] (BCR0) [p. 12-7]		
0x310	DMA status register 0 (DSR0) [p. 12-10]	Reserved		
0x314	DMA interrupt vector register 0 (DIVR0) [p. 12-11]	Reserved		
0x340	Source address register 1 (SAR1) [p. 12-6]			
0x344	Destination address register 1 (DAR1) [p. 12-7]			
0x348	DMA control register 1 (DCR1) [p. 12-8]			
0x34C	Byte count register 1 (BCR24BIT = 0) [1]		Reserved	
0x34C	Reserved	Byte count register 1 (BCR24BIT = 1) [1] (BCR1) [p. 12-7]		
0x350	DMA status register 1 (DSR1) [p. 12-10]	Reserved		
0x354	DMA interrupt vector register 1 (DIVR1) [p. 12-11]	Reserved		
0x380	Source address register 2 (SAR2) [p. 12-6]			
0x384	Destination address register 2 (DAR2) [p. 12-7]			
0x388	DMA control register 2 (DCR2) [p. 12-8]			
0x38C	Byte count register 2 (BCR24BIT = 0) [1]		Reserved	
0x38C	Reserved	Byte count register 2 (BCR24BIT = 1) [1] (BCR2) [p. 12-7]		
0x390	DMA status register 2 (DSR2) [p. 12-10]	Reserved		

Table A-10. DMA Controller Registers (Continued)

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x394	DMA interrupt vector register 2 (DIVR2) [p. 12-11]	Reserved		
0x3C0	Source address register 3 (SAR3) [p. 12-6]			
0x3C4	Destination address register 3 (DAR3) [p. 12-7]			
0x3C8	DMA control register 3 (DCR3) [p. 12-8]			
0x3CC	Byte count register 3 (BCR24BIT = 0) [1]		Reserved	
0x3CC	Reserved	Byte count register 3 (BCR24BIT = 1) [1] (BCR3) [p. 12-7]		
0x3D0	DMA status register 3 (DSR3) [p. 12-10]	Reserved		
0x3D4	DMA interrupt vector register 3 (DIVR3) [p. 12-11]	Reserved		

[1] On the 0H55J and 1H55J revisions of the MCF5307, the byte count register of the DMA channels can accommodate only 16 bits. However, on the newest revision of the MCF5307, an expanded 24-bit byte count range provides greater flexibility. For this reason, the position of the byte count register (BCR) in the memory map depends on whether a 16- or 24-bit byte counter is chosen. The selection is made by programming MPARK[BCR24BIT] in the SIM module.

In the new MCF5307, the 24-bit byte count can be selected by setting BCR24BIT = 1, making DCR[AT] available. The AT bit selects whether the DMA channels assert an acknowledge during the entire transfer or only at the final transfer of a DMA transaction.

New applications should take advantage of the full range of the 24-bit byte counter, including the AT bit. The 16-bit byte count option (BCR24BIT = 0) is kept to retain compatibility with older revisions of the MCF5307.

Glossary of Terms and Abbreviations

The glossary contains an alphabetical list of terms, phrases, and abbreviations used in this book.

A

Architecture. A detailed specification of requirements for a processor or computer system. It does not specify details of how the processor or computer system must be implemented; instead it provides a template for a family of compatible *implementations*.

Autovector. A method of determining the starting address of the service routine by fetching the value from a lookup table internal to the processor instead of requesting the value from the system.

B

Branch prediction.The process of guessing whether a branch will be taken. Such predictions can be correct or incorrect; the term 'predicted' as it is used here does not imply that the prediction is correct (successful).

Branch resolution.The determination of whether a branch is taken or not taken. A branch is said to be resolved when the processor can determine which instruction path to take. If the branch is resolved as predicted, the instructions following the predicted branch that may have been speculatively executed can complete (see completion). If the branch is not resolved as predicted, instructions on the mispredicted path, and any results of speculative execution, are purged from the pipeline and fetching continues from the nonpredicted path.

Burst. A multiple-beat data transfer.

C

Cache. High-speed memory containing recently accessed data and/or instructions (subset of main memory).

Cache coherency. An attribute wherein an accurate and common view of memory is provided to all devices that share the same memory system. Caches are coherent if a processor performing a read from

its cache is supplied with data corresponding to the most recent value written to memory or to another processor's cache.

Cache flush. An operation that removes from a cache any data from a specified address range. This operation ensures that any modified data within the specified address range is written back to main memory.

Cache line. The smallest unit of consecutive data or instructions that is stored in a cache. For ColdFire processors a line consists of 16 bytes.

Caching-inhibited. A memory update policy in which the *cache* is bypassed and the load or store is performed to or from main memory.

Cast outs. *Cache lines* that must be written to memory when a cache miss causes a *cache line* to be replaced.

Clear. To cause a bit or bit field to register a value of zero. See also Set.

Copyback. A cache memory update policy in which processor write cycles are directly written only to the cache. External memory is updated only indirectly, for example, when a modified cache line is *cast out* to make room for newer data.

E

Effective address (EA). The 32-bit address specified for an instruction.

Exception. A condition encountered by the processor that requires special, supervisor-level processing.

Exception handler. A software routine that executes when an exception is taken. Normally, the exception handler corrects the condition that caused the exception, or performs some other meaningful task (that may include aborting the program that caused the exception). The address for each exception handler is identified by an exception vector defined by the ColdFire architecture.

F

Fetch. The act of retrieving instructions from either the cache or main memory and making them available to the instruction unit.

Flush. An operation that causes a modified cache line to be invalidated and the data to be written to memory.

I

Illegal instructions. A class of instructions that are not implemented for a particular processor. These include instructions not defined by the ColdFire architecture.

Implementation. A particular processor that conforms to the ColdFire architecture, but may differ from other architecture-compliant implementations for example in design, feature set, and implementation of *optional* features. The ColdFire architecture has many different implementations.

Imprecise mode. A memory access mode that allows write accesses to a specified memory region to occur out of order.

Instruction queue. A holding place for instructions fetched from the current instruction stream.

Instruction latency. The total number of clock cycles necessary to execute an instruction and make the results of that instruction available.

Interrupt. An *asynchronous exception*. On ColdFire processors, interrupts are a special case of exceptions. See also asynchronous exception.

Invalid state. State of a cache entry that does not currently contain a valid copy of a cache line from memory.

L

Least-significant bit (lsb). The bit of least value in an address, register, data element, or instruction encoding.

Least-significant byte (LSB). The byte of least value in an address, register, data element, or instruction encoding.

Longword. A 32-bit data element

M

Master. A device able to initiate data transfers on a bus. Bus mastering refers to a feature supported by some bus architectures that allow a controller connected to the bus to communicate directly with other devices on the bus without going through the CPU.

Memory coherency. An aspect of caching in which it is ensured that an accurate view of memory is provided to all devices that share system memory.

Modified state. Cache state in which only one caching device has the valid data for that address.

Most-significant bit (msb). The highest-order bit in an address, registers, data element, or instruction encoding.

Most-significant byte (MSB). The highest-order byte in an address, registers, data element, or instruction encoding.

N

Nop. No-operation. A single-cycle operation that does not affect registers or generate bus activity.

O

Overflow. An condition that occurs during arithmetic operations when the result cannot be stored accurately in the destination register(s). For example, if two 16-bit numbers are multiplied, the result may not be representable in 16 bits.

P

Pipelining. A technique that breaks operations, such as instruction processing or bus transactions, into smaller distinct stages or tenures (respectively) so that a subsequent operation can begin before the previous one completes.

Precise mode. A memory access mode that ensures that all write accesses to a specified memory region occur in order.

S

Set (v) To write a nonzero value to a bit or bit field; the opposite of *clear*. The term ‘set’ may also be used to generally describe the updating of a bit or bit field.

Set (*n*). A subdivision of a *cache*. Cacheable data can be stored in a given location in any one of the sets, typically corresponding to its lower-order address bits. Because several memory locations can map to the same location, cached data is typically placed in the set whose *cache line* corresponding to that address was used least recently. *See* Set-associativity.

Set-associativity. Aspect of cache organization in which the cache space is divided into sections, called *sets*. The cache controller associates a particular main memory address with the contents of a particular set, or region, within the cache.

Slave. The device addressed by a master device. The slave is identified in the address tenure and is responsible for supplying or latching the requested data for the master during the data tenure.

Static branch prediction. Mechanism by which software (for example, compilers) can hint to the machine hardware about the direction a branch is likely to take.

Supervisor mode. The privileged operation state of a processor. In supervisor mode, software, typically the operating system, can access all control registers and can access the supervisor memory space, among other privileged operations.

System memory. The physical memory available to a processor.

T

Tenure. A tenure consists of three phases: arbitration, transfer, termination. There can be separate address bus tenures and data bus tenures.

Throughput. The measure of the number of instructions that are processed per clock cycle.

Transfer termination. The successful or unsuccessful conclusion of a data transfer.

U

Underflow. A condition that occurs during arithmetic operations when the result cannot be represented accurately in the destination register.

User mode. The operating state of a processor used typically by application software. In user mode, software can access only certain control registers and can access only user memory space. No privileged operations can be performed.

W

Word. A 16-bit data element.

Write-through. A cache memory update policy in which all processor write cycles are written to both the cache and memory.

INDEX

INDEX

INDEX

INDEX

S

INDEX

INDEX

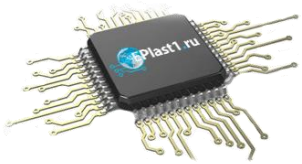
Компания «ЭлектроПласт» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Оперативные поставки широкого спектра электронных компонентов отечественного и импортного производства напрямую от производителей и с крупнейших мировых складов;
- Поставка более 17-ти миллионов наименований электронных компонентов;
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- Лицензия ФСБ на осуществление работ с использованием сведений, составляющих государственную тайну;
- Поставка специализированных компонентов (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Aeroflex, Peregrine, Syfer, Eurofarad, Texas Instrument, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits,General Dynamics и др.);

Помимо этого, одним из направлений компании «ЭлектроПласт» является направление «Источники питания». Мы предлагаем Вам помощь Конструкторского отдела:

- Подбор оптимального решения, техническое обоснование при выборе компонента;
- Подбор аналогов;
- Консультации по применению компонента;
- Поставка образцов и прототипов;
- Техническая поддержка проекта;
- Защита от снятия компонента с производства.



Как с нами связаться

Телефон: 8 (812) 309 58 32 (многоканальный)
Факс: 8 (812) 320-02-42
Электронная почта: org@eplast1.ru
Адрес: 198099, г. Санкт-Петербург, ул. Калинина, дом 2, корпус 4, литера А.