

MCF5307

USER'S MANUAL

MOTOROLA

ColdFire



MCF5307

User's Manual



MOTOROLA




MOTOROLA

MCF5307

ColdFire[®]

Integrated Microprocessor

User's Manual

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

DOCUMENTATION FEEDBACK

**FAX 512-891-8593—Documentation Comments Only (no technical questions please)
http: // www.mot.com/hpesd/docs_survey.html—Documentation Feedback Only**

The Technical Communications Department welcomes your suggestions for improving our documentation and encourages you to complete the documentation feedback form at the World Wide Web address listed above. In return for your efforts, you will receive a small token of our appreciation. Your help helps us measure how well we are serving your information requirements.

The Technical Communications Department also provides a fax number for you to submit any questions or comments about this document or how to order other documents. Please provide the part number and revision number (located in upper right-hand corner of the cover) and the title of the document. When referring to items in the manual, please reference by the page number, paragraph number, figure number, table number, and line number if needed. **Please do not fax technical questions to this number.**

When sending a fax, please provide your name, company, fax number, and phone number including area code.

For Internet Access:

Web Only: [http: // www.motorola.com/coldfire](http://www.motorola.com/coldfire)

For Hotline Questions:

FAX (US or Canada): 1-800-248-8567

Applications and Technical Information

For questions or comments pertaining to technical information, questions, and applications, please contact one of the following sales offices nearest you.

— Sales Offices —

Field Applications Engineering Available Through All Sales Offices

UNITED STATES

ALABAMA, Huntsville (205) 464-6800
ARIZONA, Tempe (602) 897-5056
CALIFORNIA, Agoura Hills (818) 706-1929
CALIFORNIA, Los Angeles (310) 417-8848
CALIFORNIA, Irvine (714) 753-7380
CALIFORNIA, Roseville (916) 922-7152
CALIFORNIA, San Diego (619) 541-2163
CALIFORNIA, Sunnyvale (408) 749-0510
COLORADO, Colorado Springs (719) 599-7497
COLORADO, Denver (303) 337-3434
CONNECTICUT, Wallingford (203) 949-4100
FLORIDA, Maitland (407) 628-2636
FLORIDA, Pompano Beach/
 Fort Lauderdale (305) 486-9776
FLORIDA, Clearwater (813) 538-7750
GEORGIA, Atlanta (404) 729-7100
IDAHO, Boise (208) 323-9413
ILLINOIS, Chicago/Hoffman Estates (708) 490-9500
INDIANA, Fort Wayne (219) 436-5818
INDIANA, Indianapolis (317) 571-0400
INDIANA, Kokomo (317) 457-6634
IOWA, Cedar Rapids (319) 373-1328
KANSAS, Kansas City/Mission (913) 451-8555
MARYLAND, Columbia (410) 381-1570
MASSACHUSETTS, Marlborough (508) 481-8100
MASSACHUSETTS, Woburn (617) 932-9700
MICHIGAN, Detroit (313) 347-6800
MINNESOTA, Minnetonka (612) 932-1500
MISSOURI, St. Louis (314) 275-7380
NEW JERSEY, Fairfield (201) 808-2400
NEW YORK, Fairfield (716) 425-4000
NEW YORK, Hauppauge (516) 361-7000
NEW YORK, Poughkeepsie/Fishkill (914) 473-8102
NORTH CAROLINA, Raleigh (919) 870-4355
OHIO, Cleveland (216) 349-3100
OHIO, Columbus/Worthington (614) 431-8492
OHIO, Dayton (513) 495-6800
OKLAHOMA, Tulsa (800) 544-9496
OREGON, Portland (503) 641-3681
PENNSYLVANIA, Colmar (215) 997-1020
 Philadelphia/Horsham (215) 957-4100
TENNESSEE, Knoxville (615) 584-4841
TEXAS, Austin (512) 873-2000
TEXAS, Houston (800) 343-2692
TEXAS, Plano (214) 516-5100
VIRGINIA, Richmond (804) 285-2100
WASHINGTON, Bellevue (206) 454-4160
 Seattle Access (206) 622-9960
WISCONSIN, Milwaukee/Brookfield (414) 792-0122

CANADA

BRITISH COLUMBIA, Vancouver (604) 293-7605
ONTARIO, Toronto (416) 497-8181
ONTARIO, Ottawa (613) 226-3491
QUEBEC, Montreal (514) 731-6881

INTERNATIONAL

AUSTRALIA, Melbourne (61-3)887-0711
AUSTRALIA, Sydney (61-2)906-3855
BRAZIL, Sao Paulo 55(11)815-4200
CHINA, Beijing 86 505-2180
FINLAND, Helsinki 358-0-35161191
 Car Phone 358(49)211501
FRANCE, Paris/Vanves 33(1)40 955 900

GERMANY, Langenhagen/ Hanover 49(511)789911
GERMANY, Munich 49 89 92103-0
GERMANY, Nuremberg 49 911 64-3044
GERMANY, Sindelfingen 49 7031 69 910
GERMANY, Wiesbaden 49 611 761920
HONG KONG, Kwai Fong 852-4808333
 Tai Po 852-6668333
INDIA, Bangalore (91-812)627094
ISRAEL, Tel Aviv 972(3)753-8222
ITALY, Milan 39(2)82201
JAPAN, Aizu 81(241)272231
JAPAN, Atsugi 81(0462)23-0761
JAPAN, Kumagaya 81(0485)26-2600
JAPAN, Kyushu 81(092)771-4212
JAPAN, Mito 81(0292)26-2340
JAPAN, Nagoya 81(052)232-1621
JAPAN, Osaka 81(06)305-1801
JAPAN, Sendai 81(22)268-4333
JAPAN, Tachikawa 81(0425)23-6700
JAPAN, Tokyo 81(03)3440-3311
JAPAN, Yokohama 81(045)472-2751
KOREA, Pusan 82(51)4635-035
KOREA, Seoul 82(2)554-5188
MALAYSIA, Penang 60(4)374514
MEXICO, Mexico City 52(5)282-2864
MEXICO, Guadalajara 52(36)21-8977
 Marketing 52(36)21-9023
 Customer Service 52(36)669-9160
NETHERLANDS, Best (31)49988 612 11
PUERTO RICO, San Juan (809)793-2170
SINGAPORE (65)2945438
SPAIN, Madrid 34(1)457-8204
 or 34(1)457-8254
SWEDEN, Solna 46(8)734-8800
SWITZERLAND, Geneva 41(22)7991111
SWITZERLAND, Zurich 41(1)730 4074
TAIWAN, Taipei 886(2)717-7089
THAILAND, Bangkok (66-2)254-4910
UNITED KINGDOM, Aylesbury 44(296)395-252

FULL LINE REPRESENTATIVES

COLORADO, Grand Junction
 Cheryl Lee Whitely (303) 243-9658
KANSAS, Wichita
 Melinda Shores/Kelly Greiving (316) 838 0190
NEVADA, Reno
 Galena Technology Group (702) 746 0642
NEW MEXICO, Albuquerque
 S&S Technologies, Inc. (505) 298-7177
UTAH, Salt Lake City
 Utah Component Sales, Inc. (801) 561-5099
WASHINGTON, Spokane
 Doug Kenley (509) 924-2322
ARGENTINA, Buenos Aires
 Argonics, S.A. (541) 343-1787

HYBRID COMPONENTS RESELLERS

Elmo Semiconductor (818) 768-7400
 Minco Technology Labs Inc. (512) 834-2022
 Semi Dice Inc. (310) 594-4631

PREFACE

The *MCF5307 ColdFire® Integrated Microprocessor User's Manual* describes the programming, capabilities, and operation of the MCF5307 device. Refer to the *ColdFire Family Programmer's Reference Manual Rev 1.0* (MCF5200PRMREV1/D) for information on the ColdFire Family of microprocessors.

CONTENTS

This user manual is organized as follows:

- Section 1: Introduction
- Section 2: Signal Description
- Section 3: ColdFire Core
- Section 4: PLL
- Section 5: Cache
- Section 6: SRAM
- Section 7: Bus Operation
- Section 8: System Integration Module (SIM)
- Section 9: Chip-Select Module
- Section 10: Parallel Port (General-Purpose I/O) Module
- Section 11: DRAM Controller
- Section 12: Timer Module
- Section 13: DMA Controller
- Section 14: UART Module
- Section 15: M-Bus Module
- Section 16: Debug Support
- Section 17: IEEE 1149.1 Test Access Port (JTAG)
- Section 18: Electrical Characteristics
- Section 19: Mechanical Characteristics
- Appendix A: Memory Map Register Description
- Appendix B: Memory Map
- Index

TABLE OF CONTENTS

Paragraph Number	Title	Page Number
---------------------	-------	----------------

Section 1 INTRODUCTION

1.1	Overview	1-4
1.1.1	ColdFire Processor Core.....	1-4
1.1.2	Multiply and Accumulate (MAC) Module	1-5
1.1.3	8K Byte Unified Cache	1-5
1.1.4	Internal 4K Byte SRAM	1-5
1.1.5	DRAM Controller	1-5
1.1.6	DMA Controller.....	1-6
1.1.7	DUART Module	1-6
1.1.8	Timer Module	1-6
1.1.9	Motorola Bus (M-Bus) Module.....	1-6
1.1.10	System Interface	1-6
1.1.10.1	External Bus Interface	1-6
1.1.10.2	Chip-Selects	1-7
1.1.10.3	16-Bit Parallel-Port Interface.....	1-7
1.1.10.4	Interrupt Controller.....	1-7
1.1.10.5	JTAG	1-7
1.1.11	System Debug Interface	1-7

Section 2 SIGNAL DESCRIPTION

2.1	Introduction	2-1
2.2	MCF5307 Bus Signals.....	2-3
2.2.1	Address Bus	2-3
2.2.1.1	Address Bus - (A[23:0])	2-3
2.2.1.2	Address Bus - (A[31:24]/PP[15:8]).....	2-3
2.2.2	Data Bus(D[31:0])	2-4
2.2.3	Read/Write - (R/W)	2-4
2.2.4	Size - (SIZ[1:0])	2-4
2.2.5	Transfer Start - (TS)	2-4
2.2.6	Address Strobe - (AS)	2-4
2.2.7	Transfer Acknowledge - (TA)	2-5
2.2.8	Transfer In Progress - (TIP/PP[7])	2-5
2.2.9	Transfer Type - (TT[1:0]/PP[1:0])	2-5
2.2.10	Transfer Modifier - (TM[2:0]/PP[4:2])	2-5

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
2.3	Interrupt Control Signals	2-7
2.3.1	Interrupt Request - (IRQ7,IRQ5, IRQ3,IRQ1)	2-7
2.4	Bus Arbitration Signals	2-7
2.4.1	Bus Request - (BR)	2-7
2.4.2	Bus Grant - (BG)	2-7
2.4.3	Bus Driven - (BD)	2-7
2.5	Clock and Reset Signals	2-7
2.5.1	Reset - (RSTI)	2-7
2.5.2	Clock Input - (CLKIN)	2-7
2.5.3	System Bus Clock Output - (BCLKO)	2-8
2.5.4	Reset Out - (RSTO)	2-8
2.5.5	Frequency Control PLL - FREQ[1:0]	2-8
2.5.6	Divide Control PCLK to BCLK - DIVIDE[1:0]	2-8
2.6	Chip-Selects	2-9
2.6.1	Chip-Selects - (CS[7:0])	2-9
2.6.2	Chip-Select Config - (CS_CONF[2:0])	2-9
2.6.3	Byte Enables/Byte Write Enables - (BE[3:0]/BWE[3:0])	2-9
2.6.4	Output Enable - (OE)	2-10
2.7	DRAM Controller Signals	2-10
2.7.1	Row Address Strokes - (RAS[1:0])	2-10
2.7.2	Column Address Strokes - (CAS[3:0])	2-10
2.7.3	DRAM Read/Write - (DRAMW)	2-10
2.7.4	Synchronous DRAM Column Address Strobe - (SCAS)	2-10
2.7.5	Synchronous DRAM Row Address Strobe - (SRAS)	2-10
2.7.6	Synchronous DRAM Clock Enable - (SCKE)	2-10
2.7.7	Synchronous Edge Select - (EDGESEL)	2-10
2.8	DMA Module Signals	2-11
2.8.1	DMA Request (DREQ[1:0]/PP[6:5])	2-11
2.9	Serial Module Signals	2-11
2.9.1	Receive Data - (RD[1:0])	2-11
2.9.2	Transmit Data - (TD[1:0])	2-11
2.9.3	Request To Send - (RTS[1:0])	2-11
2.9.4	Clear To Send - (CTS[1:0])	2-11
2.10	Timer Module Signals	2-11
2.10.1	Timer Input - (TIN[1:0])	2-12
2.10.2	Timer Output - (TOUT1, TOUT0)	2-12
2.11	Parallel Port (PP[15:0])	2-12
2.12	M-Bus Module Signals	2-12
2.12.1	M-Bus Serial Clock (SCL)	2-12
2.12.2	M-Bus Serial Data (SDA)	2-12
2.13	Debug and Test Signals	2-12
2.13.1	Test Mode - (MTMOD[3:0])	2-12

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
2.13.2	High Impedance - (HIZ)	2-13
2.13.3	Processor Clock Output - (PSTCLK)	2-13
2.13.4	Debug Data - (DDATA[3:0])	2-13
2.13.5	Processor Status - (PST[3:0])	2-13
2.14	BDM/JTAG Signals	2-14
2.14.1	Test Clock - (TCK)	2-14
2.14.2	Test Reset/Development Serial Clock - (TRST/DSCLK)	2-14
2.14.3	Test Mode Select/ Break Point (TMS/BKPT)	2-14
2.14.4	Test Data Input/Development Serial Input - (TDI/DSI)	2-14
2.14.5	Test Data Output/Development Serial Output - (TDO/DSO)....	2-15

Section 3 COLDFIRE CORE

3.1	Enhancements	3-1
3.1.1	Process	3-1
3.1.2	Clock Doubled Microprocessor Core	3-1
3.1.3	Bus Clock at 1/2, 1/3, or 1/4 Processor Clock	3-2
3.1.4	Enhanced Pipeline	3-2
3.1.5	Change of Flow Acceleration	3-5
3.1.6	Illegal Opcode Handling	3-6
3.1.7	Hardware Multiply / Accumulate and Divide Execution Units	3-6
3.1.8	Debug Module Enhancements	3-7
3.2	Programming Model	3-7
3.2.1	User Programming Model	3-7
3.2.1.1	Data Registers (D0–D7)	3-7
3.2.1.2	Address Registers (A0–A6)	3-7
3.2.1.3	Stack Pointer (A7)	3-8
3.2.1.4	Program Counter	3-8
3.2.1.5	Condition Code Register	3-8
3.2.2	Supervisor Programming Model	3-9
3.2.2.1	Status Register	3-9
3.2.2.2	Vector Base Register (VBR)	3-10
3.2.3	MAC Programming Model	3-10
3.2.3.1	Accumulator (ACC)	3-11
3.2.3.2	Mask Register (MASK)	3-11
3.2.3.3	MAC Status Register (MACSR)	3-11
3.2.4	Supervisor Programming Model	3-11
3.2.4.1	Cache Control Register (CACR)	3-12
3.2.4.2	Access Control Registers (ACR0, ACR1)	3-12
3.2.4.3	Vector Base Register (VBR)	3-12
3.2.4.4	RAMBAR0	3-12

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
3.2.4.5	ROM Base Address Register (ROMBAR0)	3-12
3.2.4.6	Status Register (SR)	3-12
3.3	Exception Processing Overview	3-13
3.4	Exception Stack Frame Definition	3-13
3.5	Processor Exceptions	3-14
3.5.1	Access Error Exception	3-14
3.5.2	Address Error Exception	3-15
3.5.3	Illegal Instruction Exception	3-15
3.5.4	Privilege Violation	3-16
3.5.5	Trace Exception	3-16
3.5.6	Debug Interrupt	3-16
3.5.7	RTE and Format Error Exceptions	3-16
3.5.8	TRAP Instruction Exceptions	3-17
3.5.9	Interrupt Exception	3-17
3.5.10	Fault-on-Fault Halt	3-17
3.5.11	Reset Exception	3-17
3.6	Integer Data Formats	3-18
3.7	Organization of Data in Registers	3-18
3.7.1	Organization of Integer Data Formats in Registers	3-18
3.7.2	Organization of Integer Data Formats in Memory	3-19
3.8	Addressing Mode Summary	3-20
3.9	Instruction Set Summary	3-21
3.9.1	Timing Assumptions	3-25
3.9.2	MOVE Instruction Execution Times	3-26
3.10	Standard One Operand Instruction Execution Times	3-27
3.11	Standard Two Operand Instruction Execution Times.....	3-28
3.12	Miscellaneous Instruction Execution Times	3-29
3.13	Branch Instruction Execution Times.....	3-30

Section 4 PHASE LOCKED LOOP

4.1	PLL Features	4-1
4.2	PLL Specifications	4-1
4.3	PLL Operation	4-2
4.3.1	Normal Mode	4-2
4.3.2	Reset / Initialization	4-2
4.3.3	Reduced-Power Mode	4-2
4.3.4	Phase Locked Loop Control Register (PLLCR)	4-2
4.4	PLL Port List	4-3
4.4.1	Inputs	4-3
4.4.2	Outputs	4-4

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
4.5	Timing Diagrams	4-5
4.5.1	PSTCLK and BCLKO	4-5
4.5.2	RSTI Timing	4-5

Section 5 CACHE

5.1	Cache Organization	5-2
5.2	Cache Operation	5-2
5.3	Cache Control Register (CACR)	5-5
5.4	Access Control Registers	5-7
5.5	Cache Management	5-8
5.6	CACHING MODES	5-9
5.6.1	Cacheable Accesses	5-10
5.6.1.1	Writethrough Mode	5-10
5.6.1.2	Copyback Mode	5-10
5.6.2	Cache-Inhibited Accesses	5-10
5.7	Cache Protocol	5-11
5.7.1	Read Miss	5-11
5.7.2	Write Miss	5-11
5.7.3	Read Hit	5-11
5.7.4	Write Hit	5-12
5.8	Cache Coherency	5-12
5.9	Memory Accesses for Cache Maintenance	5-12
5.9.1	Cache Filling	5-12
5.9.2	Cache Pushes	5-13
5.10	Push and Store Buffers	5-13
5.11	Push And Store Buffer Bus Operation	5-14
5.12	Cache Operation Summary	5-14

Section 6 SRAM

6.1	SRAM Features	6-1
6.2	SRAM Operation	6-1
6.3	SRAM Programming Model	6-1
6.3.1	SRAM Base Address Register (RAMBAR)	6-1
6.3.2	SRAM Initialization	6-3
6.3.3	SRAM Initialization Code.....	6-4
6.3.4	Power Management	6-4

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
Section 7		
BUS OPERATION		
7.1	Features	7-1
7.2	Bus And Control Signals	7-1
7.2.1	Address Bus	7-2
7.2.1.1	Address Bus - (A[23:0])	7-2
7.2.1.2	Address Bus - (A[31:24]/PP[15:8])	7-2
7.2.2	Data Bus(D[31:0])	7-3
7.2.3	Read/Write - (R/W).....	7-3
7.2.4	Size - (SIZ[1:0])	7-3
7.2.5	Transfer Start - (TS)	7-3
7.2.6	Address Strobe - (AS).....	7-3
7.2.7	Transfer Acknowledge - (TA)	7-3
7.2.8	Transfer In Progress - (TIP/PP[7])	7-4
7.2.9	Transfer Type - (TT[1:0]/PP[1:0])	7-4
7.2.10	Transfer Modifier - (TM[2:0]/PP[4:2])	7-4
7.2.11	Interrupt Request - (IRQ7,IRQ5, IRQ3,IRQ1)	7-4
7.3	Clock and Reset Signals	7-5
7.3.1	Reset - (RSTI)	7-5
7.3.2	Clock Input - (CLKIN)	7-5
7.3.3	System Bus Clock Output - (BCLKO)	7-5
7.3.4	Reset Out - (RSTO)	7-5
7.4	Bus Characteristics	7-6
7.5	Data Transfer Operations	7-6
7.5.1	Read Cycle	7-8
7.5.2	Write Cycle	7-10
7.5.3	Fast Termination Cycles	7-12
7.5.4	Back-to-Back Bus Cycles	7-14
7.5.5	Line Bus Cycles	7-15
7.5.5.1	Line Read Bus Cycles	7-17
7.5.5.2	Line Write Bus Cycles	7-20
7.6	Interrupt Exceptions	7-23
7.6.1	Level 7 Interrupts	7-24
7.6.2	Interrupt-Acknowledge Cycle	7-24
7.7	Bus Arbitration.....	7-25
7.7.1	Bus Arbitration Signals	7-26
7.7.2	Bus Request - (BR)	7-26
7.7.3	Bus Grant - (BG)	7-26
7.7.4	Bus Driven - (BD)	7-26
7.7.5	Two Master Bus Arbitration Protocol (Two-Wire Mode)	7-27
7.7.6	Multiple External Bus Master Arbitration Protocol (Three-Wire Mode)	7-33

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
7.8	Reset Operation	7-40
7.8.1	Master Reset	7-40
7.8.2	Software Watchdog Reset	7-41

Section 8 SYSTEM INTEGRATION MODULE

8.1	Introduction	8-1
8.1.1	Features	8-1
8.2	Programming Model	8-2
8.2.1	SIM Registers Memory Map	8-2
8.3	SIM Programming and Configuration	8-2
8.3.1	Module Base Address Register (MBAR)	8-2
8.3.2	Module Base Address Register (MBAR)	8-3
8.3.3	Interrupt Controller	8-5
8.3.4	Interrupt Registers (ICR[11:0], IPR, IMR, AVR, and IRQPAR) ..	8-5
8.3.4.1	Interrupt Assignment Register (IRQPAR)	8-9
8.3.5	System Protection And Reset Status	8-10
8.3.5.1	Reset Status Register (RSR)	8-10
8.3.5.2	Software Watchdog Timer (SWT).....	8-10
8.3.5.3	System Protection Control Register (SYPCR)	8-12
8.3.5.4	Software Watchdog Interrupt Vector Register (SWIVR)	8-13
8.3.5.5	Software Watchdog Service Register (SWSR)	8-14
8.3.6	Phase-Locked-Loop Clock Control for CPU STOP Instruction	8-14
8.3.6.1	Phase-Locked Loop Control Register (PLLCR)	8-14
8.3.7	Bus Arbitration Control	8-15
8.3.7.1	Default Bus Master Register (MPARK)	8-15
8.3.8	Parallel Port Pin Assignment Register (PAR)	8-19
8.3.8.1	Pin Assignment Register (PAR)	8-19

Section 9 CHIP-SELECT MODULE

9.1	Introduction	9-1
9.1.1	Features	9-1
9.2	Chip Select Signals	9-1
9.2.1	Chip-Selects - (CS[7:0])	9-1
9.2.2	Output Enable - (OE)	9-1
9.2.3	Byte Enables/Byte Write Enables - (BE[3:0]/BWE[3:0])	9-2
9.3	Chip-Select Operation	9-4

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
9.3.1	Chip-Select Module	9-4
9.3.1.1	General Chip-Select Operation	9-5
9.3.1.1.1	8-, 16-, and 32-Bit Port Sizing	9-6
9.3.2	Global Chip-Select Operation	9-6
9.4	Programming Model	9-7
9.4.1	Chip-Select Registers Memory Map	9-7
9.4.2	Chip-Select Module Registers	9-9
9.4.2.1	Chip-Select Address Register (CSAR0, CSAR1 and CSBAR)	9-9
9.4.2.2	Chip-Select Mask Register (CSMR0 - CSMR7)	9-10
9.4.2.3	Chip-Select Control Register (CSCR0 - CSCR7)	9-13
9.4.2.4	Code Example	9-15
9.5	Timing Diagrams	9-16

Section 10

PARALLEL PORT (GENERAL-PURPOSE I/O)

10.1	Introduction	10-1
10.2	Signal Descriptions	10-1
10.3	Parallel Port Operation	10-3
10.3.1	Port A Data Direction Register (PADDR)	10-3
10.3.1.1	Port A Data Register (PADAT)	10-4
10.3.1.2	Example Code	10-4

Section 11

SYNCHRONOUS/ASYNCHRONOUS DRAM CONTROLLER MODULE

11.1	Introduction	11-1
11.2	Features	11-1
11.3	Block Diagram	11-1
11.3.1	Refresh Register Block	11-3
11.3.2	Refresh Counter	11-3
11.3.3	Hit Logic	11-3
11.3.4	Control Logic and State Machine	11-3
11.3.5	Page Hit Logic	11-3
11.3.6	Address Mux	11-3
11.4	Signal List	11-3
11.4.1	Row Address Strokes - (RAS[1:0])	11-3
11.4.2	Column Address Strokes - (CAS[3:0])	11-4
11.4.3	DRAM Read/Write - (DRAMW)	11-4
11.4.4	Synchronous DRAM Column Address Strobe - (SCAS)	11-4
11.4.5	Synchronous DRAM Row Address Strobe - (SRAS)	11-4

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
11.4.6	Synchronous DRAM Clock Enable - (SCKE)	11-4
11.4.7	Synchronous Edge Select - (EDGESEL)	11-4
11.4.7.1	Edge Select Output Cell	11-4
11.5	Asynchronous Operation	11-6
11.5.1	Asynchronous Memory Map	11-6
11.5.1.1	DRAM Control Register -DCR	11-6
11.5.1.1.1	Synchronous Operation - SO	11-7
11.5.1.1.2	No Address Multiplexing - NAM	11-7
11.5.1.1.3	Refresh RAS Asserted - RRA[1:0]	11-7
11.5.1.1.4	Refresh RAS Precharged - RRP[1:0]	11-7
11.5.1.1.5	Refresh Count - RC[8:0]	11-8
11.5.1.2	Address & Control Registers - DACR0 & DACR1	11-8
11.5.1.2.1	Base Address Register - BA[31:18]	11-8
11.5.1.2.2	Refresh Enable - RE	11-8
11.5.1.2.3	Column Address Strobe Timing - CAS[1:0]	11-9
11.5.1.2.4	Row Address Strobe Precharge Timing - RP[1:0]	11-9
11.5.1.2.5	RAS Negate to CAS NEGATE - RNCN	11-9
11.5.1.2.6	RAS to CAS Delay -- RCD	11-10
11.5.1.2.7	Extended Data Out -- EDO	11-10
11.5.1.2.8	Port Size -- PS	11-10
11.5.1.2.9	Page Mode -- PM	11-11
11.5.1.3	DRAM Controller Mask Registers - DMR0 & DMR1 ...	11-11
11.5.1.3.1	Base Address Mask -- BAM[31:18]	11-11
11.5.1.3.2	Write Protect -- WP	11-12
11.5.1.3.3	Address Modifier Masks -- C/I, AM, SC, SD, UC, UD	11-12
11.5.1.3.4	Valid -- V	11-12
11.5.2	Asynchronous Operation	11-13
11.5.2.1	General Operation Guidelines	11-13
11.5.2.2	Nonpage Mode Operation.	11-16
11.5.2.3	Burst Page Mode Operation	11-17
11.5.2.4	Continuous Page Mode	11-19
11.5.2.5	Extended Data Out (EDO) Operation	11-21
11.5.2.6	Refresh Operation	11-22
11.5.2.7	External Master Support	11-23
11.6	Synchronous Operation	11-23
11.6.1	Synchronous Memory Map	11-23
11.6.1.1	DRAM Control Register	11-23

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
11.6.1.1.1	Synchronous Operation - SO	11-24
11.6.1.1.2	No Address Multiplexing - NAM	11-24
11.6.1.1.3	Command on Clock Enable - COC	11-24
11.6.1.1.4	Initiate Self Refresh Command - IS	11-25
11.6.1.1.5	Refresh Timing - RTIM.....	11-25
11.6.1.1.6	Refresh Count - RC	11-25
11.6.1.2	Address & Control Registers - DACR0 & DACR1	11-26
11.6.1.2.1	Base Address Register - BA[31:18]	11-26
11.6.1.2.2	Refresh Enable - RE	11-26
11.6.1.2.3	Column Address Strobe Latency - CASL[1:0]	11-26
11.6.1.2.4	Initiate Mode Register Set Command - IMRS	11-27
11.6.1.2.5	Command and Bank Mux - CBM[2:0]	11-27
11.6.1.2.6	Port Size - PS	11-28
11.6.1.2.7	Initiate Precharge All Command - IP	11-28
11.6.1.2.8	Page Mode -- PM	11-29
11.6.1.3	DRAM Controller Mask Registers - DMR0 & DMR1....	11-29
11.6.2	Synchronous/Asynchronous DRAM Controller Synchronous Operation	11-29
11.6.2.1	Synchronous DRAM General Operation Guidelines ..	11-29
11.6.2.1.1	Address Multiplexing	11-29
11.6.2.1.2	Power on Sequence	11-33
11.6.2.1.3	Mode Register Settings	11-33
11.6.2.1.4	General Details	11-34
11.6.2.2	Burst Page Mode	11-35
11.6.2.3	Continuous-Page Mode	11-38
11.6.2.4	Auto-Refresh Operation	11-39
11.6.2.5	Self-Refresh Operation	11-40

Section 12 TIMER MODULE

12.1	Overview	12-1
12.1.1	Key Features	12-1
12.2	Module Operation	12-3
12.2.1	General-Purpose Timer Units	12-3
12.2.1.1	Prescaler	12-3
12.2.1.2	Capture Mode	12-3
12.2.1.3	Reference Compare	12-3
12.2.1.4	Output Mode	12-3
12.3	Programming Model	12-3
12.3.1	General-Purpose Timer Registers	12-3
12.3.1.1	Timer Mode Register (TMR)	12-4

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
12.3.1	General-Purpose Timer Registers	12-3
12.3.1.1	Timer Mode Register (TMR)	12-4
12.3.1.2	Timer Reference Register (TRR).....	12-5
12.3.1.3	Timer Capture Register (TCR)	12-5
12.3.1.4	Timer Counter (TCN)	12-6
12.3.1.5	Timer Event Register (TER)	12-6
12.3.1.6	Table of TIMEOUT values	12-6
12.3.1.7	Code example	12-13

Section 13 DMA CONTROLLER MODULE

13.1	Introduction	13-1
13.2	DMA Signal Description	13-3
13.2.1	DMA Request (DREQ[1:0]/PP[6:5])	13-3
13.3	DMA Module Overview	13-3
13.4	DMA Controller Module Programming Model	13-5
13.4.1	Source Address Register (SAR)	13-6
13.4.2	Destination Address Register (DAR)	13-7
13.4.3	Byte Count Register (BCR)	13-7
13.4.4	DMA Control Register	13-8
13.4.5	DMA Status Register (DSR)	13-10
13.4.6	DMA Interrupt Vector Register	13-12
13.5	Transfer Request Generation	13-12
13.5.1	Cycle-Steal Mode	13-12
13.5.2	Continuous Mode	13-12
13.6	Data Transfer Modes	13-12
13.6.1	Single Address Transactions	13-13
13.6.2	Dual Address Transactions	13-13
13.6.2.1	Dual Address Reads	13-13
13.6.2.2	Dual Address Writes	13-13
13.7	DMA Controller Module Functional Description	13-13
13.7.1	Channel Initialization and Startup	13-14
13.7.1.1	Channel Prioritization	13-14
13.7.1.2	Programming the DMA Controller Module	13-14
13.7.2	Data Transfers	13-15
13.7.2.1	External Request Operation	13-15
13.7.2.2	Auto Alignment	13-16
13.7.2.3	BandWidth Control	13-17
13.7.3	Channel Termination	13-17
13.7.3.1	Error Conditions	13-17
13.7.3.2	Interrupts	13-17

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
Section 14		
UART MODULE		
14.1	Serial Module Overview	14-2
14.1.1	Serial Communication Channel.....	14-2
14.1.2	Interrupt Control Logic.....	14-3
14.1.3	Comparison of UART Module to MC68681.....	14-3
14.2	UART Module Signal Definitions.....	14-4
14.2.1	Transmitter Serial Data Output (TxD)	14-4
14.2.2	Receiver Serial Data Input (RxD).....	14-4
14.2.3	Clear-to-Send (\overline{CTS})	14-4
14.2.4	Request-to-Send (\overline{RTS})	14-4
14.3	Operation	14-6
14.3.1	Baud Rate Generator Logic	14-6
14.3.2	Baud Rate Generator/Timer.....	14-6
14.3.3	Calculating Baud Rates.....	14-7
14.3.3.1	System Bus Clock	14-7
14.3.3.2	External Clock	14-7
14.3.4	Transmitter and Receiver Operating Modes	14-7
14.3.4.1	Transmitter	14-7
14.3.4.2	Receiver	14-10
14.3.4.3	FIFO Stack	14-12
14.3.5	Looping Modes.....	14-13
14.3.5.1	Automatic Echo Mode	14-13
14.3.5.2	Local Loopback Mode	14-13
14.3.5.3	Remote Loopback Mode	14-13
14.3.6	Multidrop Mode	14-15
14.3.7	Bus Operation	14-17
14.3.7.1	Read Cycles	14-17
14.3.7.2	Write Cycles	14-17
14.3.7.3	Interrupt Acknowledge Cycles.....	14-17
14.4	Register Description and Programming	14-17
14.4.1	Register Description.....	14-17
14.4.1.1	Mode Register 1	14-18
14.4.1.2	Mode Register 2	14-20
14.4.1.3	Status Register (USR).....	14-22
14.4.1.4	Clock-Select Register (UCSR)	14-24
14.4.1.5	Command Register (UCR)	14-25
14.4.1.6	Receiver Buffer (URB).....	14-28
14.4.1.7	Transmitter Buffer (UTB)	14-29
14.4.1.8	Input Port Change Register (UIPCR)	14-29
14.4.1.9	Auxiliary Control Register (UACR)	14-30

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
14.4.1.10	Interrupt Status Register (UISR).....	14-30
14.4.1.11	Interrupt Mask Register (UIMR).....	14-31
14.4.1.12	Timer Upper Preload Register (UBG1).....	14-32
14.4.1.13	Timer Upper Preload Register (UBG2).....	14-32
14.4.1.14	Interrupt Vector Register (UIVR).....	14-33
14.4.1.15	Input Port Register (UIP)	14-33
14.4.1.16	Output Port Data Registers (UOP1, UPO0).....	14-34
14.4.2	Programming.....	14-34
14.4.2.1	UART Module Initialization	14-35
14.4.2.2	I/O Driver Example	14-35
14.4.2.3	Interrupt Handling	14-35
14.5	UART Module Initialization Sequence.....	14-35

Section 15 M-BUS MODULE

15.1	Overview	15-1
15.2	Interface Features	15-1
15.3	M-Bus System Configuration	15-2
15.4	M-Bus Protocol	15-3
15.4.1	START Signal	15-3
15.4.2	Slave Address Transmission	15-3
15.4.3	Data Transfer	15-4
15.4.4	Repeated START Signal	15-4
15.4.5	STOP Signal	15-4
15.4.6	Arbitration Procedure	15-4
15.4.7	Clock Synchronization	15-5
15.4.8	Handshaking	15-5
15.4.9	Clock Stretching	15-5
15.5	Programming Model	15-6
15.5.1	M-Bus Address Register (MADR)	15-6
15.5.2	M-Bus Frequency Divider Register (MFDR)	15-6
15.5.3	M-Bus Control Register (MBCR)	15-8
15.5.4	M-Bus Status Register (MBSR)	15-9
15.5.5	M-Bus Data I/O Register (MBDR)	15-11
15.6	M-Bus Programming Examples	15-11
15.6.1	Initialization Sequence	15-11
15.6.2	Generation of START	15-11
15.6.3	Post-Transfer Software Response	15-12
15.6.4	Generation of STOP	15-13
15.6.5	Generation of Repeated START	15-14
15.6.6	Slave Mode	15-14

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
15.6.7	Arbitration Lost	15-14

Section 16 DEBUG SUPPORT

16.1	Signal Description	16-2
16.1.1	Breakpoint (\overline{BKPT})	16-2
16.1.1.1	Rev A Functionality.....	16-2
16.1.1.2	Rev B Enhancement.....	16-2
16.1.2	Debug Data (DDATA[3:0])	16-2
16.1.3	Development Serial Clock (DSCLK)	16-2
16.1.4	Development Serial Input (DSI)	16-2
16.1.5	Development Serial Output (DSO).....	16-2
16.1.6	Processor Status (PST[3:0])	16-2
B6.1.7	Processor Status Clock (PSTCLK)	16-2
B6.2	Real-Time Trace Support.....	16-3
6.2.1	Processor Status Signal Encoding.....	16-4
6.2.1.1	Continue Execution (PST=\$0).....	16-4
6.2.1.2	Begin Execution of an Instruction (PST=\$1)	16-4
6.2.1.3	Entry into User Mode (PST=\$3)	16-4
6.2.1.4	Begin Execution of Pulse or WDDATA Instructions (PST=\$4)	16-4
6.2.1.5	Begin Execution of Taken Branch (PST=\$5).....	16-4
6.2.1.6	Begin Execution of RTE Instruction (PST=\$7)	16-5
6.2.1.7	Begin Data Transfer (PST-\$8-\$B)	16-5
6.2.1.8	Exception Processing (PST=\$C).....	16-6
6.2.1.9	Emulator Mode Exception Processing (PST=\$D)	16-6
6.2.1.10	Process Stopped (PST=\$E)	16-6
6.2.1.11	Processor Halted (PST=\$F)	16-6
6.3	Background Debug Mode (BDM)	16-6
6.3.1	CPU Halt	16-7
6.3.2	BDM Interface	16-8
6.3.2.1	Receive Packet Format	16-9
6.3.2.2	Transmit Packet Format	16-10
6.3.3	BDM Command Set	16-10
6.3.3.1	BDM Command Set Summary	16-10
6.3.3.2	ColdFire BDM Commands.....	16-11
6.3.3.3	Command Sequence Diagram	16-12
6.3.3.4	Command Set Descriptions.....	16-14

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
16.3.3.4.1	Read A/D Register (RAREG/RDREG)	16-14
16.3.3.4.2	Write A/D Register (WAREG/WDREG)	16-15
16.3.3.4.3	Read Memory Location (READ)	16-16
16.3.3.4.4	Write Memory Location (WRITE)	16-17
16.3.3.4.5	Dump Memory Block (DUMP)	16-19
16.3.3.4.6	Fill Memory Block (FILL)	16-21
16.3.3.4.7	Resume Execution (GO)	16-23
16.3.3.4.8	No Operation (NOP)	16-23
16.3.3.4.9	Synchronize PC to the PST/DDATA Lines (SYNC_PC)	16-24
16.3.3.4.10	Read Control Register (RCREG)	16-24
16.3.3.4.11	Write Control Register (WCREG)	16-26
16.3.3.4.12	Read Debug Module Register (RDMREG)	16-26
16.3.3.4.13	Write Debug Module Register (WDMREG)	16-27
16.3.3.4.14	Unassigned Opcodes	16-28
16.4	Real-Time Debug Support	16-28
16.4.1	Theory of Operation	16-29
16.4.1.1	Emulator Mode	16-30
16.4.1.2	Debug Module Hardware	16-30
16.4.1.2.1	Reuse of Debug Model Hardware (Rev. A)	16-31
16.4.1.2.2	The New Debug Module Hardware (Rev. B)	16-31
16.4.2	Programming Model	16-31
16.4.2.1	Address Breakpoint Registers (ABLR, ABHR)	16-32
16.4.2.2	Address Attribute Trigger Register (AATR)	16-33
16.4.2.3	Program Counter Breakpoint Register (PBR, PBMR)	16-35
16.4.2.4	Data Breakpoint Register (DBR, DBMR)	16-36
16.4.2.5	Trigger Definition Register (TDR)	16-37
16.4.2.6	Configuration/Status Register (CSR)	16-40
16.4.2.7	BDM Address Attribute (BAAR)	16-43
16.4.3	Concurrent BDM and Processor Operation	16-44
16.4.4	Motorola Recommended BDM Pinout	16-44

Section 17

IEEE 1149.1 TEST ACCESS PORT (JTAG)

17.1	Overview	17-2
17.2	JTAG Signal Descriptions	17-2
17.2.1	Test Clock - (TCK)	17-3
17.2.2	Test Reset/Development Serial Clock - (TRST/DSCLK)	17-3
17.2.3	Test Mode Select/ Break Point (TMS/BKPT)	17-3
17.2.4	Test Data Input/Development Serial Input - (TDI/DSI)	17-4
17.2.5	Test Data Output/Development Serial Output - (TDO/DSO)	17-4
17.3	JTAG Register Descriptions	17-4

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
17.3.1	JTAG Instruction Shift Register	17-4
17.3.1.1	EXTEST Instruction	17-5
17.3.1.2	IDCODE	17-5
17.3.1.3	SAMPLE/PRELOAD Instruction	17-5
17.3.1.4	HIGHZ Instruction	17-6
17.3.1.5	CLAMP Instruction	17-6
17.3.1.6	BYPASS Instruction	17-6
17.3.2	IDcode Register	17-7
17.3.3	JTAG Boundary-Scan Register	17-7
17.3.4	JTAG Bypass Register	17-13
17.4	TAP Controller	17-13
17.5	Restrictions	17-14
17.6	Disabling the IEEE 1149.1 Standard Operation	17-15
17.7	MCF5307 BSDL File	17-16
17.8	Obtaining the IEEE 1149.1 Standard	17-24

Section 18 ELECTRICAL SPECIFICATIONS

Section 19 MECHANICAL DATA

19.1	Package	19-1
19.2	Pinout	19-1

Appendix A REGISTER MEMORY MAP

Appendix B MCF5307 MEMORY MAP SUMMARY

Index

LIST OF ILLUSTRATIONS

Figure Number	Title	Page Number
1-1.	MCF5307 Block Diagram	1-4
2-1.	MCF5307 Signal Interface	2-1
3-1.	Clock Generation Block Diagram	3-2
3-2.	ColdFire Enhanced Pipeline	3-3
3-3.	Pipelined Instruction Sequence	3-4
3-4.	ColdFire Multiply-Accumulate Block Diagram	3-6
3-5.	User Programming Model	3-8
3-6.	MAC Unit User Programming Model	3-10
3-7.	Exception Stack Frame Form	3-13
3-8.	Organization of Integer Data Formats in Data Registers	3-18
3-9.	Organization of Integer Data Formats in Address Registers	3-19
3-10.	Memory Operand Addressing	3-20
4-1.	Block Diagram of Phase-Locked Loop Module	4-1
4-2.	CLKIN, PSTCLK, and BCLKOBCLKO Timing	4-5
4-3.	Reset and Initialization Timing	4-6
5-1.	MCF5307 Unified Cache	5-1
5-2.	Cache Organization and Line Format	5-2
5-3.	Caching Operation	5-3
5-4.	Cache Line State Diagrams	5-15
7-1.	Signal Relationships to BCLKO	7-6
7-2.	Connections for External Memory Port Sizes	7-7
7-3.	Read Cycle Flowchart	7-8
7-4.	Basic Read Bus Cycle	7-9
7-5.	Write Cycle Flowchart	7-10
7-6.	Basic Write Bus Cycle	7-11
7-7.	Read Cycle with Fast Termination	7-12
7-8.	Write Cycle with Fast Termination	7-13
7-9.	Back-to-Back Bus Cycles	7-15
7-10.	Line Read Burst (2-1-1-1)	7-17
7-11.	Line Read Burst (3-2-2-2)	7-18
7-12.	Line Read Burst Inhibited with Fast Termination	7-19
7-13.	Line Write Burst (2-1-1-1)	7-20
7-14.	Line Write Burst (4-2-2-2) with One Wait State	7-21

LIST OF ILLUSTRATIONS (Continued)

Figure Number	Title	Page Number
7-15.	Line Write Burst Inhibited	7-22
7-16.	Interrupt-Acknowledge Cycle Flowchart	7-25
7-17.	MCF5307 Two-Wire Mode Bus Arbitration Interface	7-27
7-18.	Two-Wire Bus Arbitration with Bus Request Asserted	7-29
7-19.	Two-Wire Implicit and Explicit Bus Ownership	7-30
7-20.	MCF5307 Two-Wire Bus Arbitration Protocol State Diagram	7-31
7-21.	Three-Wire Implicit and Explicit Bus Ownership	7-35
7-22.	Three-Wire Bus Arbitration with External to Master control Bit Asserted	7-36
7-23.	MCF5307 Three-Wire Bus Arbitration Protocol State Diagram	7-38
7-24.	Master Reset Timing	7-41
7-25.	Software Watchdog Reset Timing.	7-42
8-1.	MCF5307 Embedded System Recovery from Unterminated Access	8-11
9-1.	Connections for External Memory Port Sizes	9-6
9-2.	Chip Select Module Outputs Timing Diagram	9-16
11-1.	DRAM Controller Block Diagram Bank Register Blocks	11-2
11-2.	MCF5307 SDRAM Interface	11-5
11-3.	Edge-Select Cell	11-5
11-4.	Edge-Select Cell Output Waveforms	11-6
11-5.	Basic Non-Page Mode Operation RCD=0, RNCN=1	11-17
11-6.	Basic Nonpage-Mode Operation RCD = 1, RNCN = 0	11-17
11-7.	Burst Page Mode Read Operation	11-18
11-8.	Burst Page Mode Write Operation	11-19
11-9.	Continuous Page Mode Operation	11-20
11-10.	Write Hit in Continuous Page Mode	11-21
11-11.	EDO Read Operation	11-22
11-12.	DRAM Access Delayed By Refresh	11-23
11-13.	Mode Register Set Command	11-34
11-14.	Burst Read SDRAM Access	11-36
11-15.	Burst Write SDRAM Access	11-37
11-16.	Synchronous Continuous Page Mode Access -- Read Followed by Read ..	11-38
11-17.	Synchronous Continuous Page Mode Access -- Write Followed by Read ...	11-39
11-18.	Auto-Refresh Operation	11-40
11-19.	Self Refresh Operation	11-41
12-1.	Timer Block Diagram	12-2
13-1.	DMA Signal Diagram	13-2
13-2.	Single-Address Transfers	13-4
13-3.	Dual-Address Transfer	13-5

LIST OF ILLUSTRATIONS (Continued)

Figure Number	Title	Page Number
13-4.	DMA Controller Module Channel Offsets	13-5
13-5.	DMA Controller Module Register Model Per Channel (Continued)	13-5
13-6.	External Request Timing - Cycle-Steal Mode, Single-Address Mode	13-16
13-7.	External Request Timing - Cycle-Steal Mode, Dual-Address Mode	13-16
14-1.	Simplified Block Diagram	14-1
14-2.	External and Internal Interface Signals	14-5
14-3.	Transmitter and Receiver Functional Diagram.....	14-8
14-4.	Transmitter Timing Diagram.....	14-9
14-5.	Receiver Timing Diagram.....	14-11
14-6.	Looping Modes Functional Diagram	14-14
14-7.	Multidrop Mode Timing Diagram	14-16
14-8.	UART Mode Programming Flowchart	14-36 through 14-40
15-1.	M-Bus Module Block Diagram	15-2
15-2.	M-Bus Standard Communication Protocol	15-3
15-3.	Synchronized Clock SCL	15-5
15-4.	Flow-Chart of Typical M-Bus Interrupt Routine	15-15
16-1.	Processor/Debug Module Interface.....	16-1
16-2.	Example PST/DDATA Diagram	16-5
16-3.	BDM Serial Transfer.....	16-8
16-4.	Receive BDM Packet	16-9
16-5.	Transmit BDM Packet	16-10
16-6.	Command Sequence Diagram.....	16-14
16-7.	Debug Programming Model	16-32
16-8.	Recommended BDM Connection.....	16-45
17-1.	JTAG Test Logic Block Diagram	17-2
17-2.	JTAG TAP Controller State Machine	17-14
17-3.	Disabling JTAG in JTAG Mode	17-15
17-4.	Disabling JTAG in Debug Mode.....	17-16

LIST OF ILLUSTRATIONS (Continued)

**Figure
Number**

Title

**Page
Number**

LIST OF TABLES

Table Number	Title	Page Number
2-1.	MCF5307 Signal Index.....	2-2
2-2.	Bus Cycle Size Encodings	2-4
2-3.	Bus Cycle Transfer Type Encoding	2-5
2-4.	Transfer Modifier Encodings for TT=00	2-6
2-5.	Transfer Modifier Encodings for TT=10	2-6
2-6.	Transfer Modifier Encodings for TT=11	2-6
2-7.	CLKIN Frequency.....	2-8
2-8.	BCLK/PSTCLK Divide Ratios.....	2-8
2-9.	D[6] and D[5] Selection of CS0 Port Size.....	2-9
2-10.	D[7] Selection of CS0 Automatic Acknowledge.....	2-9
2-11.	Processor Status Signal Encodings.....	2-13
3-1	Exception Vector Assignments	3-13
3-2.	Format Field Encoding	3-14
3-3.	Fault Status Encodings	3-14
3-4.	Integer Data Formats	3-18
3-5.	Effective Addressing Modes and Categories	3-21
3-6.	Notational Conventions	3-21
3-7.	Instruction Set Summary	3-23
3-8.	Misaligned Operand References	3-26
3-9.	Move Byte and Word Execution Times	3-26
3-10.	Move Long Execution Times	3-26
3-11.	MAC Move Long Instruction Execution Times	3-27
3-12.	One Operand Instruction Execution Times	3-27
3-13.	Two Operand Instruction Execution Times	3-28
3-14.	Miscellaneous Instruction Execution Times	3-29
3-15.	General Branch Instruction Execution Times	3-30
3-16.	BRA, Bcc Instruction Execution Times	3-30
3-17.	Another Table of Bcc Instruction Execution Times	3-31
4-1.	Settings of PLLIPL[2:0] and Interrupts that wake up Core.....	4-3
4-2.	CLKIN Frequency	4-4
4-3.	BCLKO/PSTCLK Divide Ratios	4-4
5-1.	Cache Line State Transitions	5-15
6-1.	Examples of Typical RAMBAR Settings	6-4

LIST OF TABLES (Continued)

Figure Number	Title	Page Number
7-1.	ColdFire Bus Signal Summary	7-2
7-2.	Bus Cycle Size Encoding	7-3
7-3.	Bus Cycle Transfer Type Encoding	7-4
7-4.	ColdFire Bus Signal Summary	7-5
7-5.	Accesses by Matches in CS and DRAM Control Registers	7-8
7-6.	Allowable Line Access Patterns	7-16
7-7.	ColdFire Bus Signal Summary	7-26
7-8.	MCF5307 Two-Wire Bus Arbitration Protocol Transition Conditions	7-32
7-9.	MCF5307 Two-Wire Arbitration Protocol State Diagram	7-32
7-10.	MCF5307 Three-Wire Bus Arbitration Protocol Transition Conditions	7-38
7-11.	MCF5307 Three-Wire Arbitration Protocol State Diagram	7-39
8-1.	SIM Memory Map	8-2
8-2.	Interrupt Control Register Memory Map	8-5
8-3.	Interrupt Priority Scheme	8-7
8-4.	Signals Selected with IRQPAR	8-10
8-5.	SWT Timeout Period	8-13
8-6.	Settings of PLLIPL[2:0] and Interrupts that Awaken Core	8-15
8-7.	Default Bus Master Selected with PARK[1:0]	8-17
8-8.	Park on Master 1 Priority (PARK[1:0] = 00)	8-17
8-9.	Park on Alternate Master 2 Priority (PARK[1:0] = 01)	8-17
8-10.	Park on alternate master 3 priority (PARK[1:0] = 10)	8-18
8-11.	Park on current master priority (PARK[1:0] = 11)	8-18
8-12.	Signals Selected with PAR	8-19
9-1.	Data Bus Byte Write Enable Signals	9-2
9-2.	Accesses by Matches in CS Control Registers	9-5
9-3.	D[6] and D[5] Selection of $\overline{CS0}$ Port Size	9-7
9-4.	D[7] Selection of $\overline{CS0}$ Automatic Acknowledge	9-7
9-5.	Memory Map of Chip-Select Registers	9-8
9-6.	Chip-Select 2-7 Decoding	9-10
10-1.	Signals Selected with PAR	10-1
11-1.	DRAM Controller Memory Map	11-6
11-2.	RRA Encoding	11-7
11-3.	RRP Encoding	11-7
11-4.	RE Encoding	11-8
11-5.	CAS Encoding	11-9
11-6.	RAS Precharge (RP) Encoding	11-9
11-7.	RNCN Encoding	11-9
11-8.	RCD Encoding	11-10

LIST OF TABLES (Continued)

Figure Number	Title	Page Number
11-9.	EDO Encoding	11-10
11-10.	PS encoding	11-10
11-11.	PM Encoding	11-11
11-12.	Base Address Mask Encoding	11-11
11-13.	WP Encoding	11-12
11-14.	Address Modifier Bit Definitions	11-12
11-15.	Address Modifier Bit Encoding	11-12
11-16.	Valid Bit Encoding	11-13
11-17.	Address Muxing Scheme	11-13
11-18.	DRAM Addressing for Byte Wide Memories	11-14
11-19.	DRAM Addressing for 16-Bit Wide Memories	11-15
11-20.	DRAM Addressing for 32-Bit Wide Memories	11-16
11-21.	SO Bit Encoding	11-24
11-22.	NAM Bit Encoding	11-24
11-23.	COC Bit Encoding	11-24
11-24.	IS Bit Encoding	11-25
11-25.	RTIM Encoding	11-25
11-26.	RE Encoding	11-26
11-27.	Synchronous CASL Encoding	11-27
11-28.	IMRS Bit Encoding	11-27
11-29.	CBM Encoding	11-28
11-30.	PS Encoding	11-28
11-31.	IP Bit Encoding	11-29
11-32.	Synchronous PM Encoding	11-29
11-33.	Address Setup for 2M X 2 Bank X 4-Bit SDRAM X 2 as 8-Bit Port	11-30
11-34.	Address Setup for 2M X 2Bank X 4-Bit SDRAM X 4 as 16-Bit Port	11-30
11-35.	Address Setup for 2M X 2 Bank X 4-Bit SDRAM X 8 as 32-Bit Port	11-31
11-36.	Address Setup for 512 K X 2 Bank X 16-Bit SDRAM X 1 as 16-Bit Port	11-31
11-37.	Address Setup for 128 K X 2 Bank X 16-Bits SDRAM X1 as 16-Bit Port	11-32
11-38.	Address Setup for 1M X 4 Bank X 16-Bits SDRAM X 2 as 32-Bit Port	11-32
12-1.	Programming Model for Timers	12-4
12-2.	Calculated Time-out Values (45MHz System Bus Clock).	12-7
13-1.	DMA Signals.	13-3
13-2.	BWC Encoding	13-9
13-3.	SSIZE Encoding	13-10
13-4.	DSIZE Encoding	13-10
14-1.	UART Module Programming Model	14-18
14-2.	PMx and PT Control Bits.....	14-20
14-3.	B/Cx Control Bits.....	14-20

LIST OF TABLES (Continued)

Figure Number	Title	Page Number
14-4.	CMx Control Bits	14-21
14-5.	SBx Control Bits	14-22
14-6.	RCS[3:0] Control Bits	14-24
14-7.	TCS[3:0] Control Bits	14-24
14-8.	MISC[2:0] Control Bits	14-25
14-9.	TC[1:0] Control Bits	14-27
14-10.	RCx Control Bits	14-28
14-11.	Timer Mode and Source Select Bits	14-30
15-1.	M-Bus Interface Programmer's Model	15-6
15-2.	MBUS Prescaler Values	15-7
16-1.	Processor Status Encoding	16-3
16-2.	CPU-Generated Message Encoding	16-9
16-3.	BDM Command Summary	16-11
16-4.	BDM Size Field Encoding	16-12
16-5.	Control Register Map	16-25
16-6.	Definition of DRc Encoding - Read	16-27
16-7.	Definition of DRc Encoding - Write	16-28
16-8.	DDATA[3:0], CSR[31:28] Breakpoint Response	16-29
16-9.	Shared BDM/Breakpoint Hardware	16-31
16-10.	Access Size and Operand Data Location	16-37
17-1.	JTAG Pin Descriptions	17-3
17-2.	JTAG Instructions	17-5
17-3.	Boundary-Scan Bit Definitions	17-8
18-1.	Maximum Ratings	18-1
18-2.	Operating Temperature	18-1
18-3.	DC Electrical Specifications (Vcc = 3.3V dc +/- 0.3V dc)	18-2
18-4.	Clock Timing Specification	18-3
18-5.	Reset Timing Specification	18-4
18-6.	Input AC Timing Specification	18-5
18-7.	Output AC Timing Specification	18-5
18-8.	Debug AC Timing Specification	18-8
18-9.	Timer Module AC Timing Specification	18-9
18-10.	UART Module AC Timing Specifications	18-10
18-11.	M-Bus Input Timing Specifications Between SCL and SDA	18-11
18-12.	M-Bus Output Timing Specifications Between SCL and SDA	18-11
18-13.	M-Bus Timing Specifications Between BCLKO and SCL, SDA	18-13
18-14.	General Purpose I/O Port AC Timing Specifications	18-14
18-15.	IEEE 1149.1 (JTAG) AC Timing Specifications	18-15

LIST OF TABLES (Continued)

Figure Number	Title	Page Number
19-1.	Pinout Table (Left, top to bottom).....	19-2
19-2.	Pinout Table (bottom, left to right).....	19-4
19-3.	Pinout Table (right, bottom to top).....	19-6
19-4.	Pinout Table (top, right to left).....	19-8
A-1.	Register Memory Map	A-1
B-1.	MCF5307 User Programming Model	B-1
B-2.	Summary Chart of MCF5307 Internal CPU Memory Map	B-5

LIST OF TABLES (Continued)

**Figure
Number**

Title

**Page
Number**

SECTION 1 INTRODUCTION

The MCF5307 integrated microprocessor combines a ColdFire® processor core with a Multiply-Accumulate (MAC) unit, DRAM controller, DMA controller, timers, parallel and serial interfaces, and system integration. Designed for embedded control applications, the ColdFire core delivers enhanced performance while maintaining low system costs. Performance boosts are supplied to the clock-doubled core through the on-chip, 8Kbyte unified cache and 4K SRAM, which provide pipelined, single-cycle access to critical code and data. The integrated MAC module enhances the device's functionality through support of high-speed, complex, arithmetic operations; it can execute 16x16 multiplies with a 32-bit accumulate in a single cycle and 32x32 multiplies with 32-bit accumulate. The MCF5307 processor greatly reduces the time required for system design and implementation by combining common system functions on chip and providing glueless interfaces to 8-, 16-, and 32-bit DRAM, SRAM, ROM, FLASH, and I/O devices. Support for EDO DRAM and synchronous DRAM is also present.

The revolutionary ColdFire microprocessor architecture gives cost-sensitive, high-volume markets new levels of price and performance. Based on the concept of variable-length RISC instruction-set technology, ColdFire 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, Motorola incorporated RISC architecture for peak performance and a simplified version of the variable-length instruction set found in the M68000 Family for code density. The MCF5307 is the first of the ColdFire family to contain the Version 3, clock-doubled core. Increasing the internal speed of the core allows higher performance, while providing the customer with a workable low-speed external system interface.

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

The integrated peripheral functions provide high performance and flexibility. The DRAM controller can interface with up to 256 MBytes of DRAM and supports bursting and page-mode operations. In addition, this DRAM controller can connect to both extended-data-out (EDO) DRAMs and synchronous DRAMs.

Other nonmemory interfaces available include a programmable full-duplex DUART and M-Bus (I²C interface¹) module. Four channels of DMA allow for fast data transfer using a

¹ I²C is a proprietary Philips bus

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 both background-debug mode and real-time trace. This interface is present in all ColdFire-based processors and allows common emulator support across the entire ColdFire family.

The primary features of the MCF5307 integrated processor include the following:

- ColdFire Processor Core
 - Variable-length RISC, clock-doubled Version 3 microprocessor core
 - Eight-stage pipeline separated by a 384-bit queuing buffer
 - 32-bit internal address bus with up to 4 Gbytes of linear address space
 - 32-bit data bus
 - 16 user-visible 32-bit wide general-purpose registers
 - Hardware Integer Divide Unit
 - Supervisor/user modes for system protection
 - Vector base register to relocate exception-vector table
 - Optimized for high-level language constructs
- Multiply and Accumulate Unit
 - Provides high-speed, complex arithmetic processing for signal processing applications
 - 1 clock issue with 3-stage execute pipeline
 - Supports both 16x16 multiplies and 32x32 multiplies with 32-bit accumulate
- 8Kbyte Unified Cache
 - Four-way set associative organization with write through and copy back modes
 - Clock-doubled to match microprocessor core speed
 - Provides fast access to critical code and data
- 4Kbyte SRAM
 - Single-cycle access to code or data
 - Restart model maintains code in memory
- DMA Controller
 - Four fully programmable channels
 - Supports dual address and single address transfers with 32-bit data capability
 - Two address pointers that can increment or remain constant
 - 16-bit transfer counter
 - Operand packing and unpacking supported
 - Auto-alignment transfers supported for efficient block movement
 - Supports bursting and cycle steal
 - Provides two clock cycle internal access

- DRAM Controller (DRAMC)
 - Supports up to 256 MBytes of DRAM
 - Programmable refresh timer provides CAS-before-RAS refresh
 - Support for 2 separate memory banks
 - Support for extended data out DRAMs and Synchronous DRAMs
- Dual Universal Synchronous/Asynchronous Receiver/Transmitter (DUART)
 - Full duplex operation
 - Flexible baud-rate generator
 - Modem control signals available ($\overline{\text{CTS}}$, $\overline{\text{RTS}}$)
 - Processor-interrupt capability
- Dual 16-Bit General-Purpose Multimode Timers
 - 8-bit prescaler
 - Timer input and output pins
 - Processor-interrupt capability
 - 22 ns resolution
- Motorola Bus (M-Bus) Module
 - Interchip bus interface for EEPROMs, LCD controllers, A/D converters, keypads
 - Fully compatible with industry-standard I²C Bus
 - Master and slave modes support multiple masters
 - Automatic interrupt generation with programmable level
- System Interface
 - Glueless bus interface with chip selects and DRAMC support for interface to
 - 8-, 16-, and 32-bit for DRAM, SRAM, ROM, FLASH, and I/O devices
 - 8 chip-select signals; 2 that are fully programmable with base address registers, 6 at an offset of one base address register
 - Programmable wait states and port sizes
 - Programmable interrupt controller
 - Low interrupt latency
 - 4 external interrupt request inputs
 - Programmable autovector generator
 - 16-Bit general-purpose I/O interface
 - IEEE 1149.1 test (JTAG) module
- System Debug Support
 - Real-time trace for determining dynamic execution path
 - Background debug mode (BDM) for debug features while halted
 - Real time debug support, including three user-visible hardware breakpoint registers
- Clock-multiplied PLL
- 70 MIPS at 45 MHz (90 MHz clock-doubled internally)

INTRODUCTION

- Offered at 90MHz core clock with system bus capability of 45, 30, and 22.5 MHz (Operating temperature: 0 to +70 C)
- Offered at 66MHz core clock with system bus capability of 33, 22, and 16.5 MHz (Operating temperature: 0 to 70 C and -40 to +85 C)
- Fully static 3.3-volt operation with 5V tolerant I/O pads
- 208 QFP package

1.1 OVERVIEW

Figure 1-1 is a block diagram of the MCF5307 processor. The paragraphs that follow provide an overview of the integrated processor.

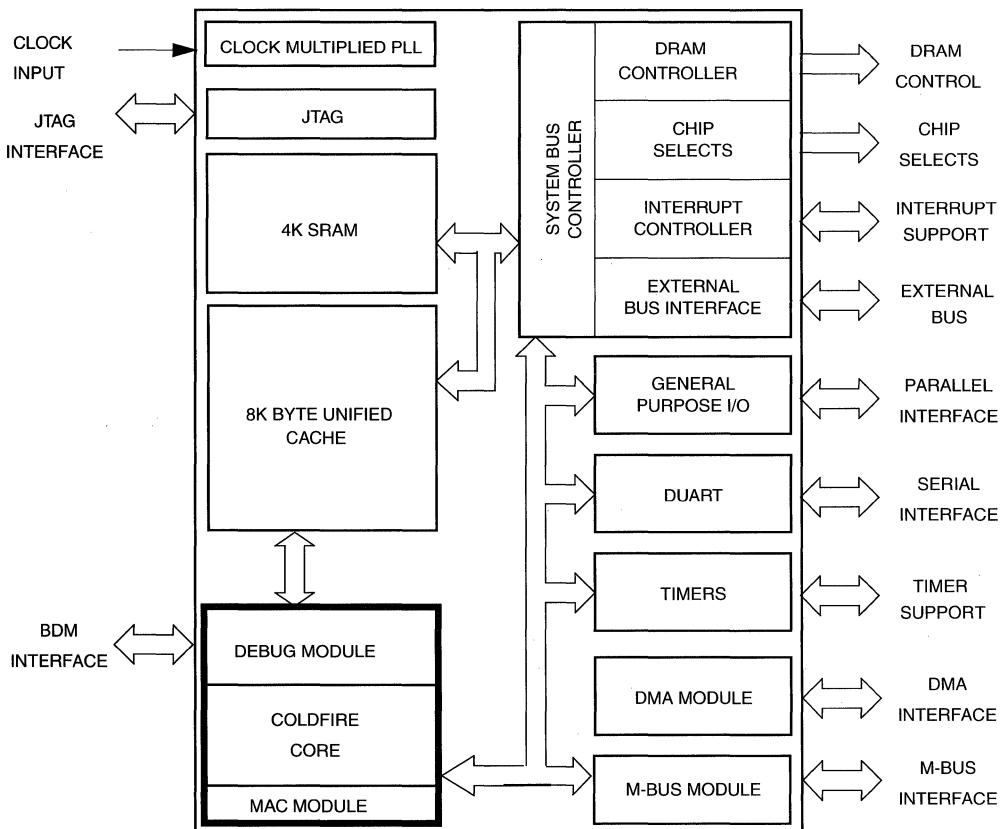


Figure 1-1. MCF5307 Block Diagram

1.1.1 ColdFire Processor Core

The ColdFire processor Version 3 core consists of two independent, decoupled pipeline structures to maximize performance while minimizing core size. The instruction fetch

pipeline (IFP) is a six-stage pipeline for prefetching instructions. The IFP contains logic for branch prediction. The prefetched instruction stream is then gated into the two-stage operand execution pipeline (OEP), which decodes the instruction, fetches the required operands and then executes the required function. Because the IFP and OEP pipelines are decoupled by an instruction buffer that serves as a FIFO queue, the IFP can prefetch instructions in advance of their actual use by the OEP, thereby minimizing time stalled waiting for instructions. The OEP is implemented in a two-stage pipeline featuring a traditional RISC datapath with a dual-read-ported register file feeding an arithmetic/logic unit (ALU).

1.1.2 Multiply and Accumulate (MAC) Module

The MAC unit provides a common set of DSP operations and enhances the integer multiply instructions in the ColdFire architecture. It provides functionality in three related areas: faster signed and unsigned integer multiplies; new multiply-accumulate operations supporting signed and unsigned operands; and new miscellaneous register operations. Multiplies of 16x16 and 32x32 with 32-bit accumulates are supported. The MAC has a single clock issue for 16x16 multiplies and implements a 3-stage execution pipeline.

1.1.3 8 Kbyte Unified Cache

The MCF5307 processor contains a high-performance nonblocking, 8-Kbyte, four-way set associative unified (instruction and data) cache. The cache improves system performance by providing low latency data to the processor core. This decouples processor performance from system memory performance and increases bus availability for alternate bus masters.

The nonblocking design of the MCF5307 cache services read hits or write hits from the processor while a line fill (caused by a cache allocation) is in progress. The cache can operate in either writethrough or copyback modes with no write-allocates for misses to writethrough memory. Cache design allows the MCF5307 to achieve 75 MIPS.

Each cache line contains 16 bytes of data, an address tag and control bits. A bursting interface for 32-, 16-, and 8-bit port sizes to quickly fill cache lines is supported. A unique 128-bit store buffer is implemented with the cache to further decouple store operation from processor execution.

1.1.4 Internal 4 Kbyte SRAM

The 4-Kbyte on-chip SRAM provides one clock-cycle access for the ColdFire core. This SRAM can store processor stack and critical code or data segments to maximize performance.

1.1.5 DRAM Controller

The MCF5307 DRAM controller provides a glueless interface for up to two banks of DRAM, each of which can be from 128 Kbytes up to 256 MBytes in size. The controller supports an 8-, 16-, or 32-bit data bus. A unique addressing scheme allows for increases in system memory size without rerouting address lines and rewiring boards. The controller operates in regular mode, or page mode, and supports extended-data-out (EDO) DRAMs and

synchronous DRAMs. At a 45MHz external bus speed, the DRAM controller supports DRAMs with access times as fast as 22 ns.

1.1.6 DMA Controller

MCF5307 provides four fully programmable DMA channels for quick data transfer. Single and dual address mode is supported with the ability to program bursting and cycle steal. Data is transferred as 32-bits and packing and unpacking is supported.

1.1.7 DUART Module

A full duplex DUART module contains an on-chip baud-rate generator, which provides both standard and nonstandard baud rates. 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. Four-byte receive buffers and two-byte transmit buffers minimize CPU service calls. The DUART module also provides several error-detection and maskable-interrupt capabilities. Modem support includes request-to-send (RTS) and clear-to-send (CTS) lines.

The system clock provides the clocking function via a programmable prescaler. You can select full duplex, autoecho loopback, local loopback, and remote loopback modes. The programmable DUART can interrupt the CPU on various normal or error-condition events.

1.1.8 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 clock. The programmable timer-output pin generates either an active-low pulse or toggles the output.

1.1.9 Motorola Bus (M-Bus) Module

The M-Bus interface is a two-wire, bidirectional serial bus that exchanges data between devices and is compliant with the I²C Bus standard. The M-Bus 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. Bus capacitance and the number of unique addresses limit the maximum communication length and the number of devices that can be connected.

1.1.10 System Interface

The MCF5307 processor provides a glueless interface to 8-, 16-, and 32-bit port size SRAM, ROM, and peripheral devices with independent programmable control of the assertion and negation of chip-selects and write-enables. The MCF5307 also supports bursting ROMs.

1.1.10.1 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 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 a DMA device, is glueless with arbitration logic internal to the MCF5307 processor. Multiple-master arbitration is also available with some external arbitration logic.

1.1.10.2 CHIP-SELECTS. Eight chip-select outputs (two that are programmable with base address registers, six at an offset of one base address register) provide signals that enable external memory and peripheral circuits for automatic wait-state insertion. These signals also interface to 8-, 16-, or 32-bit ports. The base address and access permissions are programmable with configuration registers.

1.1.10.3 16-BIT PARALLEL-PORT INTERFACE. A 16-bit general-purpose programmable parallel port serves as either an input or an output on a bit-by-bit basis.

1.1.10.4 INTERRUPT CONTROLLER. The interrupt controller provides user-programmable control of 11 internal peripheral interrupts and implements 4 external fixed interrupt request pins. You can program each internal interrupt to any one of 7 interrupt levels and 2 priority levels within each of these levels. The external interrupt request pins can be programmed to levels 1,3,5, and 7 or levels 2, 4, 6, and 7. Autovector capability is available for internal and external interrupts.

1.1.10.5 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 Joint Test Action Group, or JTAG. For more information, refer to the IEEE 1149.1A standard.

1.1.11 System Debug Interface

The ColdFire processor core debug interface supports real-time trace and debug, plus background-debug mode. A four-pin background-debug mode (BDM) interface provides system debug. The BDM is a proper subset of the BDM interface provided on Motorola's 683XX family of parts.

In real-time trace, four status lines provide information on processor activity in real time (PST pins). A four-bit wide debug data bus (DDATA) displays operand data and change-of-flow addresses, which helps track the machine's dynamic execution path.

SECTION 2 SIGNAL DESCRIPTION

2.1 INTRODUCTION

This section describes the MCF5307 input and output signals. The descriptions as shown in Table 2-1 are grouped according to relevant functionality. Figure 2-1 displays the block diagram of the MCF5307 device along with the signal interface.

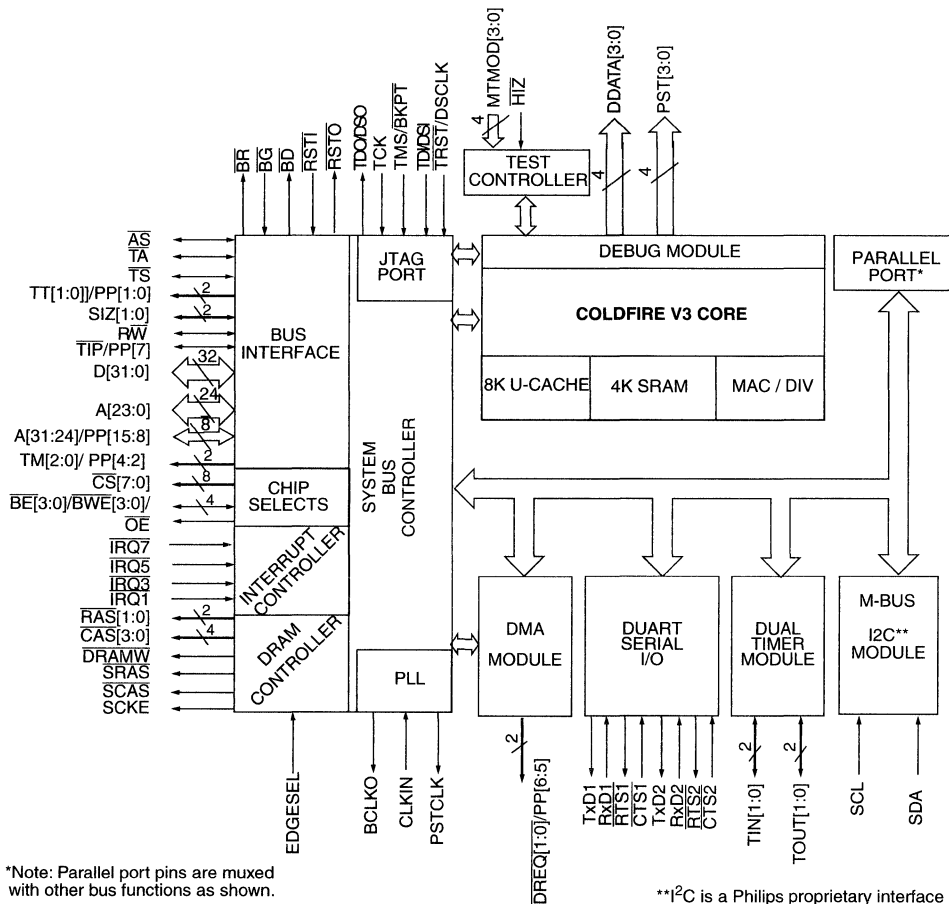


Figure 2-1. MCF5307 Block Diagram

Table 2-1. MCF5307 Signal Index

SIGNAL NAME	MNEMONIC	FUNCTION	INPUT/ OUTPUT
Address	A[31:0]	All 32 bits of the address bus. For external interrupt request, A[4:2] are used to indicate the interrupt level	In/Out
Data	D[31:0]	Data bus used to transfer byte, word, or longword data	In/Out
Read/Write	R/W	Identifies read and write transfers.	In/Out
Size	SIZ[1:0]	Indicates the data transfer size.	In/Out
Transfer Start	TS	Indicates the beginning of a bus transfer.	In/Out
Address Strobe	AS	Indicates a bus cycle has been initiated and address is stable	In/Out
Transfer Acknowledge	TA	Assertion terminates transfer synchronously.	In/Out
Transfer In Progress	TIP	Indicates a bus cycle is in progress of running	Out
Transfer Type	TT[1:0]	Indicates the transfer type: normal, CPU space or emulator mode	Out
Transfer Modifier	TM[2:0]	Provides supplemental information for each transfer type.	Out
Interrupt Request	IRQ7, IRQ5, IRQ3, IRQ1	Four external interrupts are set to default levels 1,3,5,7; user-alterable. See subsection 2.3.1.	In
Bus Request	BR	Indicates processor requires bus mastership.	Out
Bus Grant	BG	Asserted by arbiter to grant mastership to processor.	In
Bus Driven	BD	Indicates processor is currently driving the bus.	Out
Reset In	RSTI	Processor Reset Input	In
Clock Input	CLKIN	Input used to clock internal logic.	In
System Bus Clock Out	BCLKO	Output used as bus system clock reference	Out
Reset Out	RSTO	Processor Reset Output	Out
Chip-Selects[7:0]	CS[7:0]	Enables peripherals at programmed addresses. CS[1:0], CS[7] can provide IACK during an interrupt acknowledge cycle. CS[0] provides boot ROM selection	Out
Byte Enable[3:0]/ Byte Write Enable[3:0]	BE[3:0]/ BWE[3:0]	Byte enables select individual bytes in memory	Out
Output Enable	OE	Output enable for chip-select read cycles	Out
Row Address Strobe	RAS[1:0]	Provide row address strobe timing for external DRAM	Out
Column Address Strobe	CAS[3:0]	Provide column address strobe timing for external DRAM	Out
DRAM Write	DRAMW	Indicates DRAM write signal is in process	Out
Sync Row Addr Strobe	SRAS	Row address strobe timing for external SDRAM	Out
Sync Column Addr Strobe	SCAS	Column address strobe timing for external SDRAM	Out
Sync Clock Enable	SCKE	Clock enable for external SDRAM	Out
Sync Edge Select	EDGESEL	Timing select for external SDRAM	In
DMA Request	DREQ[1:0]	External DMA transfer request	In
Receive Data	RxD[1:0]	Signal is receive serial data input for DUART	In
Transmit Data	TxD[1:0]	Signal is transmit serial data output for DUART	Out
Request-To-Send	RTS[1:0]	DUART signals a ready to receive data query	Out
Clear-To-Send	CTS[1:0]	Signals to DUART that data can be transmitted to peripheral	In
Timer Input	TIN[1:0]	Provides clock input to timer or provides trigger to timer value capture logic	In
Timer Output	TOUT[1:0]	Capable of output waveform or pulse generation	Out
Parallel Port	PP[15:0]	Interfaces with I/O	In/Out
Serial Clock Line	SCL	Clock signal for M-Bus module operation	In/Out
Serial Data Line	SDA	Serial data port for M-Bus module operation	In/Out

Table 2-1. MCF5307 Signal Index (Continued)

SIGNAL NAME	MNEMONIC	FUNCTION	INPUT/ OUTPUT
Motorola Test Mode	MTMOD[3:0]	Selects whether processor is in functional or emulator mode	In
High Impedance	HIZ	Assertion three-states all output signal pins.	In
Processor Clock Out	PSTCLK	Output clock used for processor debug DDATA, PST	Out
Debug Data	DDATA[3:0]	Displays captured processor data and break-point status.	Out
Processor Status	PST[3:0]	Indicates internal processor status.	Out
Test Clock	TCK	Clock signal for IEEE 1149.1 JTAG.	In
Test Reset/Development Serial Clock	TRST/DSCLK	Asynchronous reset for JTAG controller. Clock input for debug module.	In
Test Mode Select/ Break Point	TMS/BKPT	Multiplexed signal that is TMS in JTAG mode and a hardware break-point in debug mode.	In
Test Data Input / Development Serial Input	TDI/DSI	Multiplexed serial input for the JTAG or background debug module.	In
Test Data Output/Development Serial Output	TDO/DSO	Multiplexed serial output for the JTAG or background debug module.	Out

2.2 MCF5307 BUS SIGNALS

These signals provide the external bus interface to the MCF5307 device.

2.2.1 Address Bus

The address bus provides the address of the byte or most significant byte of the word or longword being transferred. The address lines also serve as the DRAM address pins. When an external master is using the DRAM controller or Chip-Select module, the external master drives the address bus and asserts \overline{TS} or \overline{AS} to indicate a bus cycle is starting. During an interrupt acknowledge access, the lower address lines, A[4:2], indicate the interrupt level being acknowledged.

2.2.1.1 ADDRESS BUS - (A[23:0]). The lower 24 bits of the address bus become valid when \overline{TS} is asserted. Bits A[4:2] indicate the interrupt level during interrupt acknowledge cycles.

2.2.1.2 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 Pin Assignment Register (PAR) in the SIM determines the function of each of these eight multiplexed pins. You can program these on a bit-by-bit basis.

A[31:24]

When any of these pins are enabled in the PAR, they represent the most significant bits of the address bus. As much as 4 Gbytes of memory is available when all of these pins are programmed as address signals.

PP[15:8]

When any of these pins are enabled in the PAR, they represent the most significant bits of the parallel port.

2.2.2 Data Bus (D[31:0])

The data bus is a bidirectional, non-multiplexed bus. Data is latched on the bus on the rising clock edge. When interfacing with external memory or peripherals, the data bus port width, wait states, and internal termination are initially defined by D[7:0] during reset. You can program the port width for each chip-select and DRAM bank. The data bus uses the default configuration if none of the chip-selects or DRAM bank match the address decode. The default configuration is 32-bit port, with external termination, and burst inhibited. The data bus transfers byte, word, or longword-sized data. All 32 bits of the data bus are driven during writes, regardless of port width or operand size.

2.2.3 Read/Write - ($\overline{R/W}$)

When the MCF5307 processor is the bus master, it drives the $\overline{R/W}$ 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 alternate master access.

2.2.4 Size - (SIZ[1:0])

When it is the bus master, the MCF5307 processor outputs these signals to indicate the requested data transfer size. Table 2-2 shows the definition of the bus request size encodings. When the MCF5307 device is not the bus master, these signals function as inputs.

Table 2-2. Bus Cycle Size Encoding

SIZ[1:0]	Port Size
00	Longword
01	Byte
10	Word
11	Line

2.2.5 Transfer Start - (\overline{TS})

The MCF5307 processor asserts this signal during the first clock cycle when address and attributes (TM, TT, TM, \overline{TIP} , $\overline{R/W}$, size) are valid address is driven on A[31:0] and is negated in the following clock cycle. When the MCF5307 device is not the bus master, \overline{TS} is an input.

2.2.6 Address Strobe - (\overline{AS})

Address Strobe is an active-low signal that indicates the start of a bus cycle and the address is stable. 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 device is not the bus master, \overline{AS} is an input.

2.2.7 Transfer Acknowledge - ($\overline{\text{TA}}$)

When the MCF5307 device is master, the external system drives this input signal to terminate the bus transfer. The bus will continue to be driven until this synchronous signal is asserted. For write cycles, the processor will continue to drive data one clock after $\overline{\text{TA}}$ is asserted. During read cycles, the peripheral must continue to drive data until $\overline{\text{TA}}$ is recognized. The $\overline{\text{TA}}$ signal may be tied low if all bus cycles can support fast termination. Note that $\overline{\text{TA}}$ cannot be tied low if external masters are present. The MCF5307 device will drive $\overline{\text{TA}}$ if an external master requests it.

2.2.8 Transfer In Progress - ($\overline{\text{TIP/PP[7]}}$)

This multiplexed pin can serve as the transfer in progress output, or as one bit of the parallel port. Programming the Pin Assignment Register (PAR) in the SIM determines the function of this pin.

This output is asserted to indicate that a bus transfer is in progress and is negated during idle bus cycles if the bus is still granted to the processor. Note that $\overline{\text{TIP}}$ is held asserted on back-to-back bus cycles.

2.2.9 Transfer Type - (TT[1:0]/PP[1:0])

These multiplexed pins can serve as the transfer type outputs, or as two bits of the parallel port. Programming the Pin Assignment Register (PAR) in the SIM determines the function of each of these two multiplexed pins. The pins are programmable on a bit-by-bit basis.

When the MCF5307 is the bus master, it outputs these signals. They indicate the type of access for the current bus access. Table 2-3 shows the definition of the encodings.

Table 2-3. Bus Cycle Transfer Type Encoding

TT[1:0]	TRANSFER TYPE
0 0	Normal Access
0 1	Alternate Master Access
1 0	Emulator Access
1 1	CPU Space or Interrupt Acknowledge

2.2.10 Transfer Modifier - (TM[2:0]/PP[4:2])

These multiplexed pins can serve as the transfer type outputs, or as three bits of the parallel port. Programming the Pin Assignment Register (PAR) in the SIM determines the function of each of these three multiplexed pins (which are programmable on a bit-by-bit basis).

These input signals provide supplemental information for each transfer type (see Table 2-4). For emulation transfers, the TM signals indicate user or data transfer types (see Table 2-5). For CPU space transfers, the TM signals are low (see Table 2-6). For interrupt acknowledge transfers, the TM signals carry the interrupt level being acknowledged (see Table 2-6). See the alternate master section for information on TM decoding.

Table 2-4. Transfer Modifier Encodings for TT = 00

TM[2:0]	TRANSFER MODIFIER
000	Cache Push Access
011 - 100, 111	Reserved
001	User Data Access
010	User Code Access
101	Supervisor Data Access
110	Supervisor Code Access

Table 2-5. Transfer Modifier Encodings for TT = 10

TM[2:0]	TRANSFER MODIFIER
000 - 100, 111	Reserved
101	Emulator Mode Data Access
110	Emulator Mode Code Access

Table 2-6. Transfer Modifier Encodings for TT = 11

TM[2:0]	TRANSFER MODIFIER
000	CPU Space
001	Interrupt Level 1 Acknowledge
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

2.3 INTERRUPT CONTROL SIGNALS

These pins supply the external interrupt level to the MCF5307 device.

2.3.1 Interrupt Request - ($\overline{\text{IRQ7}}$, $\overline{\text{IRQ5}}$, $\overline{\text{IRQ3}}$, $\overline{\text{IRQ1}}$)

You can use these four pins as pre-defined interrupt request signals ($\overline{\text{IRQx}}$). The $\overline{\text{IRQx}}$ pins are set to default interrupt request levels for users, corresponding to levels 1, 3, 5, and 7 bus control signals. Programming the Pin Assignment Register (PAR) in the SIM determines the function of this pin. In the Pin Assignment Register, you can redefine the interrupt levels for $\overline{\text{IRQ5}}$, $\overline{\text{IRQ3}}$, and $\overline{\text{IRQ1}}$ (on a bit-by-bit basis).

2.4 BUS ARBITRATION SIGNALS

These pins provide the external bus arbitration control for the MCF5307 device.

2.4.1 Bus Request - ($\overline{\text{BR}}$)

This output signal indicates to an external arbiter that the processor needs to become bus master for one or more bus cycles. $\overline{\text{BR}}$ is negated when the MCF5307 processor begins an access to the external bus with no other internal accesses pending, and $\overline{\text{BR}}$ remains negated until another internal request occurs.

2.4.2 Bus Grant - ($\overline{\text{BG}}$)

An external arbiter asserts this input signal to indicate that the MCF5307 device can control the bus at the next rising edge of BCLKO. When the arbiter negates $\overline{\text{BG}}$, the MCF5307 processor must relinquish the bus as soon as the current transfer is complete. The external arbiter must not grant the bus to any other master until both $\overline{\text{BD}}$ and $\overline{\text{BG}}$ are negated.

2.4.3 Bus Driven - ($\overline{\text{BD}}$)

The MCF5307 device asserts this output signal to indicate that it is the current master when it is driving the bus. If the MCF5307 device is the bus master but is not using the bus, the $\overline{\text{BD}}$ signal is not asserted. If the MCF5307 processor loses bus mastership during a bus transfer, it will complete the last transfer of the current access, negate $\overline{\text{BD}}$, and three-state all bus signals on the rising edge of BCLKO. If the MCF5307 processor loses bus mastership during an idle clock cycle, it will three-state all bus signals on the rising edge of BCLKO.

2.5 CLOCK AND RESET SIGNALS

These signals provide the external system interface for the MCF5307 device.

2.5.1 Reset - ($\overline{\text{RSTI}}$)

Asserting $\overline{\text{RSTI}}$ will cause the MCF5307 processor to enter reset exception processing. When $\overline{\text{RSTI}}$ is recognized, the address bus, data bus, TT, SIZ, R/W, AS and TS will be three-stated; BR and BD will be negated.

2.5.2 Clock Input - (CLKIN)

CLKIN is the MCF5307 input clock frequency to the on-board phase-locked loop-clock generator. CLKIN is used to internally clock or sequence the MCF5307 internal bus interface

and slave module logic.

2.5.3 System Bus Clock Output - (BCLKO)

This output signal is generated by the internal PLL, and is the system bus clock output used as the bus timing reference by the external devices. BCLKO is a 1/2, 1/3, or 1/4 of the processor clock.

2

2.5.4 Reset Out - ($\overline{\text{RSTO}}$)

Following the assertion of $\overline{\text{RSTI}}$, the PLL will lose its lock temporarily. During this time, the $\overline{\text{RSTO}}$ signal will assert. When the PLL regains lock, the $\overline{\text{RSTO}}$ signal will again negate. You can use this signal to reset external devices.

2.5.5 Frequency Control PLL - $\text{FREQ}[1:0]$

This 2 bit input bus indicates the frequency range of CLKIN. These signals are multiplexed with D[3:2] and are sampled during the assertion of $\overline{\text{RESET}}$. These signals indicate to the PLL what frequency range it will be operating, as indicated in Table 2-7 shown below. Note that these signals do not affect the frequency, but are required to set up the analog PLL.

Table 2-7. CLKIN Frequency

FREQ[1:0] / D[3:2]	FREQUENCY OF CLKIN (MHZ)
00	16.6 - 27.999
01	28 - 38.999
10	39 - 45
11	NOT USED

2.5.6 Divide Control PCLK to BCLK - $\text{DIVIDE}[1:0]$

This 2-bit input bus indicates the BCLK/PSTCLK ratio. These signals are sampled during the assertion of $\overline{\text{RESET}}$ and indicate the ratios as indicated in Table 2-8 shown below.

Table 2-8. BCLK/PSTCLK Divide Ratios

DIVIDE[1:0] / D[1:0]	RATIO OF BCLK/PSTCLK
01	NOT USED
10	1/2
11	1/3
00	1/4

2.6 CHIP-SELECTS

The MCF5307 device provides eight programmable chip-selects that can directly interface with SRAM, EPROM, EEPROM, and peripherals.

2.6.1 Chip-Selects - ($\overline{CS}[7:0]$)

You can program each chip-select for an address location as well as for masking capabilities, port size and burst-capability indication, wait-state generation, and internal/external termination. A reset disables all chip-selects. $\overline{CS}[0]$ is also unique because it can function at reset as a global chip-select that allows you to select boot ROM at any defined address space. $\overline{CS}[0]$ is the only chip-select initialized out of reset. Port size and termination (internal vs. external) for $\overline{CS}[0]$ are configured by the levels on D[7:0] when \overline{RSTI} negates.

2.6.2 Chip-Select Config - ($CS_CONF[2:0]$)

These pins are multiplexed on Data Pins D[7:5] respectively. Their states are latched at the time reset (\overline{RSTI}) is negated, and this information is used to determine the mode of operation for chip-select zero ($\overline{CS}[0]$) for initial operation.

Table 2-9. D[6] and D[5] Selection of \overline{CS}_0 Port Size

D[6]	D[5]	BOOT \overline{CS}_0 PORT SIZE
0	0	32-bit port
0	1	8-bit port
1	0	16-bit port
1	1	16-bit port

Table 2-10. D[7] Selection of \overline{CS}_0 Automatic Acknowledge

D[7]	BOOT \overline{CS}_0 AA
0	Disabled
1	Enabled with 15 wait states

Provided the required address range is first loaded into CSAR, you can program \overline{CS}_0 to continue to decode for a range of addresses after the V-bit is set. After the V-bit is set for \overline{CS}_0 , a system reset can restore global chip-select.

2.6.3 Byte Enables/Byte Write Enables - ($\overline{BE}[3:0]/\overline{BWE}[3:0]$)

The four byte-enable pins are multiplexed with the byte-write-enable pins of the MCF5307 device. You can program each of these four multiplexed pins through the Chip-Select Control Register on an individual chip-select basis.

2.6.4 Output Enable - (\overline{OE})

This signal is sent to the interfacing memory and/or peripheral to enable a read transfer. It is asserted and negated on the falling edge of the clock. This signal is asserted only when there is a match of one of the chip-select for the current address decode.

2.7 DRAM CONTROLLER SIGNALS

The following DRAM signals provide a seamless interface to external DRAM. DRAM widths of 8-, 16-, and 32- bits are supported and can access as much as 256 Mbytes of memory.

2.7.1 Row Address Strobes - ($\overline{RAS}[1:0]$)

These active-low pins provide a seamless interface to Row Address Strobe inputs on industry standard DRAMs. These pins provide RAS for a given DRAM bank. Banks correspond to specific Base Address and Control information programmed in the DCM registers (see DRAM section for a description). $\overline{RAS}[0]$ corresponds to bank 0 and $\overline{RAS}[1]$ corresponds to Bank 1.

2.7.2 Column Address Strobes - ($\overline{CAS}[3:0]$)

These active-low pins provide a seamless interface to Column Address Strobe inputs on industry standard DRAMs. These provide CAS for a given DRAM bank. $\overline{CAS}[3:0]$ controls access to the most significant to least significant byte of data, respectively.

2.7.3 DRAM Read/Write - (\overline{DRAMW})

This active-low pin is asserted to signify that a DRAM write cycle is underway.

2.7.4 Synchronous DRAM Column Address Strobe - (\overline{SCAS})

This active-low output signal is registered during synchronous mode to route directly to the SCAS signal of SDRAMs.

2.7.5 Synchronous DRAM Row Address Strobe - (\overline{SRAS})

This active-low output signal is registered during synchronous mode to route directly to the \overline{SRAS} signal of external SDRAMs.

2.7.6 Synchronous DRAM Clock Enable - (SCKE)

This active high output signal is registered during synchronous mode to route directly to the SCKE signal of external SDRAMs

2.7.7 Synchronous Edge Select - (EDGESEL)

This input signal helps select additional output hold time for signals that interface to external SDRAMs.

This signal can provide three modes of operation for the SDRAM interface signals. When this signal is tied high, the interface signals to the SDRAM will change on the rising edge of the clock (BCLKO). When this signal is tied low, the interface signals to the SDRAM will change on the falling edge of the clock (BCLKO). When this signal is tied to the SDRAM

clock, the interface signals to the SDRAM will change on the rising edge of the SDRAM clock. The various configurations of this signal can provide additional output hold time for the interface signals provided to the SDRAM.

2.8 DMA MODULE SIGNALS

The direct memory access (DMA) controller module uses the following signals to provide external request for either a source or destination.

2.8.1 DMA Request ($\overline{\text{DREQ}}[1:0]/\text{PP}[6:5]$)

These multiplexed pins can serve as the DMA request inputs, or as two bits of the parallel port. Programming the Pin Assignment Register (PAR) in the SCM determines the function of each of these three multiplexed pins. The pins are programmable on a bit-by-bit basis.

These active-low inputs are asserted by a peripheral device to request an operand transfer between that peripheral and memory.

2.9 SERIAL MODULE SIGNALS

The signals listed below transfer serial data between the DUART module and external peripherals.

2.9.1 Receive Data - ($\text{RD}[1:0]$)

These are the inputs on which serial data is received by the DUART. Data is sampled on $\text{RD}[1:0]$ on the rising edge of the serial clock source, with the least significant bit received first.

2.9.2 Transmit Data - ($\text{TD}[1:0]$)

The DUART transmits serial data on these output signals. Data is transmitted on the falling edge of the serial clock source, with the least significant bit transmitted (LSB) first. When no data is being transmitted or the transmitter is disabled, these two signals are held high. $\text{TD}[1:0]$ are also held high in local loopback mode.

2.9.3 Request To Send - ($\overline{\text{RTS}}[1:0]$)

The request-to-send outputs indicate to the peripheral device that the DUART is ready to send data and requires a clear-to-send signal to initiate transfer.

2.9.4 Clear To Send - ($\overline{\text{CTS}}[1:0]$)

Peripherals drive these inputs to indicate to the MCF5307 serial module that it can begin data transmission.

2.10 TIMER MODULE SIGNALS

The signals that follow are external interface to the two general-purpose MCF5307 timers. This 16-bit timer can capture timer values, trigger external events or internal interrupts, or count external events.

2.10.1 Timer Input - (TIN[1:0])

You can program these inputs as clocks that cause events in the counter and prescalars. They can also cause capture on the rising edge, falling edge, or both edges.

2.10.2 Timer Output - (TOUT1, TOUT0)

These programmable outputs pulse or toggle on various timer events.

2.11 PARALLEL 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], $\overline{TI}P$, \overline{DREQ} [1:0]. These 16 bits are programmed for functionality with the Pin Assignment Register (PAR) in the SIM.

The system designer controls the reset value of this register by driving the external data bus bit 4 (Data[4]) with a 1 or 0 when the \overline{RSTI} (reset input to MCF5307 device) negates. The logic level on Data[4] pin that is latched at negation of \overline{RSTI} is loaded into all bit positions of PP_RESET_SEL. The system is configured as PP[15:0] if Data[4] is 0; otherwise alternate pin functions selected by PAR=1 will be used. This value may be driven with a weak pullup or pulldown resistor.

2.12 M-BUS MODULE SIGNALS

The M-Bus module acts as a quick two-wire, bidirectional serial interface between the MCF5307 processor and peripherals with an M-Bus interface (e.g., LED controller, A-to-D converter, D-to-A converter). All devices connected to the M-Bus must have open-drain or open-collector outputs.

2.12.1 M-Bus Serial Clock (SCL)

This bidirectional, open-drain signal is the clock signal for M-Bus module operation. The M-Bus module controls this signal when the bus is in master mode; all M-Bus devices drive this signal to synchronize M-Bus timing.

2.12.2 M-Bus Serial Data (SDA)

This bidirectional, open-drain signal is the data input/output for the serial M-Bus interface.

2.13 DEBUG AND TEST SIGNALS

These signals interface with external I/O to provide processor status signals.

2.13.1 Test Mode - (MTMOD[3:0])

These signals choose between the BDM and JTAG signals that are multiplexed together. When MTMOD[3:0]=0000, the part is in BDM mode; when MTMOD[3:0]=0001, it is in JTAG mode. Note: All other combinations of MTMOD are reserved.

2.13.2 High Impedance - ($\overline{\text{HIZ}}$)

The assertion of $\overline{\text{HIZ}}$ will force all output drivers to a 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 $\overline{\text{TRST}}$.

2.13.3 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 will be 2x the CLKIN.

2.13.4 Debug Data - (DDATA[3:0])

The debug data pins are four bits wide, DDATA[3:0]. This nibble-wide bus displays captured processor data and break-point status. See **Section 16 Debug Support** for additional information on this bus.

2.13.5 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 2-11. Processor Status Signal Encodings

PST[3:0]	DEFINITION
0000	Continue execution
0001	Begin execution of an instruction
0010	Reserved
0011	Entry into user-mode
0100	Begin execution of PULSE instruction
0101	Begin execution of taken branch
0110	Reserved
0111	Begin execution of RTE instruction
1000	Reserved
1001	Reserved
1010	Reserved
1011	Reserved
1100	† Exception processing
1101	† Emulator-mode entry exception processing
1110	† Processor is stopped, waiting for interrupt
1111	† Processor is halted
† These encodings are asserted for multiple cycles.	

2.14 BDM/JTAG SIGNALS

The MCF5307 device complies with the IEEE 1149.1a JTAG testing standard. The JTAG test pins are multiplexed with background debug pins.

2.14.1 Test Clock - (TCK)

2

TCK is the dedicated JTAG test logic clock that is independent of the MCF5307 processor clock. Various JTAG operations occur on the rising or falling edge of TCK. The internal JTAG controller logic is designed such that holding TCK high or low for an indefinite period of time will not cause the JTAG test logic to lose state information. If TCK will not be used, it should be tied to ground.

2.14.2 Test Reset/Development Serial Clock - ($\overline{\text{TRST}}$ /DSCLK)

The MTMOD[3:0] signals determine the function of this dual-purpose pin. If MTMOD[3:0]=0000, the DSCLK function is selected. If MTMOD[3:0]= 0001, the TRST function is selected. MTMOD[3:0] should not be changed while $\overline{\text{RSTI}} = 1$. When used as $\overline{\text{TRST}}$, this pin will asynchronously reset the internal JTAG controller to the test logic reset state, causing the JTAG instruction register to choose the “bypass” command. When this occurs, all the JTAG logic is benign and will not interfere with the normal functionality of the MCF5307 processor. Although this signal is asynchronous, Motorola recommends that $\overline{\text{TRST}}$ make only a 0 to 1 (asserted to negated) transition while TMS is held at a logic 1 value. $\overline{\text{TRST}}$ has an internal pullup so that if it is not driven low its value will default to a logic level of 1. However, if $\overline{\text{TRST}}$ will not be used, it can either be tied to ground or, if TCK is clocked, it can be tied to VDD. The former connection will place the JTAG controller in the test logic reset state immediately, while the later connection will cause the JTAG controller (if TMS is a logic 1) to eventually end up in the test logic reset state after 5 clocks of TCK.

This pin is also used as the development serial clock (DSCLK) for the serial interface to the Debug Module. The maximum frequency for the DSCLK signal is 1/2 the BCLKO frequency. See **Section 16 Debug Support** for additional information on this signal.

2.14.3 Test Mode Select/ Break Point (TMS/ $\overline{\text{BKPT}}$)

The MTMOD[3:0] signals determine this pin's dual function. If MTMOD[3:0] =0000, the $\overline{\text{BKPT}}$ function is selected. If MTMOD[3:0] = 0001, then the TMS function is selected. MTMOD[3:0] should not change while $\overline{\text{RSTI}} = 1$. When used as TMS, this input signal provides the JTAG controller with information to determine which test operation mode should be performed. The value of TMS and current state 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 pullup so that if it is not driven low, its value will default to a logic level of 1. However, if TMS will not be used, it should be tied to VDD. This pin also signals a hardware breakpoint to the processor when in the debug mode. See **Section 16 Debug Support** for additional information on this signal.

2.14.4 Test Data Input/Development Serial Input - (TDI/DSI)

This is a dual-function pin. If MTMOD[3:0] = 0000, then DSI is selected. If MTMOD[3:0] = 0001, then TDI is selected. When used as TDI, this input signal provides the serial data port

for loading the various JTAG shift registers composed of the boundary scan register, the bypass register, and the instruction register. Shifting in of data depends on the state of the JTAG controller state machine and the instruction currently in the instruction register. This data shift occurs on the rising edge of TCK. TDI also has an internal pullup so that if it is not driven low its value will default to a logic level of 1. However, if TDI will not be used, it should be tied to VDD.

This pin also provides the single-bit communication for the debug module commands. See **Section 16 Debug Support** for additional information on this signal.

2

2.14.5 Test Data Output/Development Serial Output - (TDO/DSO)

This is a dual-function pin. When $MTMOD[3:0] = 0000$, then DSO is selected. When $MTMOD[3:0] = 0001$, TDO is selected. When used as TDO, this output signal provides the serial data port for outputting data from the JTAG logic. Shifting out of data depends on the state of the JTAG controller state machine and the instruction currently in the instruction register. This data shift occurs on the falling edge of TCK. When TDO is not outputting test data, it is three-stated. TDO can also be placed in three-state mode to allow bussed or parallel connections to other devices having JTAG. This signal also provides single-bit communication for the debug module responses. See **Section 16 Debug Support** for additional information on this signal.

SECTION 3 COLDFIRE CORE

3.1 ENHANCEMENTS

The MCF5307 is the first embedded microprocessor to utilize the performance-enhanced version 3 microprocessor core. This emphasis on the design of this processor core was on performance, backward compatibility, and as a step in the progression toward much higher performance embedded microprocessors. The version 3 core is an enhanced version of the version 2 core used in Motorola embedded processors such as the CF5206. Enhancements to the version 3 core include:

- 0.35-micron, triple-level-metal CMOS
- Clock-doubled core (90MHz and 66MHz)
- Bus speeds at 1/2, 1/3, or 1/4 the processor core speed
- Enhanced pipeline
- Enhanced instruction buffer
- Change of flow acceleration
- Illegal opcode handling
- Hardware MAC and Divide execution units added
- Debug module enhancements

3.1.1 Process

The MCF5307 is manufactured in a 0.35-micron CMOS process with triple-layer-metal routing technology. This process provides a combination of high performance and low power needed for embedded system applications. Inputs are 5V tolerant and outputs are CMOS or open-drain CMOS with outputs operating from VDD+0.5V to GND-0.5V, and with guaranteed TTL-level specifications.

3.1.2 Clock Doubled Microprocessor Core

The MCF5307 incorporates a clock-doubling phase-locked loop, as shown in Figure 3-1, that provides a clock source up to 90MHz for the processor core with a 45MHz input clock. The processor along with the unified cache and the integrated SRAM all operate at the higher speed clock. When combined with the enhanced pipeline structure of the Version 3 ColdFire, the processor and its local memories provide a high level of performance for today's demanding embedded applications.

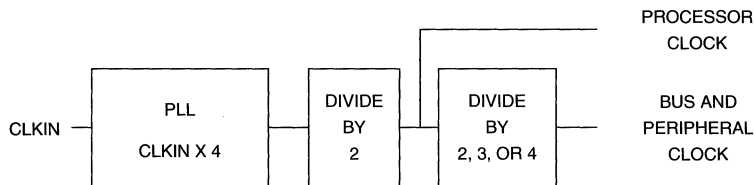


Figure 3-1. Clock Generation Block Diagram

3.1.3 Bus Clock at 1/2, 1/3, or 1/4 Processor Clock

For lower speed peripheral devices, the high-speed processor clock is running at a frequency much higher than required. For example, a UART running at 115K baud requires a much lower speed clock to keep up with the serial lines and to communicate with the microprocessor core. Running the UART at the high-speed processor clock requires higher power and a more complicated, high-speed bus interface. With the MCF5307, the bus clock can be selected at reset to be 1/2, 1/3, or 1/4 the processor clock (1, 2/3, or 1/2 CLOCKIN). This clock is the basis for bus cycles for all on-chip peripherals and the external bus interface.

3.1.4 Enhanced Pipeline

All ColdFire processor cores consist of two independent, decoupled pipeline structures to maximize performance while minimizing core size. The Instruction Fetch Pipeline (IFP) prefetches instructions, while the Operand Execution Pipeline (OEP) decodes the instruction, fetches the required operands and then executes the specified function. Because the IFP and OEP are decoupled by an instruction buffer that serves as a FIFO queue, the IFP can prefetch instructions in advance of their actual use by the OEP, thereby minimizing the time stalled waiting for the variable-length instructions. The OEP is implemented in a two-stage pipeline featuring a traditional RISC datapath with a dual-read-ported register file feeding an arithmetic/logic unit.

While one of the goals of the original ColdFire microarchitecture was to minimize overall size, the driving factor in the Version 3 design was to better balance the logic delays associated with each pipeline stage to allow the operating frequency to be raised significantly. For some functions, this required new pipeline stages to be added to support the high frequency goals. In particular, access on the processor's local high-speed bus were reimplemented to use a two-stage pipelined bus to the cache and SRAM memories. The net effect is that the Version 3 pipeline structure is considerably different from the Version 2 design. The V3 Instruction Fetch Pipeline is a four-stage design with an optional instruction buffer stage, while the Operand Execution Pipeline retains its two-stage structure. In the OEP design, each pipeline stage has multiple functions.

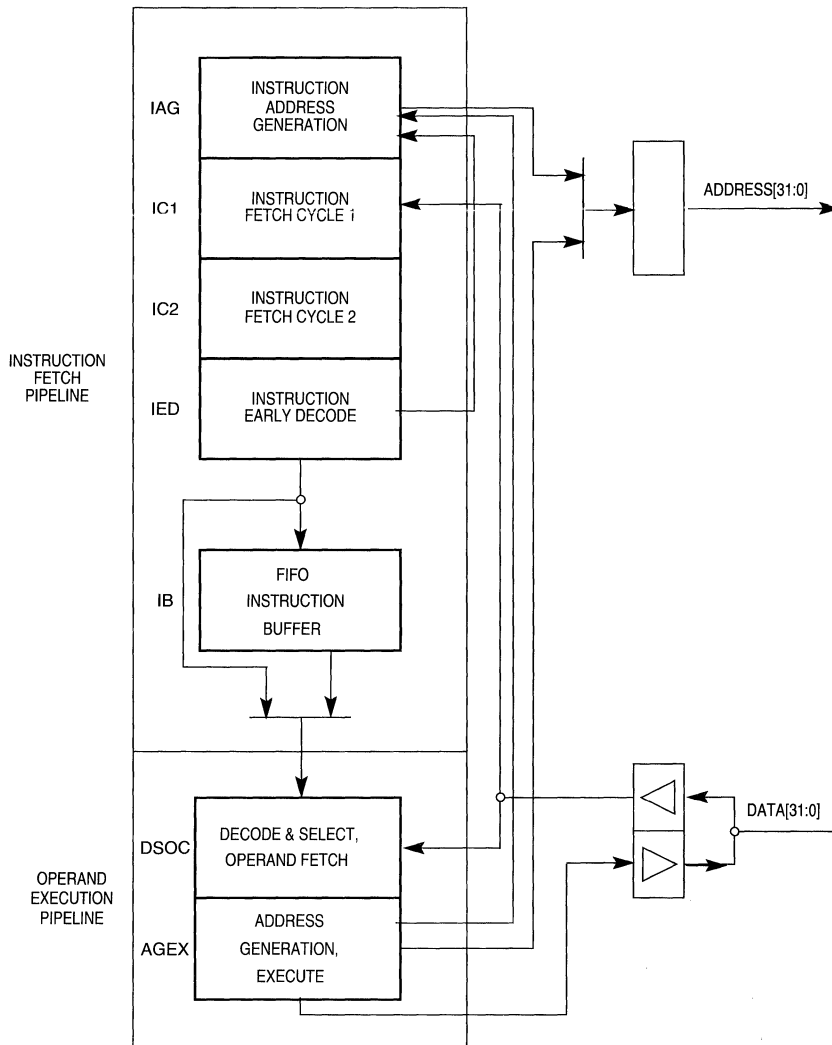


Figure 3-2. ColdFire Enhanced Pipeline

The V3 pipeline stages include the following:

- Instruction Fetch Pipeline
- Instruction Address Generation (IAG) - Calculation of the next prefetch address

- Instruction Fetch Cycle 1 (IC1) - Initiation of prefetch access on the processor's local bus
- Instruction Fetch Cycle 2 (IC2) - Completion of prefetch access on the processor's local bus.
- Instruction Early Decode (IED) - Generation of time-critical decode signals needed for the OEP.
- Instruction Buffer (IB) - Optional buffer stage using FIFO queue

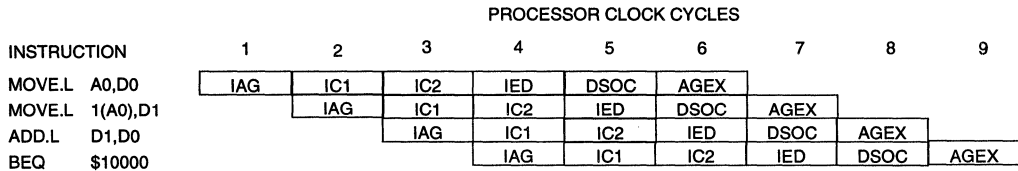


Figure 3-3. Pipelined Instruction Sequence

To more than double the processor clock rate, enhancements were required for the pipeline. Complex logical operations take a finite amount of time to execute. If there is not enough time left in the cycle, either the clock frequency must be reduced or pipelining added to divide up the logical work into synchronous stages, completing the task across several clock cycles instead of one. In doing this, the time required to complete an instruction, or instruction latency, is increased, but once the pipeline is filled, instructions are retired at a rate close to one per cycle for most instructions. The stages for the pipeline are divide into two functional blocks- the Instruction Fetch Pipeline and the Operand Execution Pipeline.

Operand Execution Pipeline:

- Decode and Select / Operand Fetch Cycle (DSOC) - Decode the instruction and fetch the required components for the effective address calculation, or the operand fetch cycle.
- Address Generation / Execute Cycle (AGEX) - Calculate the operand address, or perform the execution of the instruction.

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

It should be noted that the organization of the Version 3 instruction buffer is fundamentally different than the Version 2 approach. One of the time-critical decode fields provided by the "early decode" stage of the IFP is the instruction length. By knowing the length of the prefetched instructions, the IED field is able to package the fetched data into machine instructions and load them into the FIFO instruction buffer in that form. This approach greatly

accelerates the OEP read logic. As one instruction is completed in the OEP, the next instruction, regardless of instruction length, is read from the next sequential buffer location and loaded into the instruction registers. The Version 3 instruction buffer provides storage for eight machine instructions.

The resulting pipeline and local bus structure allows the Version 3 ColdFire processor core to deliver sustained high performance across a variety of demanding embedded applications. When operating in a 90/45MHz configuration, the MCF5307 is capable of 70 Dhrystone MIPS of performance.

3.1.5 Change Of Flow Acceleration

Because the Version 3 Instruction Fetch and Operand Execution Pipelines are decoupled by the instruction buffer, the increased depth of the IFP is generally hidden from the OEP instruction execution rate. The one exception is change-of-flow instructions, e.g., unconditional branches or jumps, subroutine calls, taken conditional branches, etc. For these instructions, the increased depth of the IFP pipeline is fully exposed. To minimize the effects of this increased depth, a logic module dedicated to change-of-flow acceleration was developed for the IED stage of the Instruction Fetch Pipeline.

The basic premise of the Version 3 branch acceleration is to detect certain types of change-of-flow instructions, calculate their target instruction address, and immediately begin fetching down the target stream. By allowing the switching of the prefetch stream to be handled completely within the IFP without any Operand Execution Pipeline intervention, the typical execution time is greatly improved.

As an example, consider a PC-relative unconditional branch using the BRA instruction. The branch acceleration logic searches the prefetch stream for this type of opcode. Once encountered, the acceleration logic calculates the target address by summing the current instruction prefetch address with a displacement contained in the instruction. This detection and calculation of the target address occurs in the IED stage of the BRA prefetch. The target address is then immediately fed back into the IAG stage, causing the current prefetch stream to be discarded and a new stream at the target address established. Given that the two pipelines are decoupled, in many cases, the target instruction is available to the OEP immediately after the BRA instruction, making its execution time appear as a single cycle.

The acceleration logic uses a static prediction algorithm when processing conditional branch (Bcc) instructions. The default prediction scheme is forward Bcc instructions are predicted as not-taken, while backward Bcc opcodes are predicted as taken. A user-mode control bit (bit 7 of the CCR) is provided to allow users to dynamically alter the prediction algorithm for forward Bcc instructions. See subsection 3.2.1.5 on the CCR for details.

Depending on the runtime characteristics of an application, processor performance may be increased significantly by the assertion or negation of this configuration bit. See subsection 3.13 on Branch Instruction Execution Times for details on individual instruction performance.

3.1.6 Illegal Opcode Handling

As an aid in conversion from M68000 Family code, all instruction bits are parsed to insure all instructions are valid. If the processor attempts execution of an illegal or nonsupported instruction, an illegal instruction exception is taken.

3.1.7 Hardware Multiply / Accumulate and Divide Execution Units

The MAC unit provides a limited set of DSP operations that are currently being used in embedded code today while also supporting the integer multiply instructions in the ColdFire microprocessor family.

The MAC unit provides functionality in three related areas:

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

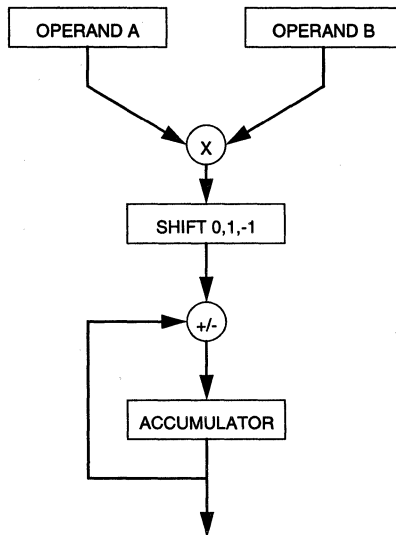


Figure 3-4. ColdFire Multiply-Accumulate Block Diagram

The ColdFire MAC has been optimized for 16x16 multiplies to keep silicon costs down. The MAC unit is tightly coupled to the processor's Operand Execution Pipeline and features a three-stage execution pipeline. The OEP can issue a 16 x 16 multiply with a 32-bit accumulate operation in a single cycle, while a 32 x 32 multiply with a 32-bit accumulation requires three cycles before the next instruction can be issued. Figure 3-4 shows the basic functionality diagram of the ColdFire MAC.

The MCF5307 processor also includes a hardware divider which performs a number of integer divide operations. The supported divide functions include: 32/16 producing a 16-bit quotient and a 16-bit remainder, 32/32 producing a 32-bit quotient, and 32/32 producing a 32-bit remainder.

3.1.8 Debug Module Enhancements

The MCF5307 is the first processor to use the enhanced Revision B debug module definition. Enhancements include the following:

- Serial BDM command to display current program counter without halting the CPU
- Added capability to logically OR hardware breakpoint triggers
- Added registers to support concurrent BDM commands and active breakpoints
- An external mechanism to generate a debug interrupt
- A program-visible register field to identify the debug module revision

The enhanced functionality of the Revision B debug specification is fully backward compatible with the original definition.

3.2 PROGRAMMING MODEL

The MCF5307 microprocessor core programming model consists of three instruction and register groups: user, user-mode MAC, and supervisor. Programs executing in user mode are restricted to the basic user and MAC instructions and programming models. System software executing in supervisor mode can reference all user-mode and MAC instructions and registers plus an additional set of privileged instructions and control registers. The appropriate programming model is selected based on the privilege level (user or supervisor) of the processor as defined by the S-bit of the status register. The following paragraphs describe the registers in the user, MAC and supervisor programming models.

3.2.1 User Programming Model

Figure 3-5 illustrates the user programming model. The model is the same as for M68000 Family microprocessors, consisting of the following registers:

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

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

3.2.1.2 ADDRESS REGISTERS (A0–A6). These registers can be used as software stack pointers, index registers, or base address registers as well as for word and longword operations.

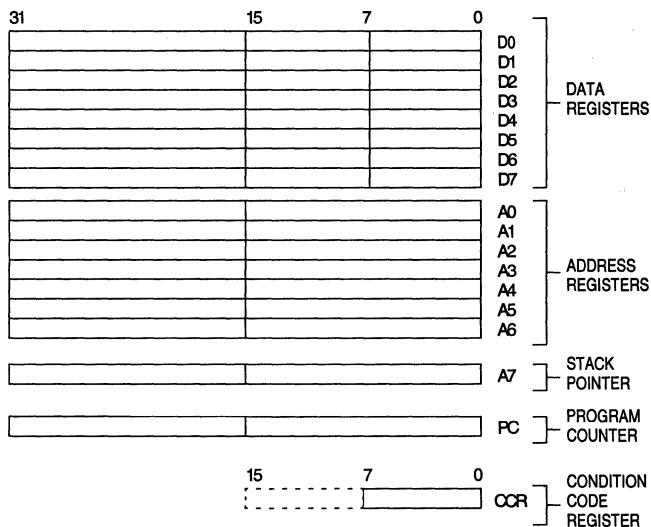


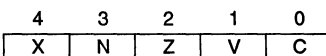
Figure 3-5. User Programming Model

3.2.1.3 STACK POINTER (A7). The ColdFire architecture supports a single hardware stack pointer (A7) for explicit references or implicit ones during stacking for subroutine calls and returns and exception handling. The initial value of A7 is loaded from the reset exception vector, address \$0. The same register is used for both user and supervisor mode as well as word and longword operations.

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

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

3.2.1.5 CONDITION CODE REGISTER . The CCR is the least significant byte of the processor status register (SR). Bits 4–0 represent indicator flags based on results generated by processor operations. Bit 4, the extend bit (X-bit), is also used as an input operand during multiprecision arithmetic computations.



X— extend condition code bit

N— negative condition code bit

Set if the most significant bit of the result is set; otherwise cleared

Z— zero condition code bit

Set if the result equals zero; otherwise cleared

V— overflow condition code bit

Set if an arithmetic overflow occurs implying that the result cannot be represented in the operand size; otherwise cleared

C— carry condition code bit

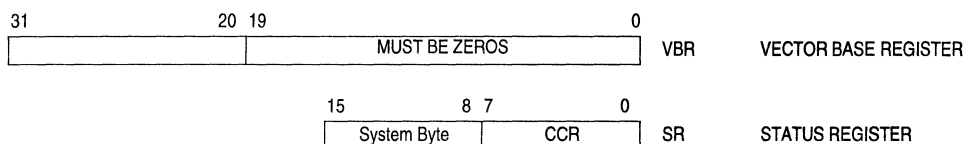
Set if a carryout of the operand MSB occurs for an addition, or if a borrow occurs in a subtraction; otherwise cleared

Set to the value of the C-bit for arithmetic operations; otherwise not affected.

3.2.2 Supervisor Programming Model

Only system programmers use the supervisor programming model to implement sensitive operating system functions, I/O control, and memory management. All accesses that affect the control features of ColdFire 5200 processors are in the supervisor programming model, which consists of the registers available to users as well as the following control registers:

- 16-bit status register (SR)
- 32-bit vector base register (VBR)



Supervisor Programming Model

Additional registers may be supported on a part-by-part basis.

The following paragraphs describe the supervisor programming model registers.

3.2.2.1 STATUS REGISTER. The SR stores the processor status and includes the CCR, the interrupt priority mask, and other control bits. In the supervisor mode, software can access the entire SR. In user mode, only the lower 8 bits are accessible (CCR). The control bits indicate the following states for the processor: trace mode (T-bit), supervisor or user mode (S-bit), and master or interrupt state (M).

Special Note: After a reset exception, the contents of the status register are \$27xx.

SYSTEM BYTE						CONDITION CODE REGISTER (CCR)									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T	0	S	M	0	I[2:0]		0	0	0	X	N	Z	V	C	

Status Register

T– trace enable

When set, the processor will perform a trace exception after every instruction.

S– supervisor / user state

Denotes whether the processor is in supervisor mode (S=1) or user mode (S=0).

M– master / interrupt state

This bit is cleared by an interrupt exception, and can be set by software during execution of the RTE or move to SR instructions.

I[2:0]– 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.

3.2.2.2 VECTOR BASE REGISTER (VBR). The VBR contains 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. The lower 20 bits of the VBR are not implemented by ColdFire 5200 processors; they are assumed to be zero, forcing the table to be aligned on a 1 Mbyte boundary.

3.2.3 MAC Programming Model

Figure 3-6 illustrates the MAC portion of the user programming model available on the MCF5307 microprocessor core. It consists of the following registers:

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

The instructions which reference the MAC registers always transfer 32 bits of data, regardless of the implemented size of the register.

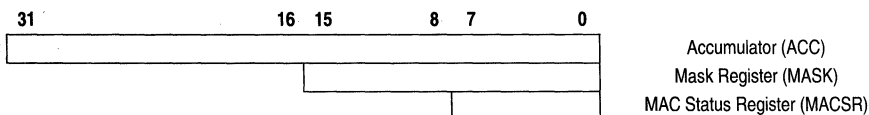


Figure 3-6. MAC Unit User Programming Model

3.2.3.1 ACCUMULATOR (ACC). This is a 32-bit general-purpose register used to accumulate the results of MAC operations.

3.2.3.2 MASK REGISTER (MASK). This is a 16-bit general-purpose register for use as an optional address mask during MAC instructions which fetch operands from memory. It is useful in the implementation of circular queues in operand memory.

3.2.3.3 MAC STATUS REGISTER (MACSR). This is an 8-bit special-purpose register which defines the operating configuration of the MAC unit, and contains indicator flags from the results of MAC instructions.

The ColdFire processors provide a simplified exception processing model. The next subsection details the model.

3.3 EXCEPTION PROCESSING OVERVIEW

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

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

ColdFire processors use an instruction restart exception model but do require more software support to recover from certain access errors. See subsection **3.5.1 Access Error Exception** for details.

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

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

Second, the processor determines the exception vector number. For all faults *except* interrupts, the processor performs this calculation based on the exception type. For interrupts, the processor performs an interrupt-acknowledge (IACK) bus cycle to obtain the vector number from a peripheral device. The IACK cycle is mapped to a special acknowledge address space with the interrupt level encoded in the address.

Third, the processor saves the current context by creating an exception stack frame on the system stack. ColdFire processors support a single stack pointer in the A7 address register; therefore, there is no notion of separate supervisor or 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 placed in the

exception stack frame defines the location of the faulting instruction (fault) or the address of the next instruction to be executed (next).

Fourth, the processor acquires the address of the first instruction of the exception handler. By definition, the exception vector table is aligned on a 1 MByte boundary. This instruction address is obtained by fetching a value from the table located at the address defined in the vector base register. The index into the exception table is calculated as $(4 \times \text{vector_number})$. Once the index value has been generated, the contents of the vector table determine the address of the first instruction of the desired handler. After the instruction fetch for the first opcode of the handler has been initiated, exception processing terminates and normal instruction processing continues in the handler.

3

ColdFire processors support a 1024-byte vector table aligned on any 1 MByte address boundary (see Table 3-1). The table contains 256 exception vectors where the first 64 are defined by Motorola and the remaining 192 are user-defined interrupt vectors.

Table 3-1. Exception Vector Assignments

VECTOR NUMBER(S)	VECTOR OFFSET (HEX)	STACKED PROGRAM COUNTER	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
5-7	\$014-\$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
48-63	\$0C0-\$0FC	-	Reserved
64-255	\$100-\$3FC	Next	User-defined interrupts

"Fault" refers to the PC of the instruction that caused the exception

"Next" refers to the PC of the next instruction that follows the instruction that caused the fault.

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.

3.4 EXCEPTION STACK FRAME DEFINITION

The exception stack frame is shown in Figure 3-7. 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.

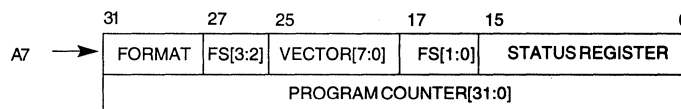


Figure 3-7. Exception Stack Frame Form

The 16-bit format/vector word contains 3 unique fields:

- A 4-bit format field at the top of the system stack is always written with a value of {4,5,6,7} by the processor indicating a two-longword frame format. See Table 3-2. This

field records any longword misalignment of the stack pointer which might have existed at the time the exception occurred.

Table 3-2. Format Field Encoding

ORIGINAL A7 @ TIME OF EXCEPTION, BITS 1:0	A7 @ 1ST INSTRUCTION OF HANDLER	FORMAT FIELD Bits 31:28
00	Original A7 - 8	0100
01	Original A7 - 9	0101
10	Original A7 - 10	0110
11	Original A7 - 11	0111

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

Table 3-3. Fault Status Encodings

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

- The 8-bit vector number, vector[7:0], defines the exception type and is calculated by the processor for all internal faults and represents the value supplied by the peripheral in the case of an interrupt. Refer to Table 3-1.

3.5 PROCESSOR EXCEPTIONS

3.5.1 Access Error Exception

For the MCF5307 processor, access errors are only reported in conjunction with an attempted store to a write-protected memory space. Thus, access errors associated with instruction fetch or operand read accesses are not possible.

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

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

The ColdFire processor uses an imprecise reporting mechanism for access errors on operand writes. Because the actual write cycle may be decoupled from the processor's issuing of the operation, the signaling of an access error appears to be decoupled from the instruction that generated the write. Accordingly, the PC contained in the exception stack frame merely represents the location in the program when the access error was signaled. All programming model updates associated with the write instruction are completed. The NOP instruction can collect access errors for writes. This instruction delays its execution until all previous operations, including all pending write operations, are complete. If any previous write terminates with an access error, it is guaranteed to be reported on the NOP instruction.

3.5.2 Address Error Exception

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

Any attempted use of a word-sized index register (Xi.w) or a scale factor of 8 on an indexed effective addressing mode generates an address error as does an attempted execution of an instruction with a full-format indexed addressing mode.

3.5.3 Illegal Instruction Exception

On the Version 2 ColdFire microprocessor implementation, only certain illegal opcodes were decoded and generated an illegal instruction exception. However, the Version 3 processor decodes the full 16-bit opcode and generates an illegal instruction exception if the execution of any non-supported instruction is attempted. Additionally, if execution of any illegal line A or line F opcode is attempted, unique exception types are generated: vector numbers 10 and 11, respectively.

ColdFire processors do not provide illegal instruction detection on the extension words on any instruction, including MOVEC. If execution of instruction with an illegal extension word is attempted, the resulting operation is undefined.

3.5.4 Privilege Violation

The attempted execution of a supervisor mode instruction while in user mode generates a privilege violation exception. See the *ColdFire Family Programmer's Reference Manual Rev 1.0* for lists of supervisor- and user-mode instructions.

3.5.5 Trace Exception

To aid in program development, the ColdFire processors provide an instruction-by-instruction tracing capability. While in trace mode, indicated by the assertion of the T-bit in the status register ($SR[15] = 1$), the completion of an instruction execution signals a trace exception. This functionality allows a debugger to monitor program execution.

The single exception to this definition is the STOP instruction. When the STOP opcode is executed, the processor core waits until an unmasked interrupt request is asserted, then aborts the pipeline and initiates interrupt exception processing.

Because ColdFire processors do not support any hardware stacking of multiple exceptions, it is the responsibility of the operating system to check for trace mode after processing other exception types. As an example, consider the execution of a TRAP instruction while in trace mode. The processor initiates the TRAP exception and then pass control to the corresponding handler. If the system requires that a trace exception be processed, it is the responsibility of the TRAP exception handler to check for this condition ($SR[15]$ in the exception stack frame asserted) and pass control to the trace handler before returning from the original exception.

3.5.6 Debug Interrupt

This special type of program interrupt is discussed in detail in **Section 16 Debug Support**. This exception is generated in response to a hardware breakpoint register trigger. The processor does not generate an IACK cycle but rather calculates the vector number internally (vector number 12). Additionally, the M-bit and the interrupt priority mask fields of the status register are unaffected by the occurrence of a debug interrupt.

3.5.7 RTE and Format Error Exceptions

When an RTE instruction is executed, 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 the RTE instruction.

The selection of the format value provides some limited debug support for porting code from 68000 applications. On M68000 Family processors, the SR was located at the top of the stack. On those processors, bit[30] of the longword addressed by the system stack pointer is typically zero. Thus, if an RTE is attempted using this "old" format, it generates a format error on a ColdFire processor.

If the format field defines a valid type, the processor: (1) reloads the SR operand, (2) fetches the second longword operand, (3) adjusts the stack pointer by adding the format value to

the auto-incremented address after the fetch of the first longword, and then (4) transfers control to the instruction address defined by the second longword operand within the stack frame.

3.5.8 TRAP Instruction Exceptions

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

3.5.9 Interrupt Exception

The 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 supported through the System Integration module (SIM).

3.5.10 Fault-on-Fault Halt

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

3.5.11 Reset Exception

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

The reset exception places the processor in the supervisor mode by setting the S-bit and disables tracing by clearing the T-bit in the SR. This exception also clears the M-bit and sets the processor’s interrupt priority mask in the SR to the highest level (level 7). Next, the VBR is initialized to zero (\$00000000). The control registers specifying the operation of any memories (e.g., cache and/or RAM modules) connected directly to the processor are disabled.

Note

Other implementation-specific supervisor registers are also affected. Refer to each of the modules in this user’s manual for details on these registers.

Once the processor is granted the bus and it does not detect any other alternate masters taking the bus, the core then performs two longword read bus cycles. The first longword at address 0 is loaded into the stack pointer and the second longword at address 4 is loaded into the program counter. After the initial instruction is fetched from memory, program execution begins at the address in the PC. If an access error or address error occurs before the first instruction is executed, the processor enters the fault-on-fault halted state.

3.6 INTEGER DATA FORMATS

Table 3-4 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 3-4. Integer Data Formats

OPERAND DATA FORMAT	SIZE
Bit	1 Bit
Byte Integer	8 Bits
Word Integer	16 Bits
longword Integer	32 Bits

3

3.7 ORGANIZATION OF DATA IN REGISTERS

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

3.7.1 Organization of Integer Data Formats in Registers

Figure 3-8 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.

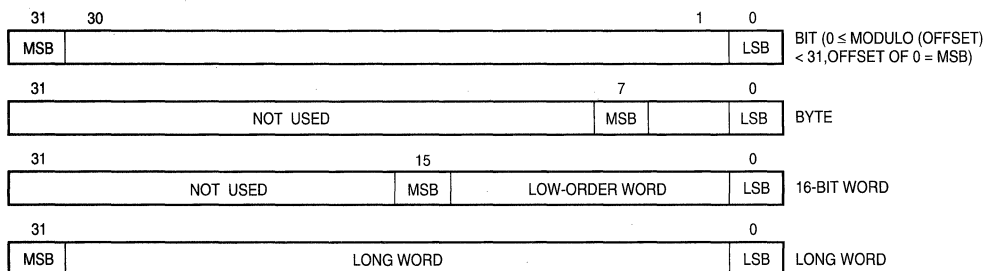


Figure 3-8. Organization of Integer Data Formats in Data Registers

Because address registers and stack pointers are 32-bits wide, address registers cannot be used for byte-size 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. When an address register is used, the entire register is affected, regardless of the operation size. If the source operand is a word size, it is sign-extended to 32 bits and then used in the operation to an address register destination. Address registers are primarily for addresses

and address computation support. Figure 3-9 illustrates the integer format for address registers.

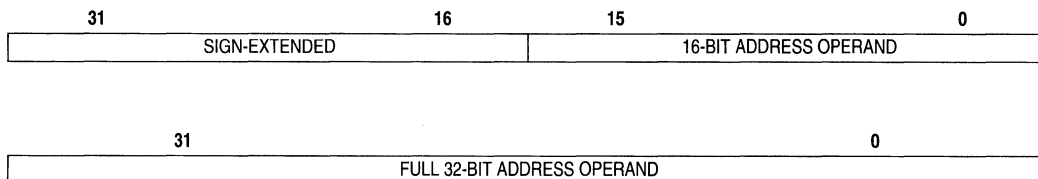


Figure 3-9. Organization of Integer Data Formats in Address Registers

3

Control registers vary in size according to function. Some control registers 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, despite privilege mode.

3.7.2 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 3-10.

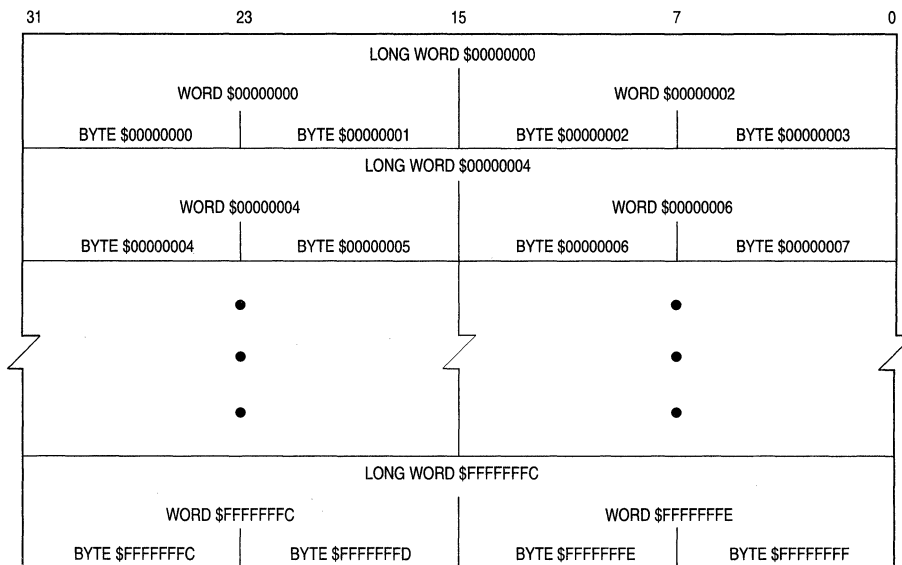


Figure 3-10. Memory Operand Addressing

3.8 ADDRESSING MODE SUMMARY

The addressing modes are grouped into categories according to the mode of use. Data addressing modes refer to data operands. Memory addressing modes refer to memory operands. Alterable addressing modes refer to alterable (writable) operands. Control addressing modes refer to memory operands without an associated size.

These categories sometimes combine to form new categories that are more restrictive. Two combined classifications are alterable memory (both alterable and memory) and data alterable (both alterable and data). Table 3-5 lists a summary of effective addressing modes

and their categories. Twelve of the most commonly used addressing modes from the MM68000 Family are available on ColdFire microprocessors.

Table 3-5. Effective Addressing Modes and Categories

ADDRESSING MODES	SYNTAX	MODE FIELD	REG. FIELD	CATEGORY			
				DATA	MEMORY	CONTROL	ALTERABLE
Register Direct	Dn	000	reg. no.	X	—	—	X
Data Address	An	001	reg. no.	—	—	—	X
Register Indirect	(An)	010	reg. no.	X	X	X	X
Address	(An)+	011	reg. no.	X	X	—	X
Address with Postincrement	—(An)	100	reg. no.	X	X	—	X
Address with Predecrement	(d ₁₆ , An)	101	reg. no.	X	X	X	X
Address with Displacement							
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	(xxx).W	111	000	X	X	X	—
Long	(xxx).L	111	001	X	X	X	—
Immediate	#<xxx>	111	100	X	X	—	—

3.9 INSTRUCTION SET SUMMARY

Table 3-6 lists the notational conventions used throughout this manual unless otherwise specified. Table 3-7 lists the 5307 microprocessor core instruction set by opcode. This instruction set is a reduced version of the 68000 instruction set. The removed instructions include BCD, bit field, logical rotate, decrement and branch, integer division, and integer multiply with a 64-bit result. In addition, nine new MAC instructions have been added.

Table 3-6. Notational Conventions

OPCODE WILDCARDS	
cc	Logical Condition (example: NE for not equal)
REGISTER OPERANDS	
An	Any Address Register n (example: A3 is address register 3)
Ax,Ay	Source and destination address registers, respectively
Dn	Any Data Register n (example: D5 is data register 5)
Dx,Dy	Source and destination data registers, respectively
Rn	Any Address or Data Register
Rx,Ry	Any source and destination registers, respectively
Rw	Any second source register
Rc	Any Control Register (example VBR is the vector base register)

Table 3-6. Notational Conventions (Continued)

REGISTER/PORT NAMES	
ACC	MAC Accumulator
DDATA	Debug Data Port
CCR	Condition Code Register (lower byte of status register)
MACSR	MAC Status Register
MASK	Mask Register
PC	Program Counter
PST	Processor Status Port
SR	Status Register
MISCELLANEOUS OPERANDS	
#<data>	Immediate data following the instruction word(s).
<ea>	Effective Address
<label>	Assembly Program Label
<list>	List of registers (example: D3–D0)
<size>	Operand data size: Byte (B), Word (W), Longword (L)
OPERATIONS	
+	Arithmetic addition or postincrement indicator
-	Arithmetic subtraction or predecrement indicator
X	Arithmetic multiplication
~	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: MSB of D0)
LSW	Least Significant Word
MSB	Most Significant Bit
MSW	Most Significant Word
CONDITION CODE REGISTER BIT NAMES	
P	Branch Prediction Bit in CCR
C	Carry Bit in CCR
N	Negative Bit in CCR
V	Overflow Bit in CCR
X	Extend Bit in CCR
Z	Zero Bit in CCR

Table 3-6. Notational Conventions (Continued)

P	Branch Prediction Bit in CCR
C	Carry Bit in CCR
N	Negative Bit in CCR
V	Overflow Bit in CCR
X	Extend Bit in CCR
Z	Zero Bit in CCR

Table 3-7. Instruction Set Summary

INSTRUCTION	OPERAND SYNTAX	OPERAND SIZE	OPERATION
ADD	Dy,<ea>x <ea>y,Dx	32 32	Source + Destination → Destination
ADDA	<ea>y,Ax	32	Source + Destination → Destination
ADDI	#<data>,Dx	32	Immediate Data + Destination → Destination
ADDQ	#<data>,<ea>x	32	Immediate Data + Destination → Destination
ADDX	Dy,Dx	32	Source + Destination + X → Destination
AND	Dy,<ea>x <ea>y,Dx	32 32	Source & Destination → Destination
ANDI	#<data>,Dx	32	Immediate Data & Destination → Destination
ASL	Dx,Dy #<data>,Dx	32 32	X/C ← (Dy << Dx) ← 0 X/C ← (Dy << #<data>) ← 0
ASR	Dx,Dy <data>,Dx	32 32	MSB → (Dy >> Dx) → X/C MSB → (Dy >> #<data>) → X/C
Bcc	<label>	8,16	If Condition True, Then PC + d _n → PC
BCHG	Dy,<ea>x #<data>,<ea>x	8,32 8,32	~(<Bit Number> of Destination) → Z, Bit of Destination
BCLR	Dy,<ea>x #<data>,<ea>x	8,32 8,32	~(<Bit Number> of Destination) → Z; 0 → Bit of Destination
BRA	<label>	8,16	PC + d _n → PC
BSET	Dy,<ea>x #<data>,<ea>x	8,32 8,32	~(<Bit Number> of Destination) → Z; 1 → Bit of Destination
BSR	<label>	8,16	SP - 4 → SP; PC → (SP); PC + d _n → PC
BTST	Dy,<ea>x #<data>,<ea>x	8,32 8,32	~(<Bit Number> of Destination) → Z
CLR	<ea>x	8,16,32	0 → Destination
CMPI	#<data>,Dx	32	Destination - Immediate Data
CMP	<ea>y,Dx	32	Destination - Source
CMPA	<ea>y,Ax	32	Destination - Source
CPUSH	(An)	32	Push and Invalidate Cache Line
DIVS	<ea>y,Dx	16 32	<ea>y → Dx {16-bit Remainder; 16-bit Quotient} <ea>y → Dx {32-bit Quotient} Note: signed operation
DIVU	<ea>y,Dx	16	<ea>y → Dx {16-bit Remainder; 16-bit Quotient} <ea>y → Dx {32-bit Quotient} Note: unsigned operation
EOR	Dy,<ea>x	32	Source ~ Destination → Destination
EORI	#<data>,Dx	32	Immediate Data ~ Destination → Destination
EXT	Dx Dx	8 → 16 16 → 32	Sign-Extended Destination → Destination
EXTB	Dx	8 → 32	Sign-Extended Destination → Destination
HALT	none	none	Enter Halted State
JMP	<ea>	none	Address of <ea> → PC

Table 3-7. Instruction Set Summary (Continued)

INSTRUCTION	OPERAND SYNTAX	OPERAND SIZE	OPERATION
JSR	<ea>	none	SP-4 → SP; PC → (SP); <ea> → PC
LEA	<ea>y, Ax	32	<ea> → An
LINK	Ax, #<data>	16	SP-4 → SP; Ax → (SP); SP → Ax; SP + d16 → SP
LSL	Dx, Dy #<data>, Dx	32 32	X/C ← (Dy << Dx) ← 0 X/C ← (Dx << #<data>) ← 0
LSR	Dx, Dy #<data>, Dx	32 32	0 → (Dy >> Dx) → X/C 0 → (Dx >> #<data>) → X/C
MAC	Rw, Rx, <shift>	16 × 16 + 32 → 32 32 × 32 + 32 → 32	ACC + (Rw × Rx){ << 1 >> 1 } → ACC
MACL	Rw, Rx, <shift>, <ea>, Ry	16 × 16 + 32 → 32 32 × 32 + 32 → 32	ACC + (Rw × Rx){ << 1 >> 1 } → ACC; (<ea> & MASK) → Ry
MOVE	<ea1>, <ea2>	8, 16, 32	<ea1> → <ea2>
MOVE from ACC	ACC, Rx	32	ACC → Rx
MOVE from CCR	Dx	16	CCR → Dx
MOVE from MACSR	MACSR, Rx MACSR, CCR	32 8	MACSR → Rx MACSR → CCR
MOVE from MASK	MASK, Rx	32	MASK → Rx
MOVE from SR	Dx	16	SR → Destination
MOVE to ACC	Rx, ACC<#<data>, ACC	32 32	Rx → ACC #<data> → ACC
MOVE to CCR	Dn, CCR, #<data>, CCR	16	Dn → CCR #<data> → CCR
MOVE to MACSR	Rn, MACSR #<data>, MACSR	32	Rn → MACSR #<data> → MACSR
MOVE to MASK	Rn, MASK #<data>, MASK	32 32	Rn → MASK #<data> → MASK
MOVE to SR	Dy, SR #<data>, SR	16	Source → SR
MOVEA	<ea>y, Ax	16, 32 → 32	Source → Destination
MOVEC	Rn, Rc	32	Rn → Rc
MOVEM	list, <ea>x <ea>y, list	32 32	Listed Registers → Destination Source → Listed Registers
MOVEQ	#<data>, Dx	8 → 32	Sign-extended Immediate Data → Destination
MSAC	Rw, Rx, <shift>	32 - 16 × 16 → 32 32 - 32 × 32 → 32	ACC - (Rw × Rx){ << 1 >> 1 } SF → ACC
MSACL	Rw, Rx, <shift>, <ea>, Ry	32 - 16 × 16 → 32 32 - 32 × 32 → 32	ACC - (Rw × Rx){ << 1 >> 1 } SF → ACC; (<ea> & MASK) → Ry
MULS	<ea>y, Dx	16 × 16 → 32 32 × 32 → 32	Source × Destination → Destination Signed or unsigned
MULU	<ea>y, Dl	16 × 16 → 32 32 × 32 → 32	Source × Destination → Destination Signed or unsigned
NEG	<ea>x	32	0 - Destination → Destination
NEGX	<ea>x	32	0 - Destination - X → Destination
NOP	none	none	PC + 2 → PC; Sync Pipeline
NOT	<ea>	32	~ Destination → Destination
OR	Dy, <ea>x <ea>y, Dx	32	Source Destination → Destination
ORI	#<data>, Dx	32	Immediate Data Destination → Destination
PEA	<ea>	32	SP-4 → SP; Address of <ea> → (SP)
PULSE	none	none	Set PST = \$4
REMS	<ea>y, Dx	32	<ea>y → Dx {32-bit Remainder} Note: Signed operation
REMU	<ea>y, Dx	32	<ea>y → Dx {32-bit Remainder} Note: Unsigned operation

Table 3-7. Instruction Set Summary (Continued)

INSTRUCTION	OPERAND SYNTAX	OPERAND SIZE	OPERATION
RTE	none	none	(SP+2) →SR; (SP+4) →PC; SP+ 8 → PC
RTS	none	none	(SP) →PC; SP + 4 → SP
Scc	Dx	8	If Condition True, Then 1's → Destination; Else 0's → Destination
STOP	#<data>	16	Immediate Data →SR; Enter Stopped State
SUB	Dy,<ea>x <ea>y,Dx	32 32	Destination - Source → Destination
SUBA	<ea>,Ax	32	Destination - Source → Destination
SUBI	#<data>,Dx	32	Destination - Immediate Data → Destination
SUBQ	#<data>,<ea>x	32	Destination - Immediate data → Destination
SUBX	Dy,Dx	32	Destination - Source - X → Destination
SWAP	Dn	16	MSW of Dn ↔ LSW of Dn
TRAP	none	none	SP - 4 → SP; PC → (SP); SP - 2 → SP; SR → (SP); SP - 2 → SP; Format → (SP); Vector Address → PC
TRAPF	none #<data>	none 16 32	PC + 2 → PC; PC + 4 → PC; PC + 6 → PC
TST	<ea>y	8,16,32	Set Integer Condition Codes
UNLK	Ax	32	Ax →SP; (SP) → Ax; SP + 4 → SP
WDDATA	<ea>	8,16,32	<ea> →DDATA port
WDEBUG	<ea>	16	<ea> → Debug Module

3.9.1 Timing Assumptions

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

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

If the operand alignment fails these guidelines, it is misaligned. The processor core decomposes the misaligned operand reference into a series of aligned accesses as shown in Table 3-8.

Table 3-8. Misaligned Operand References

ADDRESS[1:0]	SIZE	BUS OPERATIONS	ADDITIONAL C(R/W)
X1	Word	Byte, Byte	2(1/0) if read 1(0/1) if write
X1	Long	Byte, Word, Byte	3(2/0) if read 2(0/2) if write
10	Long	Word, Word	2(1/0) if read 1(0/1) if write

3

3.9.2 MOVE Instruction Execution Times

The execution times for the MOVE.{B,W} instructions are shown in Table 3-9, while Table 3-10 provides the timing for MOVE.L.

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

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

Table 3-9. 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 3-10. 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)	—	—

Table 3-10. MOVE Long Execution Times (Continued)

SOURCE	DESTINATION						
	Rx	(Ax)	(Ax)+	-(Ax)	(d16,Ax)	(d8,Ax,Xi*SF)	xxx.wl
(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)	—	—	—

The following table specifies the execution times for the Move Long instructions accessing the program-visible registers of the MAC unit.

Table 3-11. MAC MOVE Long Instruction Execution Times

OPCODE	<ea>	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	1(0/0)	—	—	—	—	—	—	1(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)	—	—	—	—	—	—	—

3.10 STANDARD ONE OPERAND INSTRUCTION EXECUTION TIMES

Table 3-12. One Operand Instruction Execution Times

OPCODE	<EA>	EFFECTIVE ADDRESS							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xi*SF)	xxx.wl	#xxx
clr.b	<ea>	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
clr.w	<ea>	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
clr.l	<ea>	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
ext.w	Dx	1(0/0)	—	—	—	—	—	—	—
ext.l	Dx	1(0/0)	—	—	—	—	—	—	—
extb.l	Dx	1(0/0)	—	—	—	—	—	—	—
neg.l	Dx	1(0/0)	—	—	—	—	—	—	—
negx.l	Dx	1(0/0)	—	—	—	—	—	—	—
not.l	Dx	1(0/0)	—	—	—	—	—	—	—
scc	Dx	1(0/0)	—	—	—	—	—	—	—
swap	Dx	1(0/0)	—	—	—	—	—	—	—
tst.b	<ea>	1(0/0)	4(1/0)	4(1/0)	4(1/0)	4(1/0)	5(1/0)	4(1/0)	1(0/0)
tst.w	<ea>	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	<ea>	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)

3.11 STANDARD TWO OPERAND INSTRUCTION EXECUTION TIMES

Table 3-13. Two Operand Instruction Execution Times

OPCODE	<EA>	EFFECTIVE ADDRESS							
		Rn	(An)	(An)+	-(An)	(d16,An) (d16,PC)	(d8,An,Xi*SF) (d8,PC,Xi*SF)	xxx.wl	#xxx
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)	—
eorl.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)	—	—	—
mac.l	Ry,Rx,ea,Rw	—	5(1/0)	5(1/0)	5(1/0)	5(1/0)	—	—	—
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)	—	—	—
moveq	#imm,Dx	—	—	—	—	—	—	—	1(0/0)
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)	—	—	—

Table 3-13. Two Operand Instruction Execution Times

OPCODE	<EA>	EFFECTIVE ADDRESS							
		Rn	(An)	(An)+	-(An)	(d16,An) (d16,PC)	(d8,An,Xi*SF) (d8,PC,Xi*SF)	xxx.wl	#xxx
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)	—
ori.l	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
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)	—	—	—	—	—	—	—

3

3.12 MISCELLANEOUS INSTRUCTION EXECUTION TIMES

Table 3-14. Miscellaneous Instruction Execution Times

OPCODE	<EA>	EFFECTIVE ADDRESS							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xi*SF)	xxx.wl	#xxx
cpushl	(Ay)	—	11(0/1)	—	—	—	—	—	—
halt		6(0/0)	—	—	—	—	—	—	—
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	<ea>,&list	—	$2+n(n/0)^2$	—	—	$2+n(n/0)^2$	—	—	—
movem.l	&list,<ea>	—	$2+n(0/n)^2$	—	—	$2+n(0/n)^2$	—	—	—
nop		3(0/0)	—	—	—	—	—	—	—
pea	<ea>	—	2(0/1)	—	—	$2(0/1)^4$	$3(0/1)^5$	2(0/1)	—
pulse		1(0/0)	—	—	—	—	—	—	—
stop	#imm	—	—	—	—	—	—	—	3(0/0) ³
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	<ea>	—	7(1/0)	7(1/0)	7(1/0)	7(1/0)	8(1/0)	7(1/0)	—
wdebug	<ea>	—	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 transferred by the MOVEM opcode.
³ The execution time for STOP is the time required until the processor begins sampling continuously for interrupts.
⁴ PEA execution times are the same for (d16,PC).
⁵ PEA execution times are the same for (d8,PC,Xi*SF).

3.13 BRANCH INSTRUCTION EXECUTION TIMES

Table 3-15. General Branch Instruction Execution Times

OPCODE	<EA>	EFFECTIVE ADDRESS							
		Rn	(An)	(An)+	-(An)	(d16,An) (d16,PC)	(d8,An,Xi*SF) (d8,PC,Xi*SF)	xxx.wl	#xxx
bsr		—	—	—	—	1(0/1) ²	—	—	—
jmp	<ea>	—	5(0/0)	—	—	5(0/0) ¹	6(0/0)	1(0/0) ¹	—
jsr	<ea>	—	5(0/1)	—	—	5(0/1)	6(0/1)	1(0/1) ²	—
rte		—	—	14(2/0)	—	—	—	—	—
rts		—	—	8(1/0)	—	—	—	—	—

Table 3-16. BRA, Bcc Instruction Execution Times

OPCODE	FORWARD TAKEN	FORWARD NOT TAKEN	BACKWARD TAKEN	BACKWARD NOT TAKEN
bra	1(0/0) ¹	—	1(0/0) ¹	—
Bcc	5(0/0)	1(0/0)	1(0/0) ³	5(0/0)

The following notes apply to the branch execution times:

1. For the jmp <ea> instructions, where <ea> is (d16,PC) or xxx.wl, the branch acceleration logic of the Instruction Fetch Pipeline calculates the target address and begins prefetching the new path. Since the Instruction Fetch and Operand Execution Pipelines are decoupled by the FIFO instruction buffer, the execution time can vary from 1 to 3 cycles, depending on the amount of decoupling.

This same mechanism is used for the bra opcode.

For all other <ea> values of the jmp instruction, the branch acceleration logic is not used, and the execution times are fixed.

2. For the jsr xxx.wl opcodes, the same branch acceleration mechanism is used to initiate the fetch of the target instruction. Depending on the amount of decoupling between the IFP and OEP, the resulting execution times can vary from 1 to 3 cycles.

The same acceleration techniques are applied to the bsr opcode.

For the remaining <ea> values for the jsr instruction, the branch acceleration logic is not used, and the execution times are fixed.

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

```

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

```

The execution times in the BRA, Bcc Table assume that CCR[7] is negated. Another representation of the Bcc execution times is shown below:

3

Table 3-17. Another Table of Bcc Instruction Execution Times

OPCODE	PREDICTED CORRECTLY AS TAKEN	PREDICTED CORRECTLY AS NOT-TAKEN	MISPREDICTED
Bcc	1(0/0)	1(0/0)	5(0/0)

The execution time for the “predicted correctly as taken” column can vary between 1 to 3 cycles depending on the amount of decoupling between the Instruction Fetch and Operand Execution Pipelines as previously discussed.

SECTION 4 PHASE LOCKED LOOP

4.1 PLL FEATURES

- The PLL locks to an input clock frequency at the CLKIN input and provides a processor clock frequency that is 2X the input clock frequency and a programmable system bus clock output that is 1/2, 1/3, or 1/4 the processor clock frequency.
- The PLL input frequency may range from 16-45MHz, giving a maximum processor clock frequency of 90MHz.
- A reduced power mode will turn off the processor clock (PSTCLK) when a STOP instruction is executed.

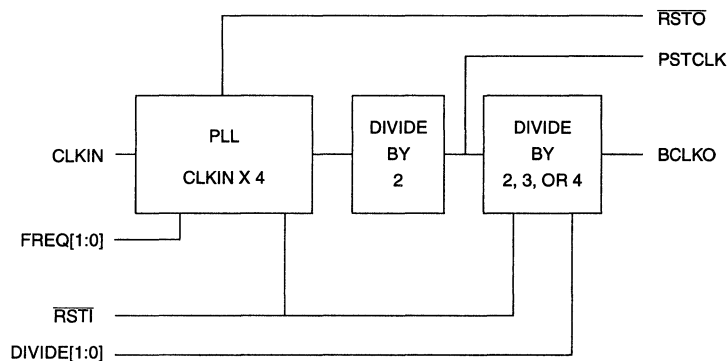


Figure 4-1. Block Diagram of Phase-Locked Loop Module

4.2 PLL SPECIFICATIONS

- PLL lock time: 2.2ms at CLKIN running 45MHz
- Input Frequency (CLKIN): 16.67MHz through 45MHz
- Output Frequency (BCLKO): BCLKO=PSTCLK/2 (16.7MHz through 45MHz) or BCLKO=PSTCLK/3 (11.1MHz through 30MHz) or BCLKO=PSTCLK/4 (8.25MHz through 22.5MHz)
- Internal Processor Clock (PSTCLK): PSTCLK=CLKIN x 2 (33.3MHz through 90MHz)

4.3 PLL OPERATION

4.3.1 Normal Mode

A user-supplied CLKIN input frequency, within the range of 16.67 through 45 MHz, is multiplied up to create a 4X internal clock. This internal clock is divided by two to create the processor clock (PSTCLK). You must specify the frequency range of the input clock using FREQ[1:0] during reset.

The processor clock is divided to create the system bus clock. The bus clock (BCLKO) divisor is determined by the logic level of the DIVIDE pins during reset. You can program the bus clock to be 1/2, 1/3 or 1/4 the frequency of the MCF5307 processor core clock, PSTCLK.

4.3.2 Reset / Initialization

The PLL receives \overline{RSTI} as an input directly from the pin. Additionally, four signals are multiplexed with D[3:0] (FREQ[1:0]:DIVIDE[1:0]) while \overline{RSTI} is asserted. These signals are sampled during reset and registered by the PLL on the negation of \overline{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 BCLKO/PSTCLK divide ratio, respectively.

4.3.3 Reduced-Power Mode

The PLL processor core clock output, PSTCLK, can be turned off in an orderly, 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. PSTCLK is disabled using the CPU STOP instruction and will resume normal operation on interrupt, as described in the following subsection describing the PLL control register (PLLCR).

4.3.4 Phase Locked Loop Control Register (PLLCR)

The ENBSTOP bit must be set to 1 for the ColdFire CPU STOP instruction to be acknowledged. After reset, this bit will be cleared to 0 and must be set to 1 if you expect the MCF5307 to enter low-power modes. Only clocks to the core processor are turned off because of the CPU STOP instruction. All internal modules remain clocked and can generate interrupts to restart the ColdFire Core.

The PLLCR control register bits PLLIPL[2:0] determine the minimum level at which an interrupt (decoded as an Interrupt Priority Level or IPL) must occur to awaken the PLL. The PLL will then turn clocks back on to the core processor and interrupt exception processing will take place. Table 4-1. outlines the settings of PLLIPL[2:0] and the interrupt ranges that will wake the core processor from a CPU STOP instruction.

The PLLCR is an 8-bit supervisor read-write register.

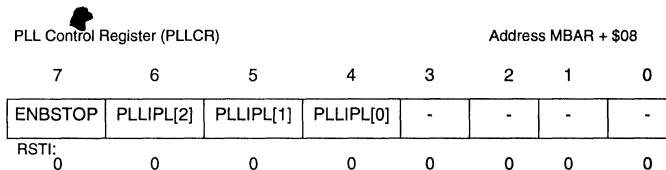


Figure 4-2. PLL Control Register (PLLCR)

4

ENBSTOP- Enable CPU STOP instruction.

0 = ENBSTOP disabled.

1 = ENBSTOP enabled, STOP instruction will turn off clocks to ColdFire core.

PLLIPL[2:0]- PLL Interrupt Priority Level to wake up from CPU STOP.

Refer to Table 4-1.

Table 4-1. Settings of PLLIPL[2:0] and Interrupts That Wake Up Core

PLLIPL[2:0]	INTERRUPTS THAT WILL TURN ON CLOCKS TO THE CORE PROCESSOR.
000	Any interrupts will wake core
001	Interrupts[7:2]
010	Interrupts[7:3]
011	Interrupts[7:4]
100	Interrupts[7:5]
101	Interrupts[7:6]
110	Only Interrupt[7]
111	No interrupts will wake core

4.4 PLL PORT LIST

4.4.1 Inputs

CLKIN

Input clock to the PLL. The input clock frequency may be between 16.67 MHz and 45 MHz. Input frequency may not change during normal operation. Changes are only recognized at reset.

$\overline{\text{RSTI}}$

Active-low asynchronous input that indicates PLL is to enter reset mode. As long as $\overline{\text{RSTI}}$ is asserted, the PLL will be held in reset and will not begin to lock.

FREQ[1:0]

This 2-bit input bus indicates the frequency range of CLKIN. These signals are multiplexed with D[3:2] and are sampled while $\overline{\text{RSTI}}$ is asserted. Table 4-2 shows the bit settings required for correct PLL operation. These signals must be set to the correct values for proper operation. Note that these signals do not affect the frequency, but are required to set up the analog PLL.

Table 4-2. CLKIN Frequency

FREQ[1:0] / D[3:2]	FREQUENCY OF CLKIN (MHZ)
00	16.6 - 27.999
01	28 - 38.999
10	39 - 45
11	NOT USED

DIVIDE[1:0]

This 2-bit input bus selects the BCLKO/PSTCLK ratio. These signals are sampled while $\overline{\text{RSTI}}$ is asserted and selects the ratios as indicated in Table 4-3 shown below.

Table 4-3. BCLKO/PSTCLK Divide Ratios

DIVIDE[1:0] / D[1:0]	RATIO OF BCLKO/PSTCLK
01	NOT USED
10	1/2
11	1/3
00	1/4

4.4.2 Outputs

BCLKO

This system bus clock output provides a divided version of the processor clock frequency (16.67MHz-45MHz), determined by DIVIDE[1:0].

PSTCLK

This clock signal output provides a buffered processor clock frequency at 2X CLKIN.

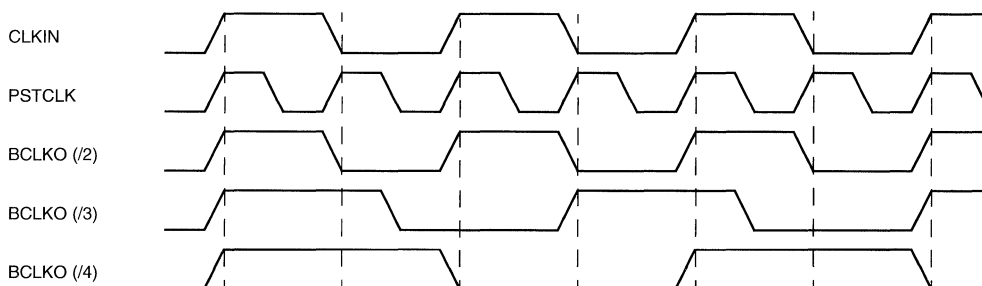
 $\overline{\text{RSTO}}$

This output provides an external reset for peripheral devices.

4.5 TIMING DIAGRAMS

4.5.1 PSTCLK and BCLKO

Figure 4-3 illustrates the timing between CLKIN, PSTCLK and the three possible versions of BCLKO. There will be an unspecified skew between CLKIN and PSTCLK, as well as CLKIN to BCLKO. PSTCLK is a delayed copy of the internal processor clock, so the skew between PSTCLK and BCLKO is also unspecified.



NOTE: THESE CLOCK SIGNALS ARE SHOWN WITH EDGES ALIGNED TO ILLUSTRATE FREQUENCY RELATIONSHIP. ACTUAL EDGES WILL HAVE SOME SKEW BETWEEN THEM.

Figure 4-3. CLKIN, PSTCLK, and BCLKO Timing

4.5.2 RSTI Timing

Figure 4-4 illustrates PLL timing during reset. As shown, \overline{RSTI} must be held low for a minimum of 80 cycles of CLKIN to allow time for the MCF5307 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{RSTI} given in the AC Specifications of this manual. On the rising edge of \overline{RSTI} , the data is latched and the PLL begins ramping to its final operating frequency. During this ramp and lock time, BCLKO will be held low. The PLL will lock in about 2.2ms with a CLKIN of 45MHz, at which time BCLKO and PSTCLK will begin normal operation in the specified mode. Actual lock time is dependent on the clock frequency at the input of the PLL. The PLL requires 100,000 CLKIN cycles to guarantee PLL lock. To allow for reset of external peripherals requiring a clock source, the \overline{RSTO} pin will remain asserted for a number of BCLKO cycles as shown in Figure 4-4.

4

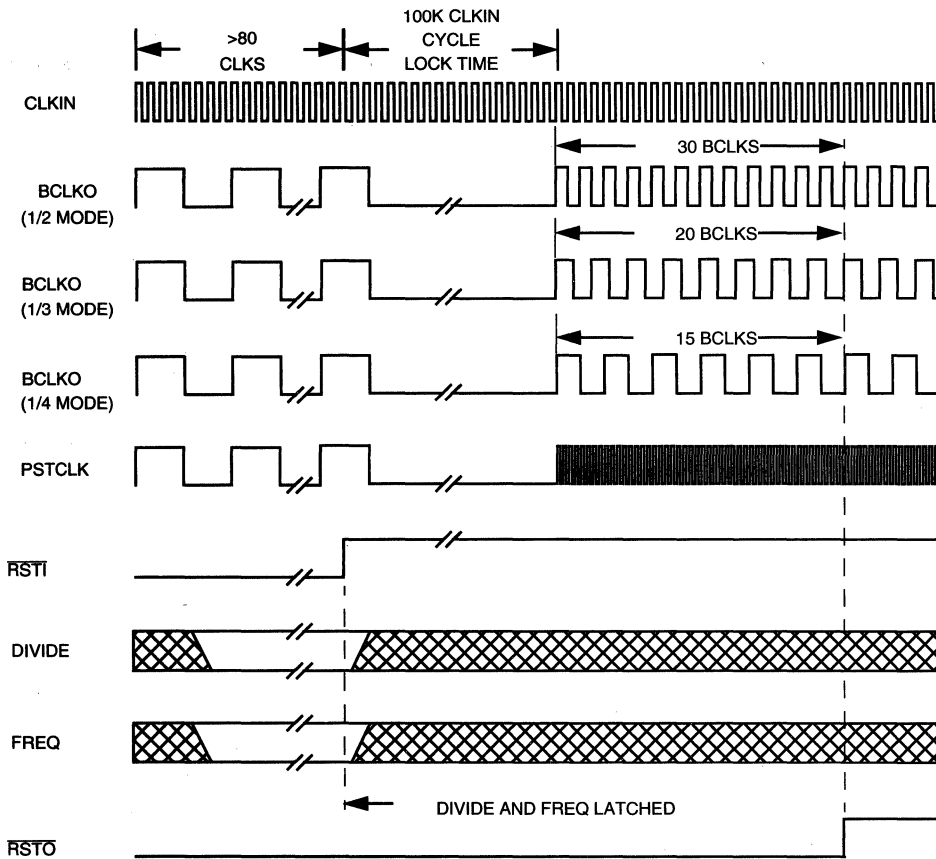


Figure 4-4. Reset and Initialization Timing

SECTION 5 CACHE

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 data to the MCF5307 instruction and data pipes, which decouples processor performance from system memory performance, resulting in an increase in bus availability for alternate bus masters.

The MCF5307 nonblocking cache services read hits or write hits from the processor while a fill (caused by a cache allocation) is in progress.

As shown in Figure 5-1, both instruction and data accesses are performed using a single bus connected to the cache. All addresses from the processor to the cache are physical addresses. If the address matches one of the cache entries, the access hits in the cache. For a read operation, the cache supplies the data to the processor, and for a write operation, the data from the processor updates the cache. If the access does not match one of the cache entries (misses in the cache) or 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 bus controller (SIM). Throughout this section, all cache accesses on the internal bus have a corresponding access on the external bus by way of the SIM.

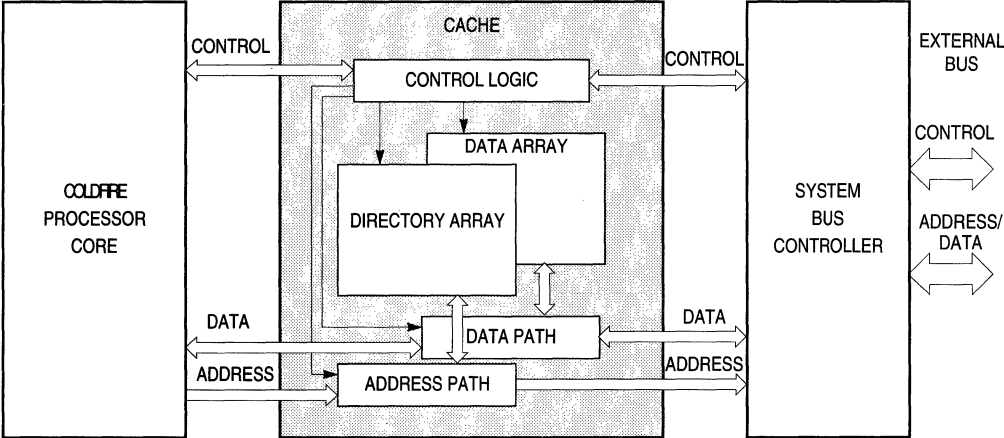


Figure 5-1. MCF5307 Unified Cache

The MCF5307 device does not implement a bus snooper. You must use software to maintain cache coherency with other possible bus masters.

5.1 CACHE ORGANIZATION

The 4-way set associative cache is organized as four levels (ways) of 128 sets with each line containing 16 bytes (four longwords) of storage. Figure 5-2 illustrates the cache organization (as well as the terminology used) along with the cache line format.

Address bits A[10:4] provide an index to select a set. Levels are selected according to the rules of set association (discussed under **5.2 Cache Operation**).

Each line consists of an address tag (upper 21 bits of the address), two status bits, and four longwords of data. The two status bits consist of a valid bit and a dirty bit for the line. Address bits A3 and A2 select the longword within the line.

5

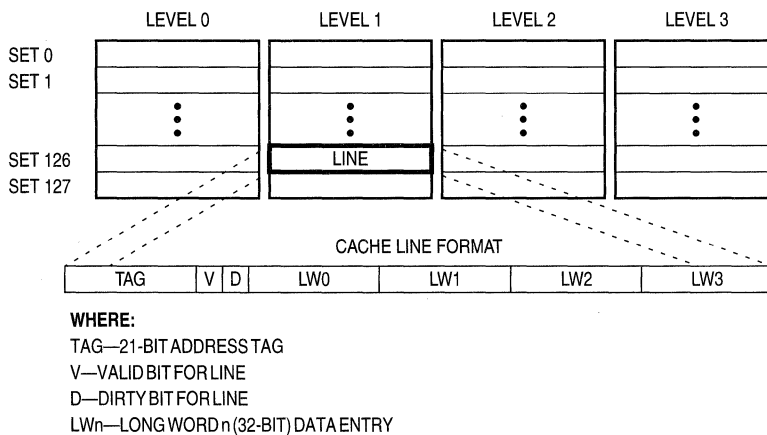


Figure 5-2. Cache Organization and Line Format

5.2 CACHE OPERATION

The cache stores an entire line, thereby providing validity on a line-by-line basis. For burst-mode accesses, only those that successfully read four longwords can be cached.

A cache line is always in one of three states: invalid, valid, or dirty. For invalid lines, the V-bit is clear, causing the cache line to be ignored during lookups. Valid lines have their V-bit set and D-bit cleared, indicating the line contains valid data consistent with memory. Dirty cache lines have the V- and D-bits set, indicating that the line has valid entries that have not been written to memory.

A cache line changes states from valid or dirty to invalid if the execution of the CPUSHL instruction explicitly invalidates the cache line. The cache should be explicitly cleared by setting the CINVA bit of the CACR after a hardware reset because reset does not invalidate

the cache lines. Following initial power-up, the cache contents will be undefined. The V- and D-bits may be set on some lines, necessitating the clearing of the cache before it is enabled.

Figure 5-3 illustrates the general flow of a caching operation. To determine if the address is already allocated in the cache, (1) the cache set index A10-A4 is used to select one cache set. A set is defined as the grouping of four lines (one from each level), corresponding to the same index into the cache array. (2) The address bits of higher order (A31-A11) than the cache set index (A10-A4) are used as a tag reference or used to update the cache line tag field. (3) The four tags from the selected cache set are compared with the tag reference. If any one of the four tags matches the tag reference and the tag status is either valid or dirty, a cache hit has occurred. (4a) A cache hit indicates that the data entries (LW0-LW3) in that cache line contain valid data (for a read access), or (4b) can be written with new data (for a write access).

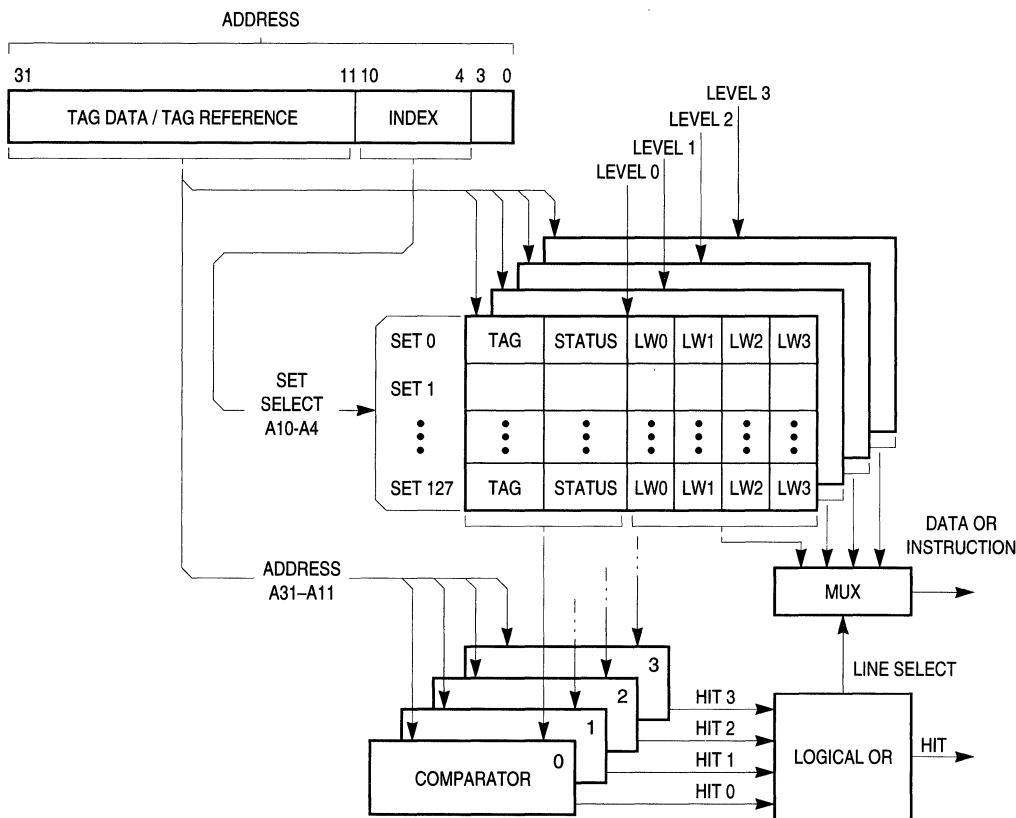


Figure 5-3. Caching Operation

To allocate an entry into the cache, the cache set index (A10-A4) is used to select one of the cache's 128 sets. The status of each of the four cache lines for the selected set is examined. The cache control logic first looks for an invalid cache line to use for the new entry. If no invalid cache lines are available, a line from one of the four levels must be deallocated to host the new entry. The cache controller uses a pseudo-round-robin replacement algorithm to determine which cache line will be deallocated and replaced. After a cache line is allocated, the replacement pointer increments to point to the next level. During half-cache lock operation (HLCK equal to 1), the replacement pointer is forced to point to either level 2 or level 3.

In the process of deallocation, a cache line that is valid and not dirty is invalidated. A dirty cache line is placed in a push buffer (to do an external cache line push) before being invalidated. Once a cache line is invalidated, a new entry can replace it.

5

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

1. The new address tag bits A[31:A11] are written to the tag
2. The data bits LW0–LW3 are updated with the new memory data
3. The cache line status changes to a valid state.

Read cycles that miss in the cache allocate normally as previously described. Write cycles that miss in the cache do not allocate on a cachable writethrough region, but do allocate for addresses in a cachable copyback region. A copyback byte, word, longword, or line write miss will

1. Cause the cache to initiate a line fill,
2. Allocate space for a new line,
3. Set the status bits to indicate valid and dirty
4. Write the data in the allocated space. No write to memory occurs.

Read hits do not change the status of the cache line and no deallocation or replacement occurs. Write hits in cacheable writethrough regions perform an external write cycle; write hits in cacheable copyback regions do not perform an external write cycle.

If the cache hits on a read access, data is driven back to the processor core. If the cache hits on a write access, the data is written to the appropriate portion of the accessed cache line. If the data access is misaligned, the misalignment module breaks up the access into a sequence of smaller aligned cache accesses. Any misaligned operand reference generates at least two cache accesses. Because entry validity is provided only on a line basis, the entire line must be loaded from system memory on a cache miss for the cache to contain any valid information for that line address.

Noncacheable write accesses (i.e., those designated as cache-inhibited by the Cache Control Register (CACR) or Access Control Register (ACR)) bypass the cache and perform a corresponding external write. Normally, noncacheable read accesses bypass the cache and the read access is performed on the external bus. The exception to this normal operation occurs when all of the following conditions are true during a noncacheable read:

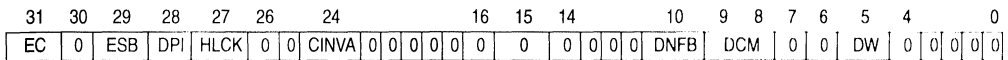
- Noncacheable fill buffer bit (DNFB) is set in the Cache Control Register (CACR)
- Access is an instruction read
- Access is normal (i.e., transfer type (TT) equals 0)

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 another fill occurs or a cache invalidate all occurs.

Valid cache entries that match during noncacheable address accesses are neither pushed nor invalidated. Such a scenario suggests that the associated cache mode for this address space was changed. Use the CPUSHL instruction to push and/or invalidate the cache entry, or set the CINVA bit of the CACR to invalidate the entire cache before switching cache modes.

5.3 CACHE CONTROL REGISTER (CACR)

The CACR is a 32-bit register that contains cache control information. The CACR can be written via the MOVEC register instruction (register control field of the MOVEC instruction = \$002). A hardware reset clears the CACR, which disables the cache; however, reset does not affect the tags, state information, and data within the cache. The CACR is illustrated below. NOTE - all bits shown as “0” are reserved



Cache Control Register (CACR)

EC—Enable Cache

- 0 = cache disabled
- 1 = cache enabled

Bit 30—Reserved

ESB — Enable Store Buffer

- 0 = all writes to writethrough or noncacheable imprecise space will bypass the store buffer and generate bus cycles directly.
- 1 = the 4 entry first-in-first-out (FIFO) store buffer is enabled; this buffer defers pending writes to writethrough or cache-inhibited imprecise regions to maximize performance

Accesses to cache-inhibited precise space always bypass the store buffer.

DPI—Disable CPUSHL Invalidation

- 0 = each cache line is invalidated as it is pushed
- 1 = CPUSHLed lines remain valid in the cache

HLCK—1/2 Cache Lock Mode

- 0 = cache operates in normal full cache mode
- 1 = cache operates in one-half cache lock mode

CACHE

When this mode is enabled, levels 0 and 1 of the cache are locked such that their lines will never be displaced or allocated.

Bits 26–25—Reserved

CINVA—Cache Invalidate All

- 0 = no invalidation is performed
- 1 = initiate an invalidation of the entire cache

Writing a 1 to this bit will initiate entire cache invalidation. Once invalidation is complete, this bit will automatically return to 0 (i.e., you do not have to set it back to 0). This bit is always read as a 0.

5

Bits 23–11—Reserved

DNFB—Default Noncacheable Fill Buffer

- 0 = fill buffer is not used to store noncacheable accesses
- 1 = fill buffer is used to store noncacheable accesses
- fill buffer used only for normal (TT = 0) instruction reads of a noncacheable region
- the instructions are loaded into the fill buffer via a burst access (same as a line fill)

Note

It is possible that this feature can cause a coherency problem for self-modifying code. If enabled and a noncacheable access occurs that uses the fill buffer, the instructions remain valid in the fill buffer until a cache-invalidate-all instruction is issued, another noncacheable burst, or any miss that initiates a fill occurs. If a write occurs to the line in the fill buffer, the write will go to the external bus without updating or invalidating the fill buffer. Any subsequent reads of that written data will be serviced by the fill buffer and receive stale information.

DCM—Default Cache Mode

This field selects the default cache mode and access precision as follows:

- 00 = cacheable, writethrough
- 01 = cacheable, copyback
- 10 = cache-inhibited, precise exception model
- 11 = cache-inhibited, imprecise exception model

Bits 7,6—Reserved

DW—Default Write Protect

This bit indicates the default write privilege.

- 0 = read and write accesses permitted
- 1 = write accesses not permitted

Bits 4–0—Reserved

5.4 ACCESS CONTROL REGISTERS

The 32-bit Access Control Registers (ACR0 and ACR1) assign access control attributes to specific regions of address space. The ACR registers can be written via the MOVEC instruction. (ACR0 has register control field of the MOVEC instruction = \$004; ACR1 has register control field of the MOVEC instruction = \$005). For overlapping regions, ACR0 takes priority. The control attributes are cache mode and write protection. Data transfers to and from these registers are longword transfers. The register below illustrates the ACR format. The paragraphs that follow describe the fields within the ACR. Bits 12–7, 4, 3, 1, and 0 always read as zero. At reset, the enable bit is forced to zero.

31	24 23	16 15	14 13	12	11	10	9	8	7	6	5	4	3	2	1	0				
ADDRESSBASE			ADDRESSMASK				E	S- FIELD	0	0	0	0	0	0	CM	0	0	W	0	0

5

Access Control Register Format (ACR0, ACR1)

Bits 31–24— Address Base

This 8-bit field is compared with address bits A31–A24. Addresses that match in this comparison (and are otherwise eligible) are assigned the access control attributes of this register.

Bits 23–16— Address Mask

Because this 8-bit field contains a mask for the address base field, setting a bit in this field causes the corresponding bit in the address base field to be ignored. Regions of memory larger than 16 Mbytes can be assigned the access control attributes of this register by setting some of the address mask bits to 1's. The low-order bits of this field can be set to define contiguous regions larger than 16 Mbytes. The mask can define multiple noncontiguous regions of memory.

E—Enable

This bit enables or disables the access control attributes of the region defined by this register:

- 0 = access control attributes disabled
- 1 = access control attributes enabled

S—Supervisor Mode

This field specifies the way FC2 matches an address:

- 00 = match only if FC2 = 0 (user mode access)
- 01 = match only if FC2 = 1 (supervisor mode access)
- 1X = ignore FC2 when matching

CACHE

Bits 12–7—Reserved by Motorola

CM—Cache Mode

This field selects the cache mode and access precision as follows:

- 00 = cacheable, writethrough
- 01 = cacheable, copyback
- 10 = cache-inhibited, precise exception model
- 11 = cache-inhibited, imprecise exception model

W—Write Protect

This bit indicates the write privilege of the ACR region.

- 0 = read and write accesses permitted
- 1 = write accesses not permitted

Bits 4,3,1,0—Reserved by Motorola

5

5.5 CACHE MANAGEMENT

By using the MOVEC instruction to access the CACR, you enable and configure the cache. A hardware reset clears the CACR, disabling the cache, and removing all configuration information; however, reset does not affect the tags, state information, and data within the cache. You must set the CINVA bit in the CACR to invalidate the cache before enabling it.

The CINVA bit of the CACR allows invalidation of the entire cache only. The privileged CPUSHL instruction supports cache management by selectively pushing and invalidating an individual cache line. The address register used with the CPUSHL instruction directly addresses the cache's directory array. The CPUSHL instruction will either push and invalidate a line, or push and leave the line valid, depending on the state of the DPI bit of the CACR. To push the entire cache, you must implement a software loop to index through all 128 sets and each of the 4 lines within each set (for a total of 512 lines). The state of the cache enable bit in the CACR does not affect the operation of CPUSHL instruction nor the CINVA bit of the CACR.

The CPUSHL instruction flushes a cache line. The instruction format is shown below, where AN is an address register.

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

The contents of AN used with the CPUSHL instruction specify cache row and line indexes. This differs from the MC68040 where AN specifies a physical address. The format for AN is

31						11	10				4	3	2	1	0
			0								Set Index				Line Index

Bits A10 - A4 specify a set index and bits 3-0 specify the cache level. Address bits A3-A2 are “don't cares” for this instruction.

The following code example flushes the entire M5307 unified cache:

```

_cache_flush:
    nop                ; synchronize - flush store buffer
    moveq.l           #0,d0        ; disable cache...
    movec.d0          cacr         ; ... by clearing the cacr

    moveq.l           #0, d0        ; zero levelcounter
    moveq.l           #0, d1        ; zero setcounter
    move.l            d0, a0        ; initialize An

setloop:
    cpushl            bc, (a0)      ; push the cache line identified by a0
    add.l             #0x0010, a0    ; increment setindex by 1
    addq.l            #1, d1        ; increment setcounter

    cmpi.l            #128, d1      ; check if sets for current level are done

    bne               setloop      ; more sets to flush

    moveq.l           #0, d1        ; zero set counter
    addq.l            #1, d0        ; increment level counter
    add.l             d0, d1        ; form set and level for An
    move.l            d1, a0        ; initialize An
    cmpi.l            #4, d0        ; check if levels are done
    bne               setloop

    rts

```

5

5.6 CACHING MODES

Every cache access has an associated caching mode that determines how the cache handles the access. An access can be cacheable in either the writethrough or copyback modes, or it can be cache-inhibited in precise or imprecise modes. For normal accesses, the CM field (from the ACR) corresponding to the address of the access specifies one of these caching modes. When the access address does not match either of the ACRs, the default caching mode defined by the DCM field of the CACR is used.

Addresses matching an ACR can also be write-protected using the W bit of that ACR. Addresses that do not match either of the ACRs can be write-protected using the DW bit of the CACR.

Reset disables the cache and places 0's in all CACR and ACR bits. Consequently, after reset, the defaults are writethrough cache mode and no addresses are write-protected. Note that you—and not reset—invalidate cache entries.

The ACRs allow the defaults to be overridden. In addition, some instructions (e.g., CPUSHL) and processor core operations perform accesses that have an implicit caching mode associated with them. The following paragraphs discuss the different caching accesses and their related cache modes.

5.6.1 Cacheable Accesses

If the CM field of an ACR or the default field of the CACR indicates writethrough or copyback, the access is cacheable. A read access to a writethrough or copyback region is read from the cache if matching data is found. Otherwise, the data is read from memory and updates the cache. When a line is being read from memory for both a writethrough read miss and a copyback read miss, the longword within the line that contains the core-requested data is fetched first, and the requested data is given immediately to the processor. This operation releases the processor while the remaining three longwords of the line are read from memory and stored in the cache.

The following paragraphs describe the writethrough and copyback modes in detail.

5

5.6.1.1 WRITETHROUGH MODE. Write accesses to regions specified as writethrough are always passed on to the external bus, although the cycle can be buffered (depending on the state of the ESB bit in the CACR). Writes in writethrough mode are handled with a no-write-allocate policy—i.e., 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 instruction or data-read accesses that hit in the cache; read misses cause a new cache line to be loaded into the cache.

5.6.1.2 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 D-bit without an external bus access. The dirty 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 dirty cache line for replacement, the current 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.

5.6.2 Cache-Inhibited Accesses

You can designate as cache-inhibited those address space regions containing targets such as I/O devices and shared data structures in multiprocessing systems. If the corresponding CM field (of the ACR) or DCM field (of the CACR) indicates precise or imprecise, then the access is cache-inhibited. The caching operation is identical for both cache-inhibited modes. The difference between these inhibited cache modes has to do with recovery from an external bus error.

Noncacheable write accesses bypass the cache and a corresponding external write is performed. Normally, noncacheable read accesses bypass the cache and the read access is performed on the external bus. The exception to this normal operation occurs when all of the following conditions are true during a noncacheable read:

- The noncacheable fill buffer bit (DNFB) is set in the Cache Control Register (CACR)
- Access is an instruction read

- Access is normal (i.e., 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 another fill occurs or a “cache invalidate all” occurs.

If the CM field indicates either noncacheable precise or noncacheable imprecise modes, the cache controller bypasses the cache and performs an external transfer. If a cache line matching the current address is already resident in the cache and the cache mode for that region is cache-inhibited, the cache does not automatically push the line if it is dirty, nor does it invalidate the line if it is valid. You should first execute a CPUSHL instruction or set the CINVA bit of the CACR (to invalidate the entire cache) prior to switching the cache mode.

If the CM field indicates precise mode, the sequence of read and write accesses to the region is guaranteed to match the sequence of the instruction order. 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 will not be deferred past operand-read accesses that miss in the cache (i.e., that must be read from the bus). Precise operation forces operand-read accesses for an instruction to occur only once by preventing the instruction from being interrupted after the operand-fetch stage. Otherwise, *if not in precise mode* and an exception occurs, the instruction is aborted and the operand may be accessed again when the instruction is restarted. These guarantees apply only when the CM field indicates the precise mode and the accesses are aligned.

All CPU space-register accesses (e.g. MOVEC) are always treated as noncacheable and precise.

5.7 CACHE PROTOCOL

The following paragraphs describe the cache protocol for processor accesses and assumes that the data is cacheable (i.e., writethrough or copyback).

5.7.1 Read Miss

A processor read that misses in the cache requests a bus transaction causes from the cache controller. 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.

5.7.2 Write Miss

The cache controller handles processor writes that miss in the cache differently for writethrough and copyback regions. Byte, word, longword, or line write misses to copyback regions load the cache line from an MBUS line read. The new cache line is then updated with write data and the D-bit for the line is set, leaving the cache line in the dirty state. Write misses to writethrough regions write directly to memory without loading the corresponding cache line into the cache.

5.7.3 Read Hit

On a read hit, the cache provides the data to the processor core. No MBUS transaction is performed 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 dirty data will not be pushed out to memory when a read hit occurs within that line. You should first execute a CPUSHL instruction or set the CINVA bit of the CACR (to invalidate the entire cache) before switching the cache mode.

5.7.4 Write Hit

The cache controller handles processor writes that hit in the cache differently for writethrough and copyback regions. For write hits to a writethrough region, the portions of the cache line(s) corresponding to the size of the access are updated with the data. The data is also written to the external memory. The cache line state remains unchanged. If the access is copyback, the cache controller updates the cache line and sets the D-bit for the line. An external write is not performed and the cache line state changes to (or remains in) the dirty state.

5

5.8 CACHE COHERENCY

The MCF5307 provides limited support for maintaining cache coherency in multimaster environments. Both writethrough and copyback memory update techniques are supported to maintain coherency between the cache and memory.

The cache does not support snooping (i.e., cache coherency is not supported while alternate masters are using the bus).

5.9 MEMORY ACCESSES FOR CACHE MAINTENANCE

The cache controller performs all maintenance activities that supply data from the cache to the processor core. The activities include requesting accesses to the SIM for reading new cache lines and writing dirty cache lines to memory. The following paragraphs describe the memory accesses resulting from cache-fill and push operations. Refer to **Section 7 Bus Operation** for detailed information about the required bus cycles.

5.9.1 Cache Filling

When a new cache line is required, the internal bus controller requests a line read from the SIM. The SIM requests a burst-read transfer by indicating a line access with the size signals (SIZ[1:0]).

The responding device supplies four longwords of data in sequence. If the responding device does not support the burst mode, it should assert the $\overline{\text{TBI}}$ signal for the first longword of the line access. The SIM responds by terminating the line access and completing the remainder of the line read as three sequential longword reads.

SIM line accesses implicitly request burst-mode operations from memory. For more information regarding burst-mode accesses on the external bus, see the Bus Operations section for additional detail.

When a cache line read is initiated, the first cycle tries to load the longword entry corresponding to the address the processor core requested. Subsequent transfers are for the remaining longword entries in the cache line.

A bus error occurring during a burst operation aborts the operation. If the bus error occurs during the first cycle of a burst, the data from the bus is ignored and the line is not cached. If the access is a data cycle, exception processing proceeds immediately. If the cycle is for an instruction prefetch, a bus-error exception is not taken immediately, but will be taken if the instruction flow subsequently causes an attempt of the instruction. Refer to the Bus Operations section for more information about this operation.

When a bus error occurs on the second cycle or later, the burst operation aborts and the line is not cached. The processor may or may not take an exception, depending on the status of the pending data request. If the bus-error cycle contains a portion of a data operand that the processor is specifically waiting for (e.g., the second half of a misaligned operand), the processor immediately takes an exception. Otherwise, no exception occurs and the cache line fill is repeated the next time data within the line is required.

5.9.2 Cache Pushes

When the cache controller selects a dirty cache line for replacement, memory must be updated with the dirty data before the line is replaced. 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 dirty 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 dirty cache line is written back to memory and the push buffer is invalidated.

5.10 PUSH AND STORE BUFFERS

The push buffer reduces latency for requested new data on a cache miss by temporarily holding displaced dirty data while the new data is fetched from memory. The push buffer contains 16 bytes of storage (one displaced cache line).

If a cache miss displaces a dirty line, the miss read reference is immediately placed on the MBUS. While waiting for the response, the current contents of the cache location load into the push buffer. Once the bus transaction (burst read) completes, the cache controller can generate the appropriate line-write bus transaction to write the contents of the push buffer into memory.

The store buffer implements a FIFO buffer that can defer pending writes to imprecise regions in order to maximize performance. The store buffer can support as many as four entries (16 bytes maximum) for this purpose.

For operand writes destined for the store buffer, the processor core incurs no stalls. The store buffer effectively provides a measure of decoupling between the pipeline's ability to generate writes (one write per cycle maximum) and the ability of the external bus to retire those writes. When writing to imprecise regions, a stall will occur only in the event of a full store buffer and a write operation is present on the internal bus. The internal write cycle is held, stalling the operand execution pipeline.

If the store buffer is not used (i.e., 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 EX cycle of the operand execution pipeline (OEP) until external bus transfer termination is received. This means each write operation is stalled for 5 cycles, making the minimum write time equal to 6 cycles when the store buffer is not used.

The store buffer enable bit (bit ESB of the CACR) controls the enabling of the store buffer. This bit can be set and cleared via the MOVEC instruction. At reset, this bit is cleared and all writes are precise. The ACR CM field or CACR DCM field generates the mode used when this bit is set. The cacheable writethrough and the cache-inhibited imprecise modes use the store buffer.

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

5

5.11 PUSH AND STORE BUFFER BUS OPERATION

Once 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 (e.g., cache miss to process) during the continued instruction execution by the processor pipeline, the pipeline will stall until the push and store buffers are empty before generating the required external bus transaction.

Certain instructions and exception processing that synchronize the processor core guarantee the push and store buffers are empty before proceeding.

5.12 CACHE OPERATION SUMMARY

The following paragraphs discuss the operational details for the cache and present state diagrams depicting the cache line state transitions.

The cache supports a line-based protocol allowing individual cache lines to be in one of three states: invalid, valid, or dirty. To maintain coherency with memory, the cache supports both writethrough and copyback modes, specified by the CM field for the matched ACR or the DCM field of the CACR if no ACR matches.

Read misses and write misses to copyback regions cause the cache controller to read a new 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 for the line. If all lines in the row are already valid or dirty, the pseudo-round-robin replacement algorithm selects one of the four lines and replaces the tag and data contents of the line with the new line information. Before replacement, dirty lines are temporarily buffered and later copied back to memory after the new line has been read from memory. Figure 5-4 illustrates the three possible states for a cache line, with the possible processor-initiated transitions. Transitions are labeled with a capital letter, indicating the previous state, followed by a number indicating the specific case listed in Table 5-1.

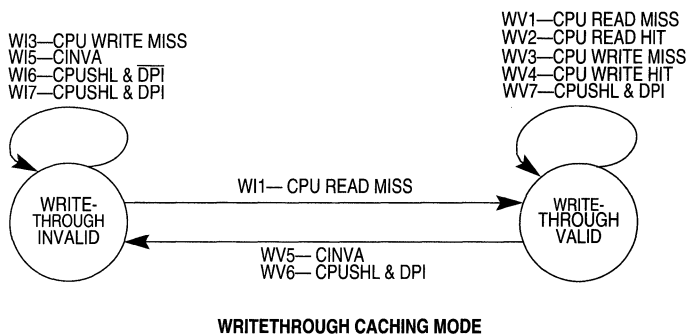
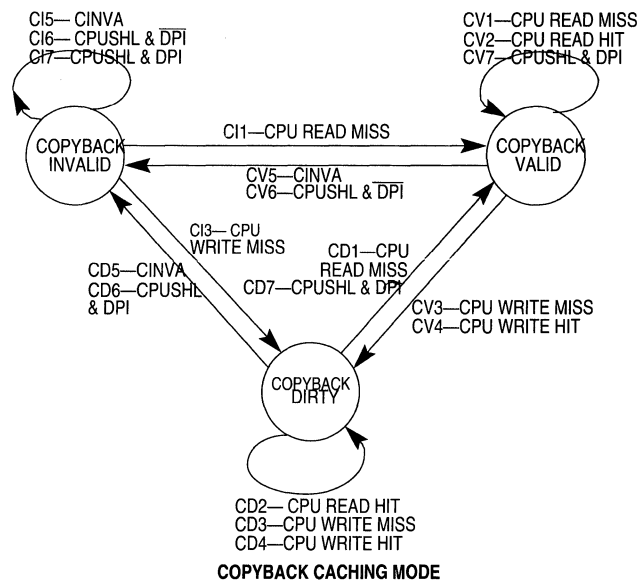


Figure 5-4. Cache Line State Diagrams

Table 5-1. Cache Line State Transitions

CACHE OPERATION	CURRENT STATE					
	INVALID CASES		VALID CASES		DIRTY CASES	
READ MISS	(C,W)1	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; Remain in current state.	CD1	Push dirty cache line to push 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)2	Not possible.	(C,W)V2	Supply data to processor; Remain in current state.	CD2	Supply data to processor; Remain in current state.

Table 5-1. Cache Line State Transitions (Continued)

CACHE OPERATION	CURRENT STATE					
	INVALID CASES		VALID CASES		DIRTY CASES	
WRITE MISS (COPYBACK MODE)	CI3	Read line from memory and update cache; Write data to cache; Go to dirty state.	CV3	Read new line from memory and update cache; Write data to cache; Go to dirty state.	CD3	Push dirty cache line to push buffer; Read new line from memory and update cache; Write push buffer contents to memory; Remain in current state.
WRITE MISS (WRITE-THROUGH MODE)	WI3	Write data to memory; Remain in current state.	WV3	Write data to memory; Remain in current state.	WD3	Write data to memory; Remain in current state.
WRITE HIT (COPYBACK MODE)	CI4	Not possible.	CV4	Write data to cache; Go to dirty state.	CD4	Write data to cache; Remain in current state.
WRITE HIT (WRITE-THROUGH MODE)	WI4	Not possible.	WV4	Write data to memory and to cache; Remain in current state.	WD4	Write data to memory and to cache; Go to valid state.
CACHE INVALIDATE	(C,W)5	No action; Remain in current state.	(C,W)V5	No action; Go to invalid state.	CD5	No action (dirty data lost); Go to invalid state.
CACHE PUSH	(C,W)6	No action; Remain in current state.	(C,W)V6	No action; Go to invalid state.	CD6	Push dirty cache line to memory; Go to invalid state.
CACHE PUSH	(C,W)7	No action; Remain in current state.	(C,W)V7	No action; Remain in current state.	CD7	Push dirty cache line to memory; Go to valid state.

Note

The shaded areas indicate that the cache mode has changed for the region corresponding to this cache line. In writethrough mode, a cache line should never be dirty.

To avoid these states:

1. Execute a CPUSHL instruction, or
2. Set the CINVA bit of the CACR (to invalidate the entire cache) before switching the cache mode.

5

SECTION 6 SRAM

6.1 SRAM FEATURES

- 4K (4096 Byte) SRAM, organized as 1024 x 32 Bits
- Single-Cycle Access
- Physically Located on Processor's High-Speed Local Bus
- Memory Location Programmable on any 32 KByte Address
- Memory Address Space Mappings Defined by User
- Byte, Word, Longword Address Capabilities
- Memory Mapping Defined by the Customer

6.2 SRAM OPERATION

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

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

6.3 SRAM PROGRAMMING MODEL

6.3.1 SRAM Base Address Register (RAMBAR)

The configuration information in the SRAM Base Address Register (RAMBAR) controls the operation of the SRAM module.

- The RAMBAR is the register that holds the base address of the SRAM. The RAMBAR is accessed as control register \$C04 using the privileged MOVEC instruction. The MOVEC instruction provides write-only access to this register.
- The RAMBAR register can be accessed from the Debug module in a similar manner. From the Debug module, the register can be read or written.
- All undefined bits in the register are reserved. These bits are ignored during writes to the RAMBAR, and return zeroes when read from the debug module.

- The RAMBAR valid bit is cleared by reset, disabling the SRAM module. All other bits are unaffected.

The RAMBAR register contains four control fields. These fields are detailed in the following subsections. The next illustration is that of the SRAM Base Address Register (RAMBAR).

SRAM Base Address Register (RAMBAR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BA31	BA30	BA29	BA28	BA27	BA26	BA25	BA24	BA23	BA22	BA21	BA20	BA19	BA18	BA17	BA16
RESET:	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA15	-	-	-	-	-	-	WP	-	-	C/I	SC	SD	UC	UD	V
RESET:	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

SRAM Base Address Register (RAMBAR)

BA[31:15] - Base Address

This field defines the 0-modulo-32K base address of the SRAM module. The SRAM memory occupies a 4KByte space defined by the contents of the Base Address field. By programming this field, the SRAM may be located on any 32KByte boundary within the processor's four gigabyte address space.

WP - Write Protect

This field allows only read accesses to the SRAM. When this bit is set, any attempted write access will generate an access error exception to the ColdFire processor core.

- 0 = Allow read and write accesses to the SRAM module
- 1 = Allow only read accesses to the SRAM module

C/I, SC, SD, UC, UD - Address Space Masks

This five bit field allows certain types of accesses to be "masked," or inhibited from accessing the SRAM module. The address space mask bits are:

- C/I = CPU Space/Interrupt acknowledge cycle mask
- SC = Supervisor Code address space mask
- SD = Supervisor Data address space mask
- UC = User code address space mask
- UD = User Data address space mask

For each address space bit:

- 0 = An access to the SRAM module can occur for this address space
- 1 = Disable this address space from the SRAM module. If a reference using this address space is made, it is inhibited from accessing the SRAM module, and is processed like any other non-SRAM reference.

These bits are useful for power management as detailed in Section 6.3.5.

V - Valid

The valid bit (V-bit) is specified by RAMBAR[0]. A hardware reset clears this bit. When set, this bit enables the SRAM module; otherwise, the module is disabled.

- 0 = Contents of RAMBAR are not valid
- 1 = Contents of RAMBAR are valid

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

```

if (RAMBAR[0] = 1)
  if (requested address[31:12] = {RAMBAR[31:15],3'b000})
    if (ASn of the requested type = 0)
      Access is mapped to the SRAM module
      if (access = read)
        Read the SRAM and return the data
      if (access = write)
        if (RAMBAR[8] = 0)
          Write the data into the SRAM
        else Signal a write-protect access error

```

where ASn refers to the five address space masks (C/I, SC, SD, UC, UD).

6.3.2 SRAM Initialization

After a hardware reset, the contents of the SRAM module are undefined. The valid bit of the RAMBAR is cleared, disabling the module. If the SRAM needs to be initialized with instructions or data, perform the following steps:

1. Load the RAMBAR mapping the SRAM module to the desired location within the address space.
2. Read the source data and write it to the SRAM. There are various instructions to support this function, including memory-to-memory move instructions, or the MOVEM opcode. The MOVEM instruction is optimized to generate line-sized burst fetches on 0-modulo-16 addresses, so this opcode generally provides maximum performance.

3. After the data has been loaded into the SRAM, it may be appropriate to load a revised value into the RAMBAR with a new set of “attributes.” These attributes consist of the write-protect and address space mask fields.

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

6.3.3 SRAM Initialization Code

The code segment below describes how to initialize the SRAM. The code sets the base address of the SRAM at \$20000000 and then initializes it to zeros.

```
RAMBASE      EQU    $20000000          ;set this variable to $20000000
move.l       #RAMBASE+1,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         ;if done, then exit; else continue looping
```

6.3.4 Power Management

As noted previously, depending on the configuration defined by the RAMBAR, instruction fetch and operand read accesses may be sent to the SRAM and unified cache simultaneously. If the access is mapped to the SRAM module, it sources the read data, and the unified cache access is discarded. If the SRAM is used only for data operands, asserting the ASn bits associated with instruction fetches can decrease power dissipation. Additionally, if the SRAM contains only instructions, masking operand accesses can reduce power dissipation.

Consider the examples in Table 6-1 of typical RAMBAR settings:

Table 6-1. Examples of Typical RAMBAR Settings

DATA CONTAINED IN SRAM	RAMBAR[7:0]
Code Only	\$2B
Data Only	\$35
Both Code And Data	\$21

SECTION 7

BUS OPERATION

This section describes the function of the bus, the signals that control the bus, and the bus cycles provided for data-transfer operations. Operation of the bus is defined for transfers initiated by the MCF5307 device as a bus master and for transfers initiated by an alternate bus master. This section includes descriptions of the error conditions, bus arbitration, and the reset operation.

7.1 FEATURES

The following list summarizes the key bus operation features:

- Up to 32 bits of address and data
- Access 8-, 16-, and 32-bit port sizes
- Generates byte, word, longword, and line size transfers
- Bus arbitration for external masters
- Burst and burst-inhibited transfer support
- Internal termination generation
- Termination generation for alternate masters
- Synchronous external signal termination supported

7.2 BUS AND CONTROL SIGNALS

This section defines the signals required by the MCF5307 bus. Although the timing of all of these signals is referenced to the BCLKO, it is not considered a bus signal. It is expected that the clock will be routed as needed to meet application requirements.

This section describes MCF5307 signals. Table 7-1 summarizes the signals. A brief description of the signals functionality follows.

The following signals are defined as the bus attribute signals: $\overline{R/W}$, TT, TM, and SIZ. The following signals are defined as the bus control signals: \overline{AS} , \overline{TA} , \overline{TS} , and \overline{TIP} .

Note: an overbar indicates an active-low signal.

Table 7-1.CF-Bus Signal Summary

SIGNAL NAME	DIRECTION	DESCRIPTION
A[31:0]	In/Out	Address Bus
D[31:0]	In/Out	Data Bus
R/W	In/Out	Read/Write
SIZ[1:0]	In/Out	Transfer Size
TT[1:0]	In/Out	Transfer Type
TM[2:0]	In/Out	Transfer Modifier
\overline{AS}	In/Out	Address Strobe
\overline{TS}	In/Out	Transfer Start
\overline{TIP}	In/Out	Transfer In Progress
\overline{TA}	In/Out	Transfer Acknowledge
$\overline{IRQ}[7,5,3,1]$	In	Interrupt Request

7

7.2.1 Address Bus

The address bus provides the address of the byte or most significant byte of the word or longword being transferred. The address lines also serve as the DRAM address pins. When an external master is using the DRAM controller or Chip-Select Module, the external master drives the address bus and asserts \overline{TS} or \overline{AS} to indicate a bus cycle is starting. During an interrupt-acknowledge access, the lower address lines, A[4:2], indicate the interrupt level being acknowledged.

7.2.1.1 ADDRESS BUS - (A[23:0]). The lower 24 bits of the address bus become valid when \overline{TS} is asserted. Bits A[4:2] indicate the interrupt level during interrupt-acknowledge cycles.

7.2.1.2 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 Pin Assignment Register (PAR) in the SIM determines the function of each of these eight multiplexed pins. You can program these on a bit-by-bit basis.

A[31:24]

When any of these pins are enabled in the PAR, they represent the most significant bits of the address bus. As much as 4 Gbytes of memory is available when all of these pins are programmed as address signals.

PP[15:8]

When any of these pins are enabled in the PAR, they represent the most significant bits of the parallel port.

7.2.2 Data Bus (D[31:0])

The data bus is a bidirectional, nonmultiplexed bus. Data gets latched on the bus on the rising clock edge. When interfacing with external memory or peripherals, the data bus port width, wait states, and internal termination are initially defined by D[7:0] during reset. You can program the port width for each chip-select and DRAM bank. The data bus uses the default configuration if none of the chip-selects or DRAM bank match the address decode. The default configuration is 32-bit port, with external termination, and burst inhibited. The data bus transfers byte, word, or longword-sized data. All 32 bits of the data bus are driven during writes, regardless of port width or operand size.

7.2.3 Read/Write - ($\overline{R/\overline{W}}$)

When the MCF5307 processor is the bus master, it drives the $\overline{R/\overline{W}}$ 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 alternate master access.

7.2.4 Size - (SIZ[1:0])

When it is the bus master, the MCF5307 processor outputs these signals to indicate the requested data transfer size. Table 7-2 shows the definition of the bus request size encodings. When the MCF5307 device is not the bus master, these signals function as inputs.

Table 7-2. Bus Cycle Size Encoding

SIZ[1:0]	PORT SIZE
00	Longword
01	Byte
10	Word
11	Line

7.2.5 Transfer Start - (\overline{TS})

The MCF5307 processor asserts this signal during the first clock cycle when address and attributes (TM, TT, TM, $\overline{TI\overline{P}}$, $\overline{R/\overline{W}}$, size) are valid address is driven on A[31:0] and is negated in the following clock cycle. When the MCF5307 device is not the bus master, \overline{TS} is an input.

7.2.6 Address Strobe - (\overline{AS})

Address Strobe is an active-low signal that indicates the start of a bus cycle and the address is stable. 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 device is not the bus master, \overline{AS} is an input.

7.2.7 Transfer Acknowledge - (\overline{TA})

When the MCF5307 device is master, the external system drives this input signal to terminate the bus transfer. The bus will continue to be driven until this synchronous signal

is asserted. For write cycles, the processor will continue 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. The \overline{TA} signal may be tied low if all bus cycles can support fast termination. Note that \overline{TA} **cannot be tied low if external masters are present**. The MCF5307 device will drive \overline{TA} if an external master requests it.

7.2.8 Transfer In Progress - ($\overline{TIP}/PP[7]$)

This multiplexed pin can serve as the transfer in progress output, or as one bit of the parallel port. Programming the Pin Assignment Register (PAR) in the SIM determines the function of this pin.

This output is asserted to indicate that a bus transfer is in progress and is negated during idle bus cycles if the bus is still granted to the processor. Note that \overline{TIP} is held asserted on back-to-back bus cycles.

7.2.9 Transfer Type - ($TT[1:0]/PP[1:0]$)

These multiplexed pins can serve as the transfer type outputs, or as two bits of the parallel port. Programming the Pin Assignment Register (PAR) in the SIM determines the function of each of these two multiplexed pins. The pins are programmable on a bit-by-bit basis.

When the MCF5307 is the bus master, it outputs these signals. They indicate the type of access for the current bus access. Table 7-3 shows the definition of the encodings.

Table 7-3. Bus Cycle Transfer Type Encoding

TT[1:0]	TRANSFER TYPE
0 0	Normal Access
0 1	Alternate Master Access
1 0	Emulator Access
1 1	CPU Space or Interrupt Acknowledge

7.2.10 Transfer Modifier - ($TM[2:0]/PP[4:2]$)

These multiplexed pins can serve as the transfer type outputs, or as three bits of the parallel port. Programming the Pin Assignment Register (PAR) in the SIM determines the function of each of these three multiplexed pins (which are programmable on a bit-by-bit basis).

These input signals provide supplemental information for each transfer type (see Table 7-4). For emulation transfers, the TM signals indicate user or data transfer types (see Table 7-6). For CPU space transfers, the TM signals are low (see Table 7-7). For interrupt acknowledge transfers, the TM signals carry the interrupt level being acknowledged (see Table 7-7). See the alternate master section for information on TM decoding.

7.2.11 Interrupt Request - ($\overline{IRQ7}, \overline{IRQ5}, \overline{IRQ3}, \overline{IRQ1}$)

You can use these four pins as predefined interrupt request signals (\overline{IRQx}). The \overline{IRQx} pins

are set to default interrupt request levels for users, corresponding to levels 1, 3, 5, and 7 bus control signals. Programming the Pin Assignment Register (PAR) in the SIM determines the function of this pin. In the Pin Assignment Register, you can redefine the interrupt levels for $\overline{IRQ5}$, $\overline{IRQ3}$, and $\overline{IRQ1}$ (on a bit-by-bit basis).

7.3 CLOCK AND RESET SIGNALS

These signals provide the external system interface for the MCF5307 device.

Table 7-4. CF-Bus Signal Summary

SIGNAL NAME	DIRECTION	DESCRIPTION
\overline{RSTI}	In	Reset In
CLKIN	In	Clock Input
BCLKO	Out	System Bus Clock Output
\overline{RSTO}	Out	Reset Out

7.3.1 Reset - (\overline{RSTI})

Asserting \overline{RSTI} will cause the MCF5307 processor to enter reset exception processing. When \overline{RSTI} is recognized, the address bus, data bus, TT, SIZ, $\overline{R/W}$, \overline{AS} and \overline{TS} will be three-stated; \overline{BR} and \overline{BD} will be negated.

7.3.2 Clock Input - (CLKIN)

CLKIN is the MCF5307 input clock frequency to the on-board phase-locked loop-clock generator. CLKIN is used to internally clock or sequence the MCF5307 internal bus interface and slave module logic.

7.3.3 System Bus Clock Output - (BCLKO)

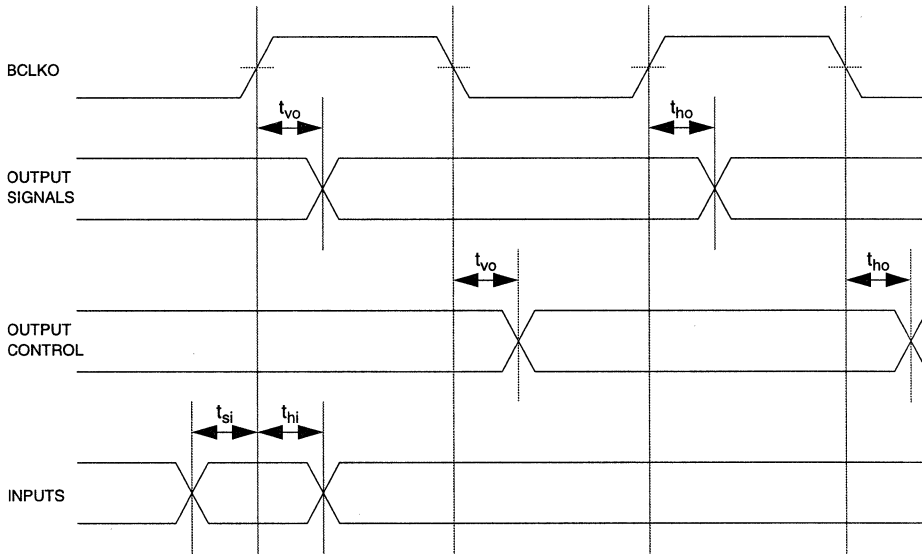
This output signal is generated by the internal PLL, and is the system bus clock output used as the bus timing reference by the external devices. BCLKO is a 1/2, 1/3, or 1/4 of the processor clock.

7.3.4 Reset Out - (\overline{RSTO})

Following the assertion of \overline{RSTI} , the PLL will temporarily lose its lock. During this time, the \overline{RSTO} signal will assert. When the PLL regains lock, the \overline{RSTO} signal will again negate. You can use this signal to reset external devices.

7.4 BUS CHARACTERISTICS

The MCF5307 processor uses a single clock signal (CLKIN) to generate its internal clock as well as the bus clock (BCLKO). Therefore, the external bus operates at the same speed as the internal clock rate, where all bus operations are synchronous to the rising edge of BCLKO, and the bus control signals (\overline{AS}) are synchronous to the falling edge of the BCLKO, which is visually represented in Figure 7-1.



t_{vo} = Propagation delay of signal relative to BCLKO edge

t_{ho} = Output hold time relative to BCLKO edge

t_{si} = Required input setup time relative to BCLKO edge

t_{hi} = Required input hold time relative to BCLKO edge

Figure 7-1. Signal Relationships to BCLKO

7.5 DATA TRANSFER OPERATIONS

The transfer of data between the MCF5307 processor and other devices involves the following signals:

1. Address bus (A[31:0])
2. Data bus (D31:0])
3. Control signals (\overline{AS})
4. Attribute signals (R/\overline{W} , TT, TM, \overline{TIP} , SIZ)

The MCF5307 bus supports byte, word, and longword operand transfers and allows accesses to 8-, 16-, and 32-bit data ports. With the MCF5307 device, you can select the port size of the specific memory, enable internal transfer termination, and set the number of wait states for the external slave being accessed by programming the Chip-Select Control Registers (CSCRs) and the DRAM Controller Control Registers (DCCRs). For more information on programming these registers, refer to the SIM, Chip-Select and DRAM Controller sections. Figure 7-2 illustrates the byte lanes that external chip-select memory and DRAM should be connected to.

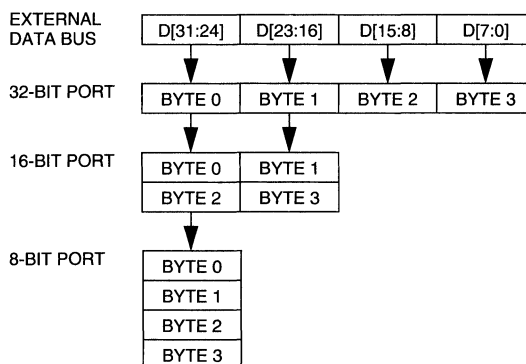


Figure 7-2. Connections for External Memory Port Sizes

The MCF5307 processor will compare the address and mask in chip-select 0 - 7 control registers, then the address and mask in DRAM bank 0 - 1 control registers.

- If the address and attributes do not match in a single chip-select register or DRAM control register, the MCF5307 will run an external bus cycle with external termination on a 32-bit port with burst-inhibited transfers.
- Should an address and attribute match in multiple chip-select registers, the matching chip-select signals will be driven; however, the MCF5307 device will run an external bus cycle with external termination on a 32-bit port with burst-inhibited transfers.
- Should an address and attribute match both DRAM controller registers or a DRAM control register and a chip-select, the operation is undefined.

Table 7-5 shows the type of access depending on what matches are made in the CS and DRAM control registers

Basic operation of the MCF5307 bus is a three-clock bus cycle. 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 set up and stable. Data and \overline{TA} are sampled during the second clock of a bus-read cycle. During a read, the external device provides data and is sampled at the rising edge at the end of the second bus clock. This data is concurrent with \overline{TA} , which is also sampled at the rising edge of the clock. During a write,

Table 7-5. Accesses by Matches in CS and DRAM Control Registers

NUMBER OF CHIP SELECTS REGISTER MATCHES	NUMBER OF DRAM CONTROLLER REGISTER MATCHES	TYPE OF ACCESS
None	None	External
Single	None	As defined by Chip -elect Control Register
Multiple	None	External
None	Single	As defined by DRAM Control Register
Single	Single	Undefined
Multiple	Single	Undefined
None	Multiple	Undefined
Single	Multiple	Undefined
Multiple	Multiple	Undefined

the ColdFire device drives data from the rising clock edge at the end of the first clock to the rising clock edge at the end of the bus cycle. You can add wait states between the first and second clocks by delaying the assertion of \overline{TA} . 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 7-4 and Figure 7-6 show the basic read and write operations.

7.5.1 Read Cycle

During a read cycle, the MCF5307 device receives data from memory or from a peripheral device. The read cycle flowchart is shown in Figure 7-3 while the read cycle timing diagram is shown in Figure 7-4.

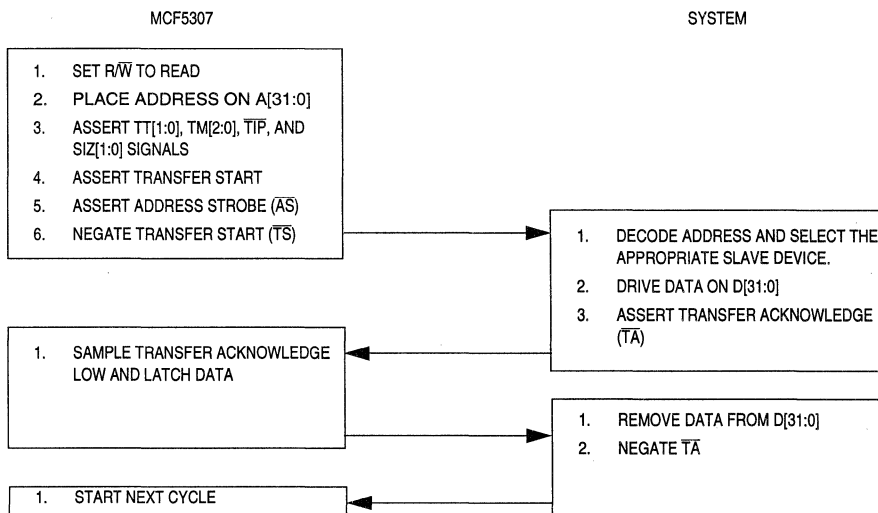


Figure 7-3. Read Cycle Flowchart

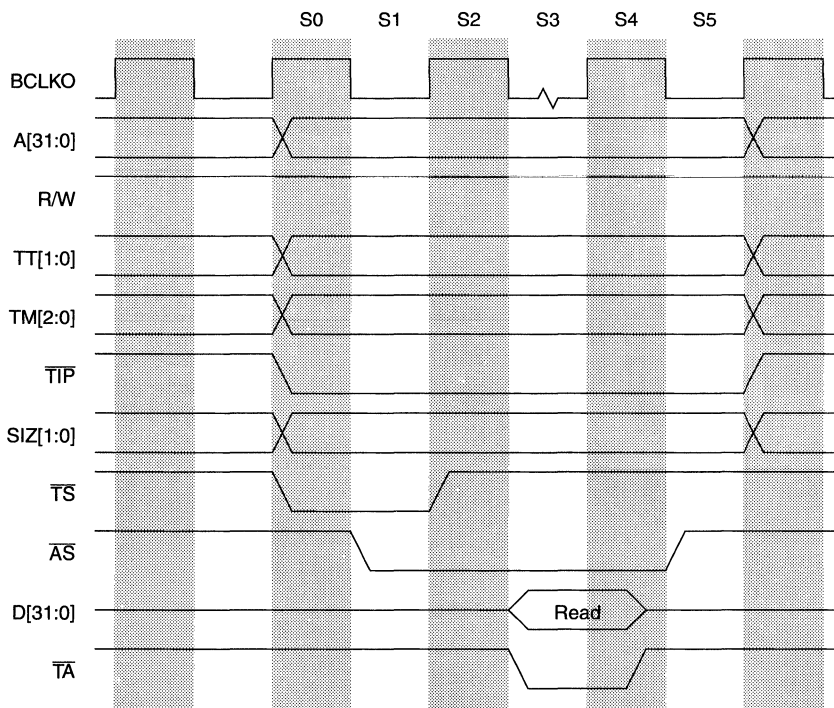


Figure 7-4. Basic Read Bus Cycle

A basic bus cycle has a minimum of six states (S0-S5). The various signals are asserted during specific states of a read cycle as follows:

STATE 0

The read cycle starts in state 0 (S0). The MCF5307 device places a valid address on the address bus, and asserts the R/\overline{W} , $TT[1:0]$, $TM[2:0]$, \overline{TIP} , $SIZ[1:0]$, and \overline{TS} signals.

STATE 1

\overline{AS} is asserted on the falling edge of state 1 (S1) to indicate that address and attributes are set up and stable.

STATE 2

\overline{TS} is negated on the rising edge of state 2 (S2).

STATE 3

During state 3 (S3), the MCF5307 device waits for a cycle termination signal (\overline{TA}). If \overline{TA} is not asserted before the rising edge at the end of S3, the MCF5307 device inserts wait

states (full clock cycles) until \overline{TA} is asserted. Data is provided by the external device and is sampled on the rising edge of S3 with \overline{TA} .

STATE 4

During state 4 (S4), the external device should remove the data from the data bus and negate \overline{TA} .

STATE 5

\overline{AS} is negated on the falling edge of state 5 (S5).

7.5.2 Write Cycle

During a write cycle, the MCF5307 device sends data to the memory or to a peripheral device.

The write cycle flowchart is shown in Figure 7-5 while the write cycle timing diagram is shown in Figure 7-6.

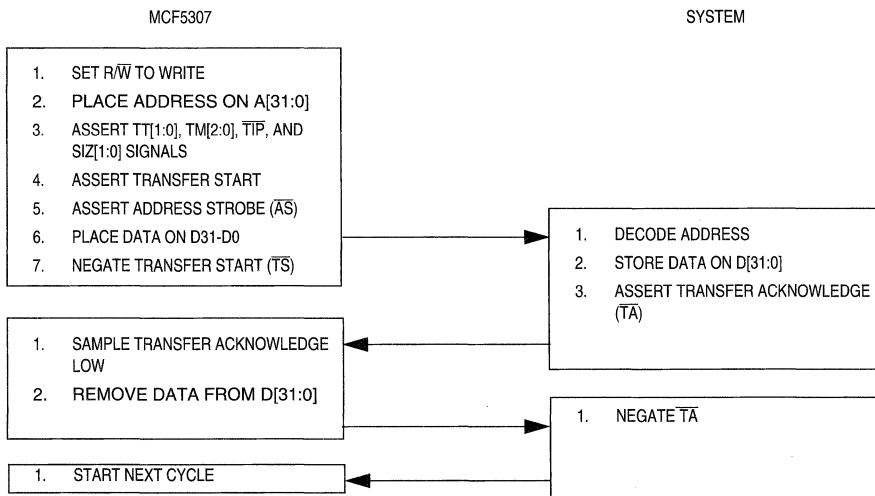


Figure 7-5. Write Cycle Flowchart

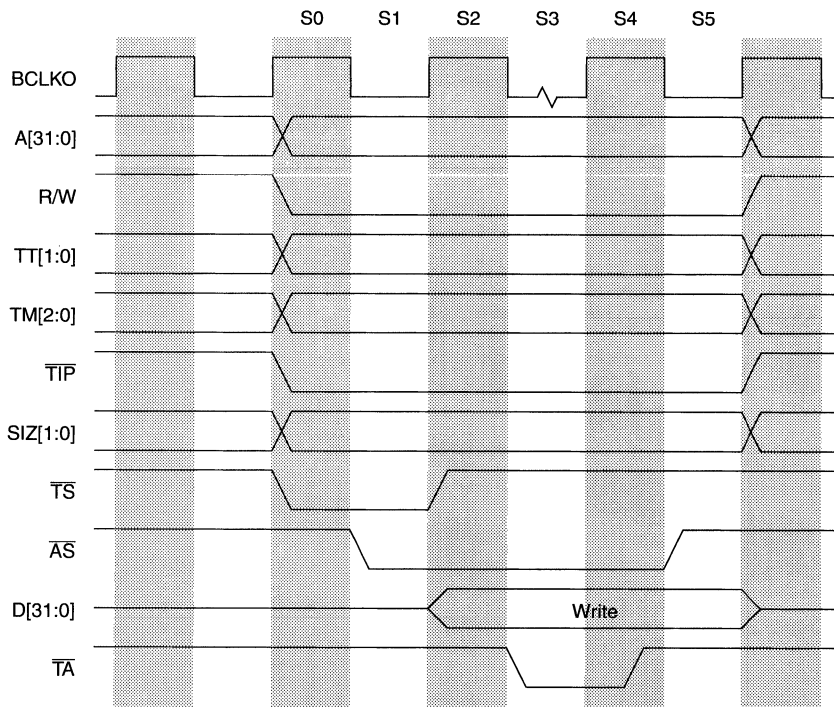


Figure 7-6. Basic Write Bus Cycle

The descriptions of the six states of a basic write cycle follows:

STATE 0

The read cycle starts in state 0 (S0). The MCF5307 device places a valid address on the address bus. R/\overline{W} is driven to a logic low. The MCF5307 device asserts the $TT[1:0]$, $TM[2:0]$, \overline{TIP} , $SIZ[1:0]$, and \overline{TS} signals.

STATE 1

\overline{AS} is asserted on the falling edge of state 1 (S1) to indicate that address and attributes are set up and stable.

STATE 2

\overline{TS} is negated on the rising edge of state 2 (S2). The data bus is driven out of the high impedance state as data is placed on the bus.

STATE 3

During state 3 (S3), the MCF5307 device waits for a cycle termination signal (\overline{TA}). If \overline{TA} is not asserted before the rising edge at the end of S3, the MCF5307 device inserts wait states (full clock cycles) until \overline{TA} is asserted. The external device provides the data, which is sampled on the rising edge of S3 with \overline{TA} .

STATE 4

During state 4 (S4), the external device should negate \overline{TA} .

STATE 5

\overline{AS} is negated on the falling edge of state 5 (S5).

7.5.3 Fast Termination Cycles

Two clock cycle transfers are supported on the MCF5307 bus. In most cases, this will be impractical to use in a system because the termination must take place in the same half clock that \overline{AS} is asserted. As this will be atypical, it is not referred to as the “zero-wait-state case,” but is called the “fast-termination case.” Figure 7-7 and Figure 7-8 show the fast termination bus cycles.

7

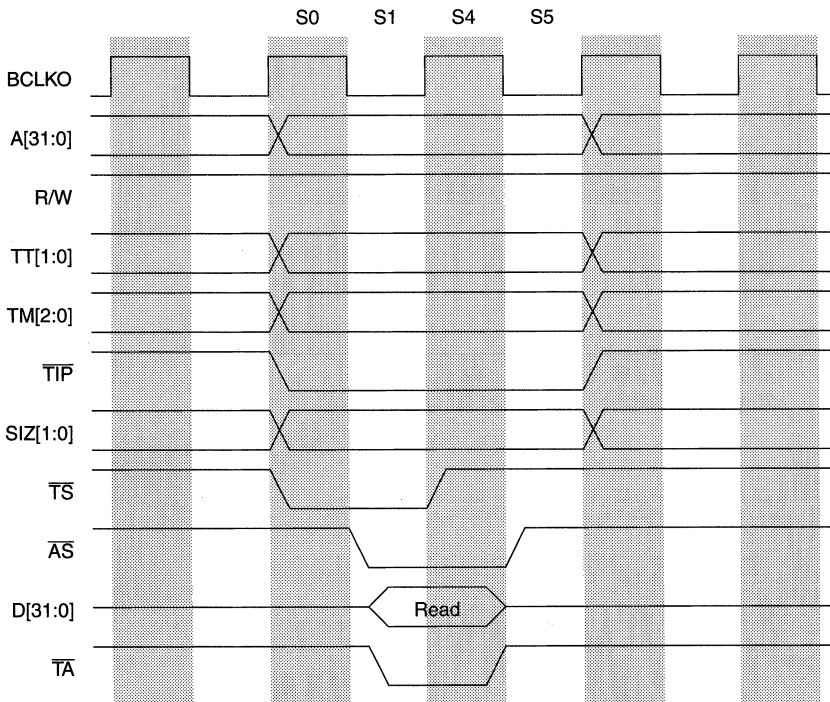


Figure 7-7. Read Cycle with Fast Termination

The descriptions of the four states of a fast termination read cycle follows:

STATE 0

The read cycle starts in state 0 (S0). The MCF5307 device places a valid address on the address bus, and asserts the $\overline{R/\overline{W}}$, $\overline{TT[1:0]}$, $\overline{TM[2:0]}$, \overline{TIP} , $\overline{SIZ[1:0]}$, and \overline{TS} signals.

STATE 1

\overline{AS} is asserted on the falling edge of state 1 (S1) to indicate that address and attributes are set up and stable. To get fast termination, \overline{TA} must be asserted during S1. Data is provided by the external device and is sampled on the rising edge of S1 with \overline{TA} .

STATE 4

During state 4 (S4), the external device should remove the data from the data bus and negate \overline{TA} . \overline{TS} is negated on the rising edge of state 4 (S4).

STATE 5

\overline{AS} is negated on the falling edge of state 5 (S5).

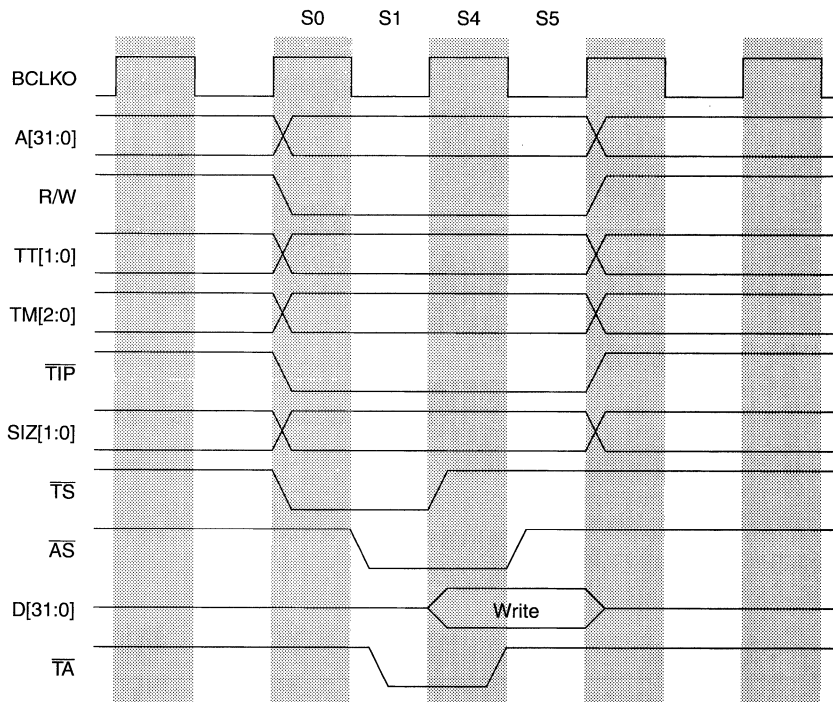


Figure 7-8. Write Cycle with Fast Termination

The descriptions of the four states of a fast termination write cycle follows:

The descriptions of the four states of a fast termination write cycle follow:

STATE 0

The read cycle starts in state 0 (S0). The MCF5307 device places a valid address on the address bus. R/\overline{W} is driven to a logic low. The MCF5307 device asserts the $TT[1:0]$, $TM[2:0]$, \overline{TIP} , $SIZ[1:0]$, and \overline{TS} signals.

STATE 1

\overline{AS} is asserted on the falling edge of state 1 (S1) to indicate that address and attributes are set up and stable.

STATE 4

During state 4 (S4), the external device should negate \overline{TA} . \overline{TS} is negated on the rising edge of state 4 (S4). The data bus is driven out of the high-impedance state as data is placed on the bus. During state 4 (S4), the MCF5307 device waits for a cycle termination signal (\overline{TA}).

STATE 5

\overline{AS} is negated on the falling edge of state 5 (S5).

7.5.4 Back-to-Back Bus Cycles

The MCF5307 device allows back-to-back bus cycles on the bus. The MCF5307 device will run back-to-back bus cycles whenever possible. For example, when a longword read is started on a word-size bus, the processor will perform two word reads back to back. The distinguishing feature of this type of cycle is the \overline{TIP} signal being asserted continuously. Figure 7-9 shows two bus cycles that occur back to back.

For illustration purposes, a read and a write cycle are used. There are no restrictions on cycles being either reads or writes for them to be back to back. You cannot tell the processor when or when not to initiate a back-to-back cycle.

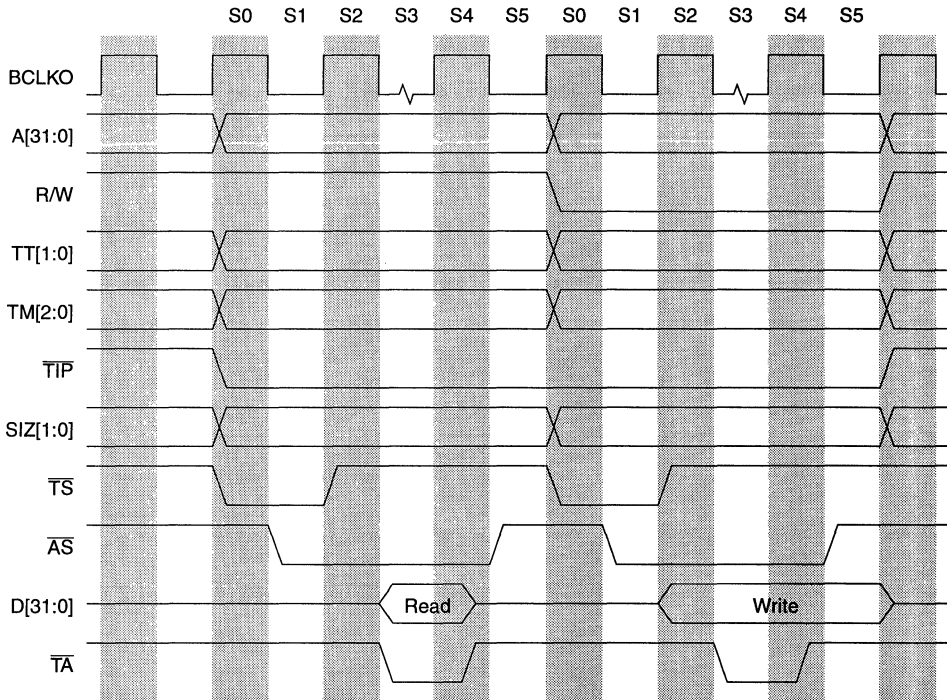


Figure 7-9. Back-to-Back Bus Cycles

7.5.5 Line Bus Cycles

The MCF5307 bus can support 2-1-1-1 burst cycles to maximize cache performance and optimize DMA transfers. You can add wait states by delaying termination of the cycle. The initiation of a burst cycle is encoded on the size pins.

In the MCF5307 device, any region of the memory space can be declared “burst inhibited” by clearing the Chip-Select Burst Read-Enable and Burst Write-Enable bits for that region. If a line access is initiated to a region that is burst inhibited, the first access on the MF5307 bus will be encoded as a line access, but the following accesses will be separated by \overline{AS} negations. The transfer size encoded on SIZ[1:0] will not exceed the programmed port size.

A line is defined as a 16-byte value, aligned in memory on 16-byte boundaries. Although the line itself is aligned on 16-byte boundaries, the line access does not necessarily begin on the aligned address. Therefore, the bus interface supports line transfers on multiple

address boundaries. The allowable patterns during a line accesses are shown in Table 7-6.

Table 7-6. Allowable Line Access Patterns

ADDR[3:2]	LONGWORD ACCESSES
00	0 - 4 - 8 - C
01	4 - 8 - C - 0
10	8 - C - 0 - 4
11	C - 0 - 4 - 8

7.5.5.1 LINE READ BUS CYCLES. Figure 7-10 shows a line access read with zero wait states. Figure 7-11 shows a line access read with one wait state. Figure 7-12 shows a line read access that is burst inhibited with fast termination.

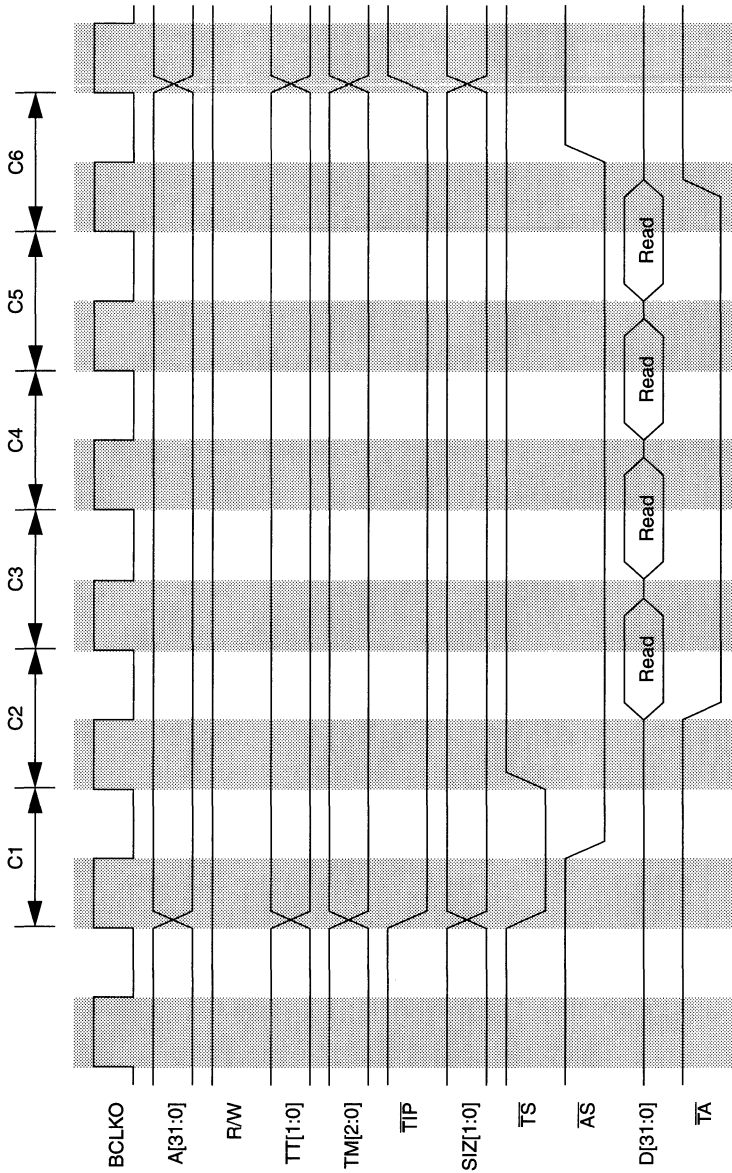


Figure 7-10. Line Read Burst (2-1-1-1)

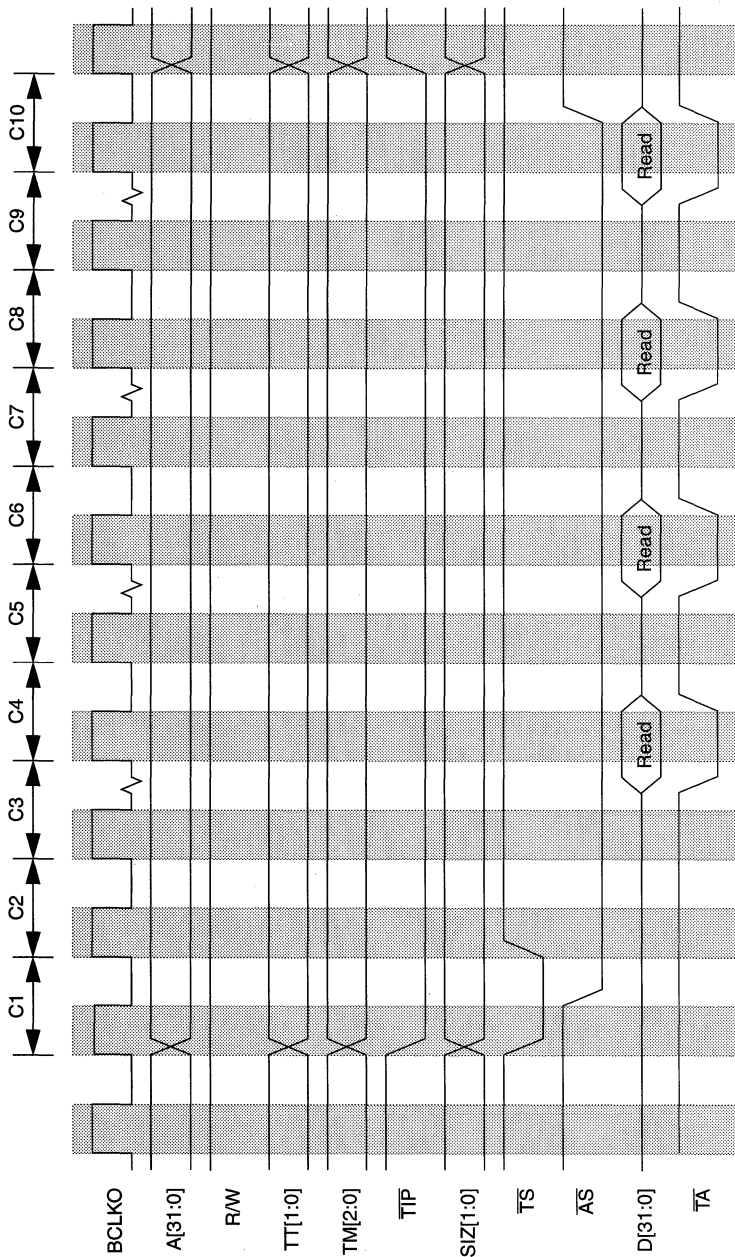


Figure 7-11. Line Read Burst (3-2-2-2)

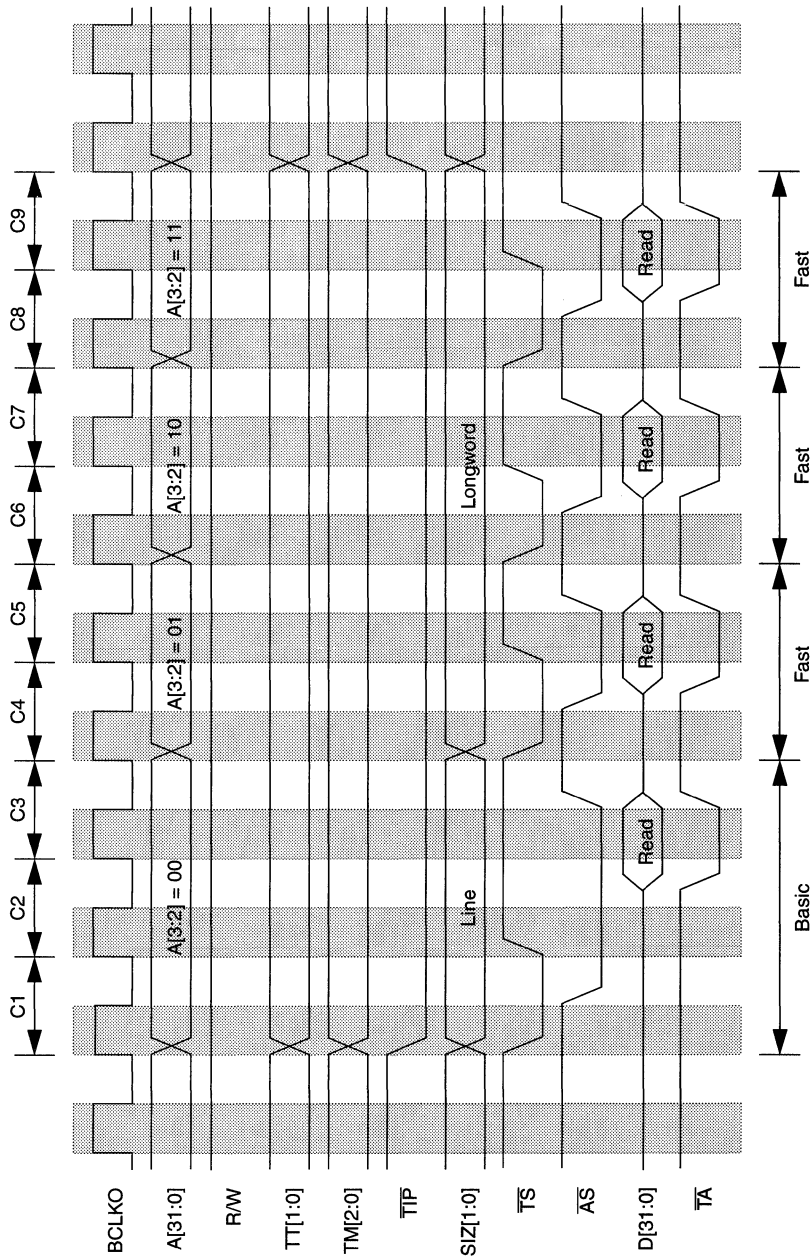


Figure 7-12. Line Read Burst-Inhibited with Fast Termination

7.5.5.2 LINE WRITE BUS CYCLES. Figure 7-13 and Figure 7-14 show burst line access writes. Figure 7-15 shows a burst-inhibited line write.

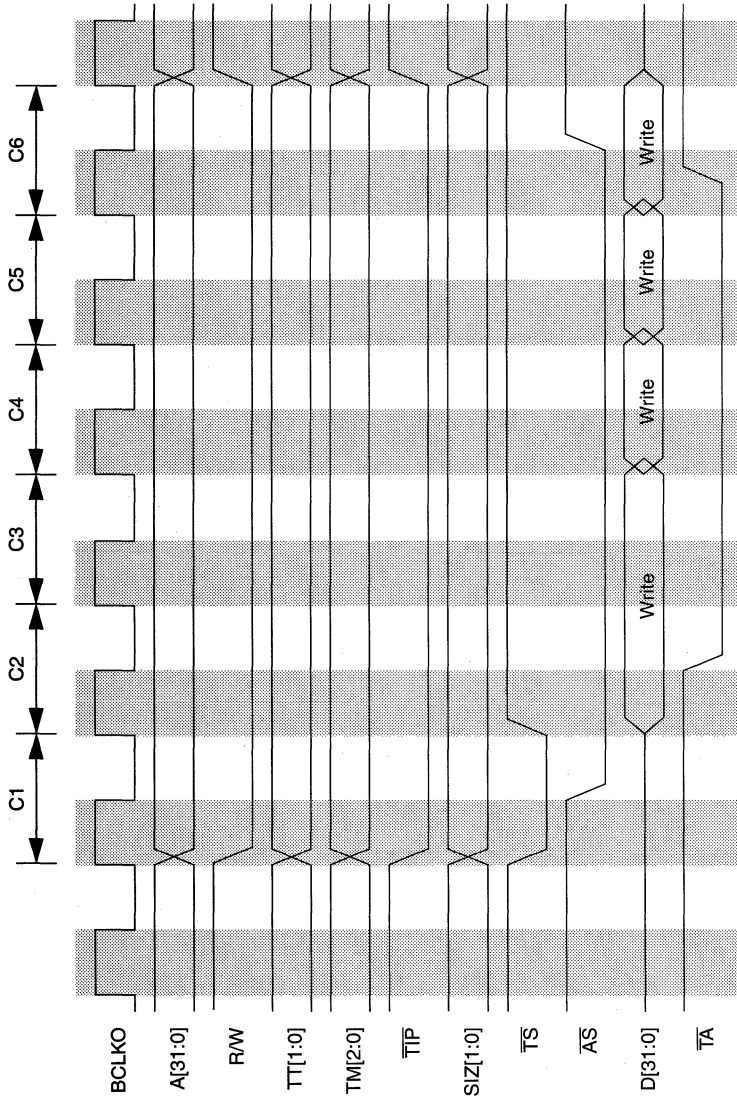


Figure 7-13. Line Write Burst (2-1-1-1)

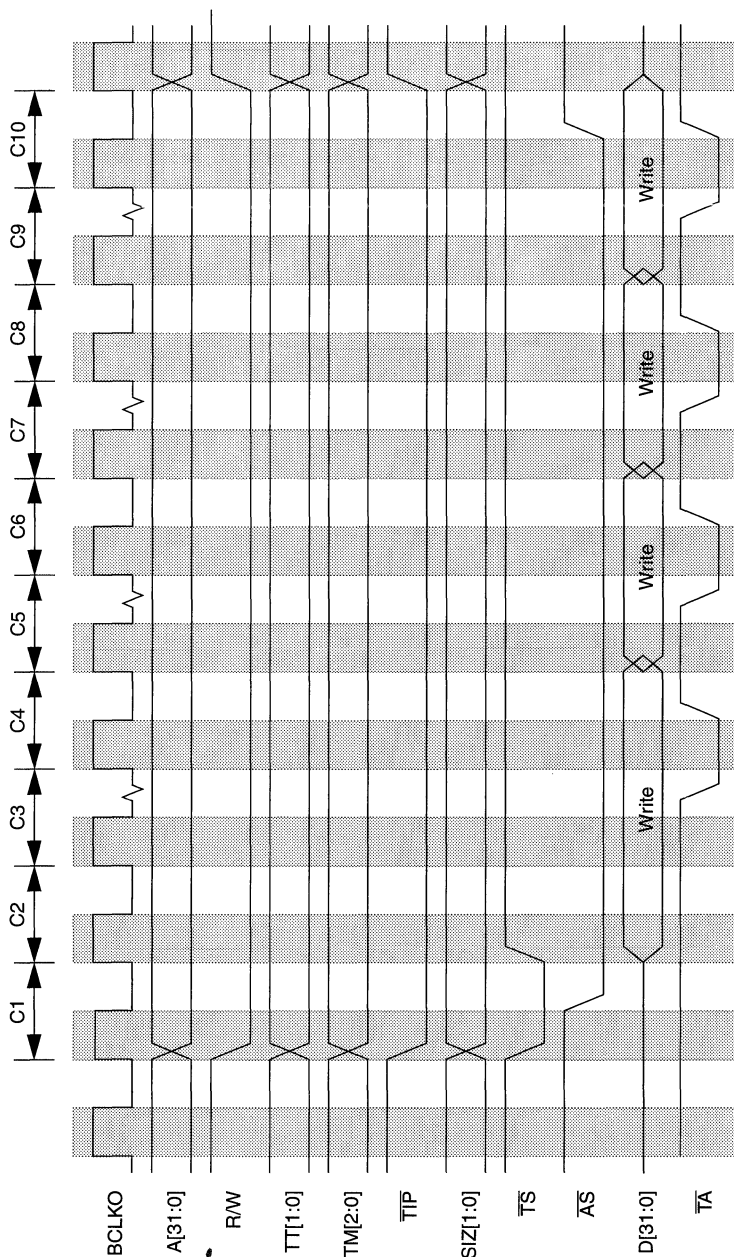


Figure 7-14. Line Write Burst (4-2-2-2) with One Wait State

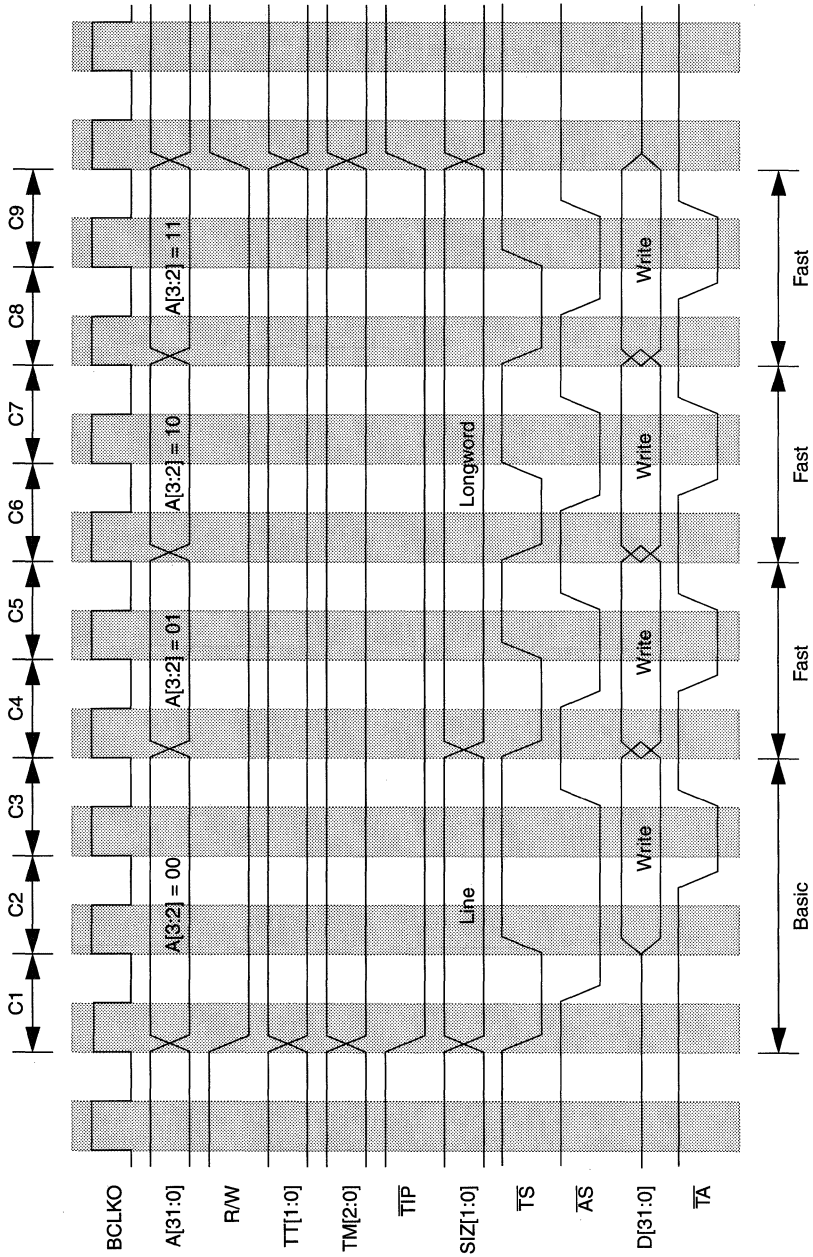


Figure 7-15. Line Write Burst-Inhibited

7.6 INTERRUPT EXCEPTIONS

When a peripheral device needs the MCF5307 device or is ready to send information that the ColdFire core requires, it can signal the ColdFire core to take an interrupt exception. The interrupt exception transfers control to a routine that responds appropriately. The peripheral device uses the interrupt-request signals ($\overline{\text{IRQx}}$) to signal an interrupt condition to the MCF5307 device.

The MCF5307 device has two levels of interrupt masking: System Integration Module (SIM) interrupt controller, and the Status Register three-bit interrupt priority mask.

The first level of interrupt masking is in the interrupt controller in the System Integration Module (SIM). The Interrupt Mask Registers in the SIM compare interrupt inputs with interrupt mask levels that you program. The SIM outputs only those interrupts that have not been masked.

The Status Register (SR) provides the second level of interrupt masking in the ColdFire core through the three-bit interrupt priority mask. The interrupt priority mask bits contain the highest priority level that the ColdFire core ignores. When an interrupt request has a priority higher than the value in the mask, the ColdFire core makes the request a pending interrupt. For more information about the Status Register refer to **Status Register in Section 3 ColdFire Core**.

The MCF5307 device continuously samples the external interrupt input signals and synchronizes these signals. An interrupt request must be held constant for at least two consecutive BCLKO periods to be considered a valid input. Also, the interrupt request must maintain the interrupt request level until the MCF5307 device acknowledges the interrupt with an interrupt-acknowledge cycle to guarantee that the interrupt is recognized.

NOTE

Interrupt levels 1 through 6 are level-sensitive only. Interrupt level 7 is both level sensitive and edge triggered as described in **7.6.1 Level 7 Interrupts**.

The MCF5307 device takes an interrupt exception for a pending interrupt within one instruction boundary after processing any other pending exception with a higher priority. Thus, the MCF5307 device executes at least one instruction in an interrupt exception handler *before* recognizing another interrupt request.

If autovector generation is used for internal interrupts (the AVEC bit in the appropriate interrupt control register in the SIM is 1), the interrupt acknowledge vector will be generated internally and no interrupt acknowledge cycle will be generated on the external bus. Refer to **Section 8 SIM** for ICR programming.

If autovector generation is used for external interrupts (the appropriate AVEC bit in the auto vector control register in the SIM is 1), no interrupt acknowledge cycle will be shown on the external bus ($\overline{\text{AS}}$ will not be asserted) unless the BLK bit in the auto vector control

register is 0. Consequently, you must clear the external interrupt in the interrupt service routine. Refer to **Section 8 SIM** for more information on the auto vector control register.

7.6.1 Level 7 Interrupts

Level 7 interrupts are handled differently than interrupt levels 1 through 6. A level 7 interrupt is a nonmaskable interrupt; therefore, a 7 in the interrupt mask does not disable a level 7 interrupt.

Level 7 interrupts are edge triggered by a transition from a lower priority request to the level 7 request, as opposed to interrupt levels 1 through 6, which are level sensitive. Therefore, if $\overline{IRQ7}$ remains asserted, the MCF5307 device will only recognize one level 7 interrupt because only one transition from a lower level request to a level 7 request occurred. For the processor to recognize a level 7 interrupt followed by another level 7 interrupt, one of the two following sequences must occur:

1. The interrupt request level on the interrupt control pins changes from a lower request level to level 7 and remains at level 7 until the interrupt-acknowledge bus cycle begins. Later, the interrupt request level returns to a lower interrupt request level and then back to level 7, causing a second transition on the interrupt control lines.
2. The interrupt request level on the interrupt control pins changes from a lower request level to level 7 and remains at level 7. If the interrupt handling routine for the level 7 interrupt lowers the interrupt mask level, a second level 7 interrupt will be recognized even though no transition has occurred on the interrupt control pins. After the level 7 interrupt handling routine completes, the MCF5307 device will compare the interrupt mask level to the interrupt request level on the \overline{IRQx} signals. Because the interrupt mask level will be lower than the requested level, the interrupt mask will be set back to level 7. The level 7 request on $\overline{IRQ7}$ must be held until the second interrupt-acknowledge bus cycle has begun to ensure that the interrupt is recognized.

For more information on interrupts in general, see the application note *A Discussion of Interrupts for the MC68000* (AN1012).

7.6.2 Interrupt-Acknowledge Cycle

When the MCF5307 device 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. It differs from a normal read cycle in the following respects:

- $TT[1:0] = \$3$ to indicate a CPU space or acknowledge bus cycle
- $TM[2:0]$ = the level of interrupt being acknowledged
- Address signals $A[31:5]$ are set to all ones ($\$7FFFFFFF$)

- Address signals A[4:2] are set to the interrupt request level being acknowledged
- Address signals A[1:0] are set to all zeros (\$0)

The responding device places the vector number on D[31:24] of the data bus during the interrupt-acknowledge bus cycle and the cycle is terminated normally with \overline{TA} . Figure 7-16 illustrates an interrupt-acknowledge cycle terminated with \overline{TA} .

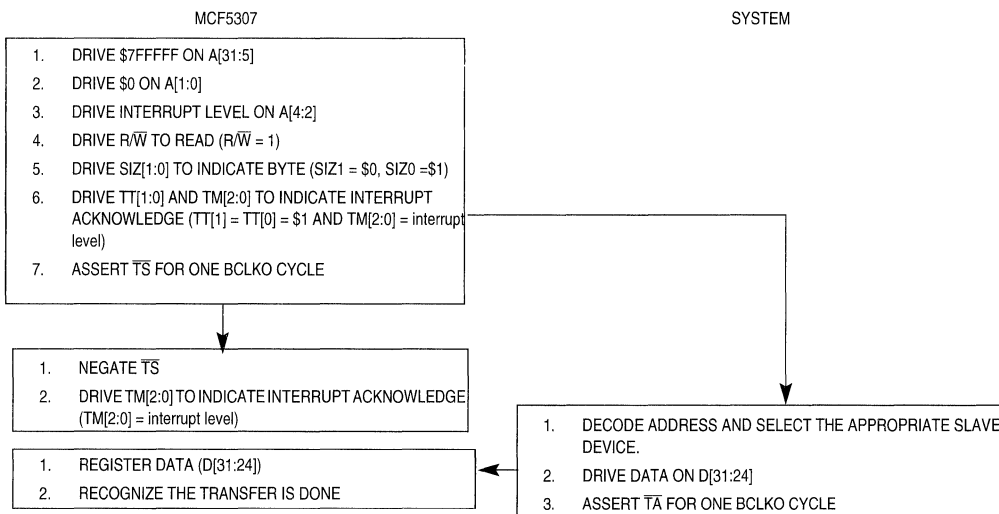


Figure 7-16. Interrupt-Acknowledge Cycle Flowchart

7.7 BUS ARBITRATION

The MCF5307 bus protocol provides for one bus master at a time: either the MCF5307 or an external device. If more than one external bus master is connected to the bus, an external arbiter can prioritize requests and determine which device is granted bus access. Bus arbitration is the protocol by which the MCF5307 device or an external device becomes the bus master. When the MCF5307 device is the bus master, it uses the bus to read instructions and transfer data not contained in its internal cache or memory to and from external memory. When an alternate bus master owns the bus, the MCF5307 device can monitor the alternate bus master's transfers and assert chip-select and DRAM control, and transfer termination signals. This capability is discussed in more detail in the Chip-Select Control Register subsection in **Section 9 Chip-Selects**, and DRAM subsection in **Section 11 DRAM**.

The MCF5307 bus arbitration can be used in two modes. A two-wire mode is provided for systems where the MCF5307 device and a single external bus master are the only two masters arbitrating for use of the external bus. This arbitration mode uses the bus grant (BG) and bus driven (BD) signals. The external bus master can ignore the bus request (BR) signal.

The second mode is provided for systems where multiple external bus masters are arbitrating for use of the external bus. This arbitration mode requires an external bus arbiter and uses the \overline{BG} , \overline{BD} , and \overline{BR} signals to control usage of the external bus.

In either arbitration mode, the bus arbitration unit in the MCF5307 device operates synchronously and transitions between states on the rising edge of CLK.

For systems where the MCF5307 device is the only possible bus master, the bus can be continuously granted to the MCF5307 processor by tying the \overline{BG} signal to GND. An arbiter is not required.

7.7.1 Bus Arbitration Signals

This subsection defines the signals required for bus arbitration. Although the timing of all of these signals is referenced to the system clock, the system clock is not considered a bus signal. It is expected that the clock will be routed as needed to meet application requirements. See Table 7-7 .

Table 7-7: ColdFire Bus Signal Summary

SIGNAL NAME	DIRECTION	DESCRIPTION
\overline{BR}	Output	Bus Request
\overline{BG}	Input	Bus Grant
\overline{BD}	Output	Bus Driven

7.7.2 Bus Request - (\overline{BR})

This output signal indicates to an external arbiter that the processor needs to become bus master for one or more bus cycles. \overline{BR} is negated when the MCF5307 processor begins an access to the external bus with no other internal accesses pending, and \overline{BR} remains negated until another internal request occurs.

7.7.3 Bus Grant - (\overline{BG})

An external arbiter asserts this input signal to indicate that the MCF5307 device can control the bus at the next rising edge of BCLKO. When the arbiter negates \overline{BG} , the MCF5307 processor must relinquish the bus as soon as the current transfer is complete. The external arbiter must not grant the bus to any other master until both \overline{BD} and \overline{BG} are negated.

7.7.4 Bus Driven - (\overline{BD})

The MCF5307 device asserts this output signal to indicate that it is the current master when it is driving the bus. If the MCF5307 device is the bus master but is not using the bus, the \overline{BD} signal is not asserted. If the MCF5307 device loses bus mastership during a bus transfer, it will complete the last transfer of the current access, negate \overline{BD} , and three-state all bus signals on the rising edge of BCLKO. If the MCF5307 device loses bus

mastership during an idle clock cycle, it will three-state all bus signals on the rising edge of BCLKO.

7.7.5 Two-Master Bus Arbitration Protocol (Two-Wire Mode)

The two-wire mode of bus arbitration allows the MCF5307 device to share the external bus with a single external bus master without requiring an external bus arbiter. Figure 7-17 shows the MCF5307 device connecting to an external bus master using the two-wire mode. In this mode, the active-low \overline{BG} input of the MCF5307 device is connected to the active-high HOLDREQ output of the external bus master and the active-low bus-driven (\overline{BD}) output of the MCF5307 device is connected to the active-high HOLDACK input of the external bus master. Because the external bus master controls the assertion/negation of HOLDREQ, it controls when the MCF5307 device is granted the bus, making the MCF5307 device the lower priority master.

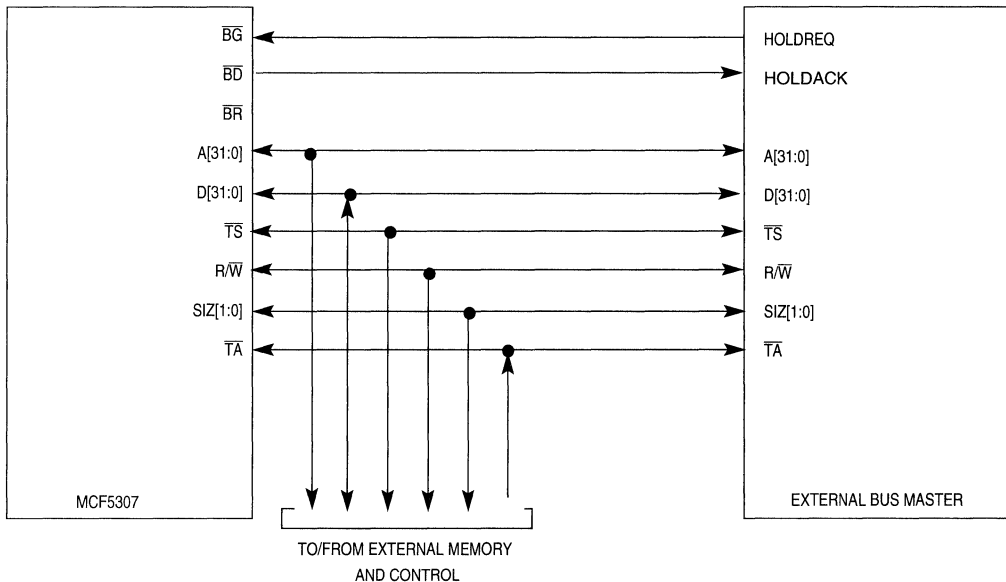


Figure 7-17. MCF5307 Two-Wire Mode Bus Arbitration Interface

When the alternate master is not using the bus, it negates HOLDREQ driving \overline{BG} low, granting the bus to the MCF5307 device. When the MCF5307 processor has an internal bus request pending and \overline{BG} is low, the MCF5307 device will drive \overline{BD} low, negating HOLDACK to the external bus master. When the external bus master needs the external bus, it asserts HOLDREQ, driving \overline{BG} high, requesting the MCF5307 device to relinquish the bus. If \overline{BG} is negated while a bus cycle is in progress and if the external to master control bit is cleared, the MCF5307 processor will relinquish the bus at the completion of the bus cycle. Note that the MCF5307 device considers the individual transfers of a burst

or burst-inhibited access to be a single bus cycle and does not relinquish the bus until the completion of the last transfer of the series.

When the bus has been granted to the MCF5307 device, one of two situations can occur. In the first case, if the MCF5307 processor has an internal bus request pending, the MCF5307 device asserts \overline{BD} to indicate explicit bus ownership and begins the pending bus cycle by asserting \overline{TS} . The MCF5307 device 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 device will negate \overline{BD} . As long as \overline{BG} is asserted, \overline{BD} remains asserted to indicate the bus is owned by the MCF5307 device and it continuously drives the address bus, attributes, and control signals.

In the second situation, the bus is granted to the MCF5307 device, but it does not have an internal bus request pending, so it takes implicit ownership of the bus. The MCF5307 device does not drive the bus and does not assert bus driven \overline{BD} if the bus is implicitly owned. If an internal bus request is generated, the MCF5307 processor assumes explicit ownership of the bus. If explicit ownership was assumed because of an internal request being generated, the MCF5307 part immediately begins an access and asserts bus driven \overline{BD} .

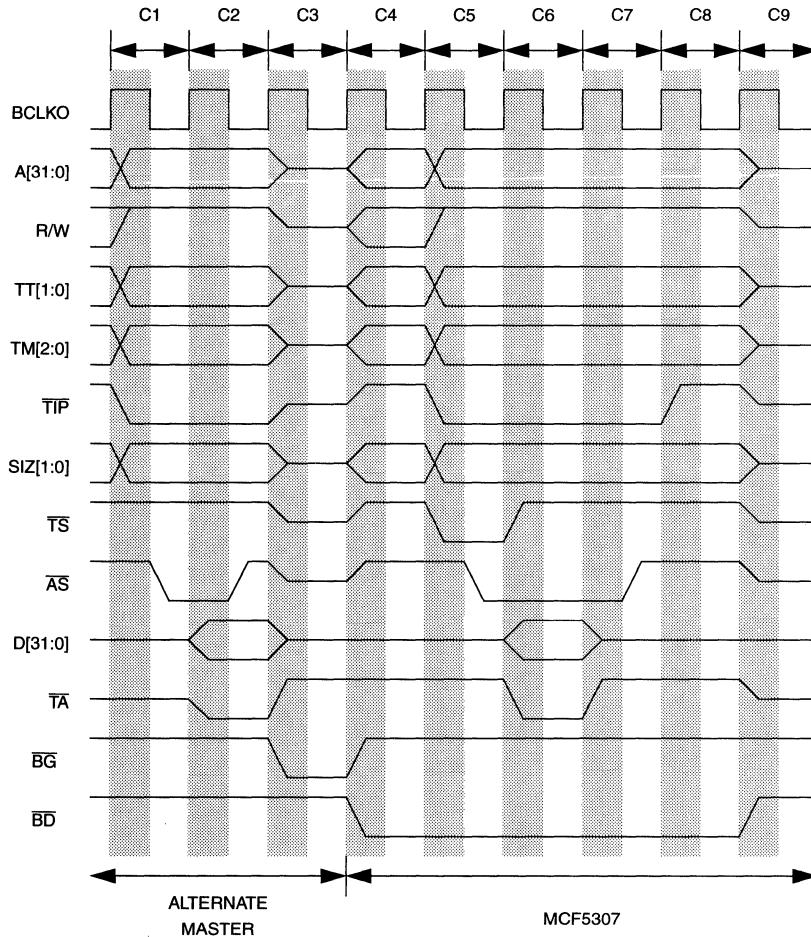


Figure 7-18. Two-Wire Bus Arbitration with Bus Request Asserted

In Figure 7-18, the alternate master is owner of the external bus during C1 and C2. During C3 the alternate master relinquishes control of the bus by asserting \overline{BG} to the MCF5307 device. At this point, there is an internal access pending so the MCF5307 device asserts \overline{BD} during C4 and begins the access. Thus, the MCF5307 device becomes the explicit master of the external bus. Also during C4, the alternate master removes the grant from the MCF5307 device by negating \overline{BG} . Because the MCF5307 processor is the current bus master, the MCF5307 device continues to assert \overline{BD} until the current transfer has completed. Because \overline{BG} is negated, the MCF5307 processor negates \overline{BD} during C9 and

three-states the external bus, thereby returning external bus ownership to the alternate master.

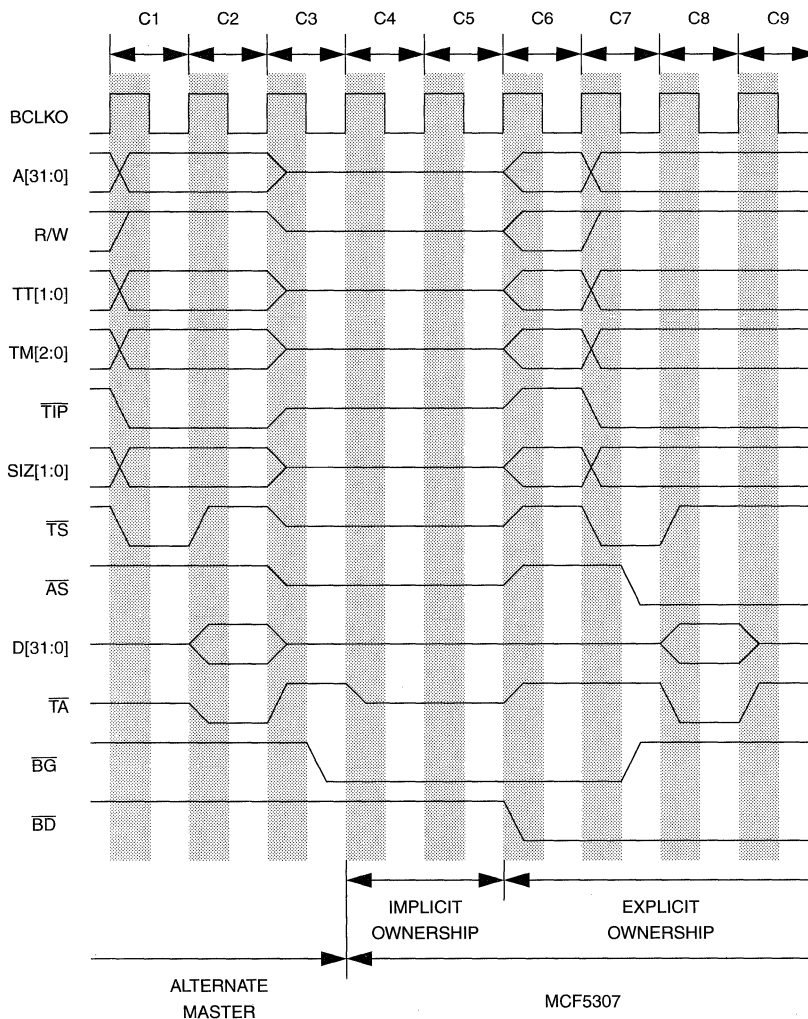


Figure 7-19. Two-Wire Implicit and Explicit Bus Ownership

In Figure 7-20, the alternate master has ownership of the external bus during Clock 1 (C1) and Clock 2 (C2). In Clock 3 (C3) the alternate master releases control of the bus by asserting \overline{BG} to the MCF5307 device. During Clock 4 (C4) and Clock 5 (C5), the MCF5307 processor is implicit owner because an internal access is not pending. In C5, an internal bus request becomes pending, causing the MCF5307 processor to take explicit ownership of the bus in Clock 6 (C6) by asserting \overline{BD} . In Clock 7 (C7), the alternate master

master removes the bus grant to the MCF5307 device. The MCF5307 processor does not relinquish the bus (the MCF5307 device continues to assert \overline{BD}) until the end of the transfer.

NOTE

The MCF5307 device can start a transfer during the CLK cycle after \overline{BG} is asserted. The alternate master should not assert \overline{BG} to the MCF5307 device until it has stopped driving the bus. \overline{BG} cannot be asserted while the alternate master transfer is still in progress or damage to the part could occur.

Figure 7-20 is a bus arbitration state diagram for the MCF5307 bus arbitration protocol. Table 7-8 lists the conditions that cause bus arbitration state changes. Table 7-9 describes the MCF5307 bus ownership, bus driving and assertion of bus driven (\overline{BD}) for each state of the bus arbitration state machine.

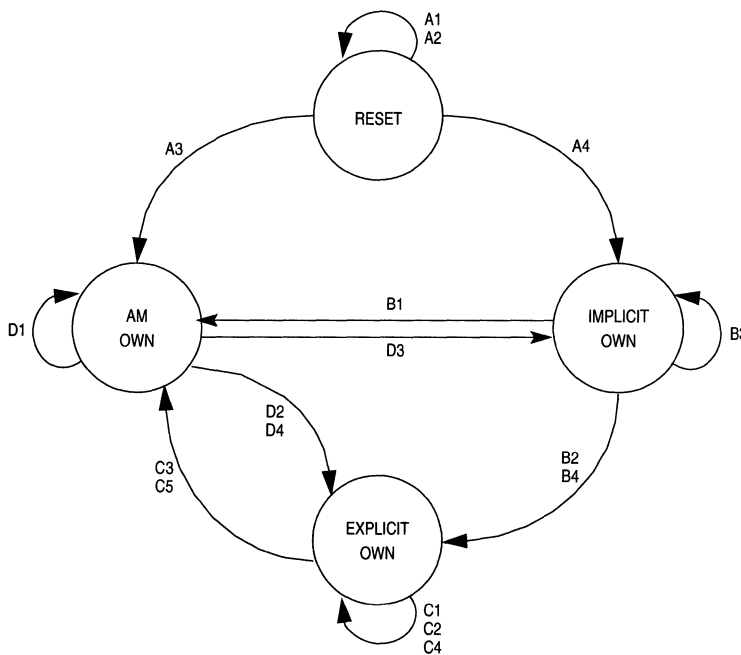


Figure 7-20. MCF5307 Two-Wire Bus Arbitration Protocol State Diagram

Table 7-8. MCF5307 Two-Wire Bus Arbitration Protocol Transition Conditions

PRESENT STATE	CONDITION LABEL	RSTI	SOFTWARE WATCHDOG RESET	BG	INTERNAL BUS REQUEST	TRANSFER IN PROGRESS	END OF CYCLE	NEXT STATE
RESET	A1	A	-	-	-	-	-	Reset
	A2	N	A	-	-	-	-	Reset
	A3	N	N	N	-	-	-	Am Own
	A4	N	N	A	-	-	-	Implicit Own
IMPLICIT OWN	B1	N	N	N	-	-	-	Am Own
	B2	N	N	A	-	-	-	Explicit Own
	B3	N	N	A	N	-	-	Implicit Own
	B4	N	N	A	A	-	-	Explicit Own
EXPLICIT OWN	C1	N	N	A	-	-	-	Explicit Own
	C2	N	N	N	-	-	-	Explicit Own
	C3	N	N	N	-	N	-	Am Own
	C4	N	N	N	-	A	N	Explicit Own
	C5	N	N	N	-	A	A	Am Own
AM OWN	D1	N	N	N	-	-	-	Am Own
	D2	N	N	A	-	-	-	Explicit Own
	D3	N	N	A	N	-	-	Implicit Own
	D4	N	N	A	A	-	-	Explicit Own

NOTES

1)“N” means negated; “A” means asserted; “AM” means alternate master.2)End of Cycle: Whatever terminates a bus transaction whether it is normal or bus error. Note that bus cycles that result from a burst inhibited transfer are considered part of that original transfer.

Table 7-9. MCF5307 Two-Wire Arbitration Protocol State Diagram

STATE	OWN	BUS STATUS	BD
Reset	No	Not Driven	Negated
Implicit Own	Yes	Not Driven	Negated
Explicit Own	Yes	Driven	Asserted
Am Own	No	Not Driven	Negated

The MCF5307 device can be in any one of four arbitration states during bus operation: reset, alternate master ownership, implicit ownership, and explicit ownership.

The MCF5307 processor enters the **reset state** whenever \overline{RSTI} or software watchdog reset is asserted in any bus arbitration state. When \overline{RSTI} and the software watchdog reset are negated, the MCF5307 device proceeds to the implicit ownership state or alternate master ownership state, depending on \overline{BG} .

The **alternate master ownership state** denotes the MCF5307 device does not have ownership (\overline{BG} negated) of the bus and the MCF5307 device does not drive the bus. The MCF5307 processor can assert memory control signals (i.e., $\overline{CS}[7:0]$, $\overline{WE}[3:0]$, $\overline{RAS}[1:0]$ or $\overline{CAS}[3:0]$) and transfer acknowledge (\overline{TA}) during this state.

The **implicit ownership state** indicates that the MCF5307 device owns the bus because \overline{BG} is asserted to it. The MCF5307 device, however, is not ready to begin a bus cycle. In this case, the MCF5307 processor keeps the bus three-stated until an internal bus request occurs.

The MCF5307 device explicitly owns the bus when the bus is granted to it (\overline{BG} asserted) and at least one bus cycle has been initiated. The MCF5307 processor asserts \overline{BD} in this state to indicate the MCF5307 device has explicit ownership of the bus. Until \overline{BG} is negated, the MCF5307 device retains explicit ownership of the bus whether or not active bus cycles are being executed. Once \overline{BG} is negated, the MCF5307 processor will relinquish the bus at the end of the current bus cycle. When the MCF5307 processor is ready to relinquish the bus, it negates \overline{BD} and three-states the bus signals.

7.7.6 Multiple External Bus Master Arbitration Protocol (Three-Wire Mode)

The three-wire mode of bus arbitration allows the MCF5307 device to share the external bus with any number of external bus masters. In this mode, an external arbiter must be provided to assign priorities to each of the possible bus masters and determine which master should be allowed use of the external bus. The bus arbitration signals of the MCF5307 device, \overline{BR} , \overline{BD} , and \overline{BG} connect to the bus arbiter, allowing the bus arbiter to control use of the external bus by the MCF5307 device.

The MCF5307 processor requests the bus from the external bus arbiter by asserting \overline{BR} whenever an internal bus request is pending (the ColdFire core requests an access). The MCF5307 processor continues to assert \overline{BR} until after the start of the external bus transfer. The MCF5307 processor can negate \overline{BR} at any time regardless of the \overline{BG} status. If the bus is granted to the MCF5307 processor when an internal bus request is generated, the MCF5307 processor will assert \overline{BD} , allowing the access to begin immediately. The MCF5307 processor always drives \overline{BR} and \overline{BD} . They cannot be directly wire-ORed with other devices.

The external arbiter asserts \overline{BG} to indicate to the MCF5307 processor that it has been granted the bus and can begin a bus cycle after the rising edge of the next CLK. If \overline{BG} is negated while a bus cycle is in progress, the MCF5307 processor relinquishes the bus at the completion of the bus cycle. To guarantee that the bus is relinquished, \overline{BG} must be negated prior to the rising edge of the CLK in which the last \overline{TA} is asserted. Note that the MCF5307 processor considers any series of bus transfers of a burst or a burst-inhibited transfer to be a single bus cycle and does not relinquish the bus until completion of the last transfer of the series.

When the bus has been granted to the MCF5307 processor in response to the assertion of \overline{BR} , one of two situations can occur. In the first case, if the MCF5307 processor has an internal bus request pending, the MCF5307 processor asserts \overline{BD} to indicate explicit bus ownership and begins the pending bus cycle by asserting \overline{TS} . The MCF5307 processor continues to assert \overline{BD} until the external bus master 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 processor owns the and the MCF5307 processor continuously drives the address bus, attributes, and control signals.

In the second situation, the bus is granted to the MCF5307 processor, but the processor does not have an internal bus request pending. In this case, the MCF5307 processor takes implicit ownership of the bus. Implicit ownership of the bus occurs when the MCF5307 processor is granted the bus, but there are no pending bus cycles. The MCF5307 device does not drive the bus and does not assert \overline{BD} if the bus is implicitly owned. If an internal bus request is generated, the MCF5307 device assumes explicit ownership of the bus. If explicit ownership was assumed due to an internal request being generated, the MCF5307 processor immediately begins an access and asserts \overline{BD} . Figure 7-21 illustrates implicit and explicit bus ownership due to an internal bus request being generated.

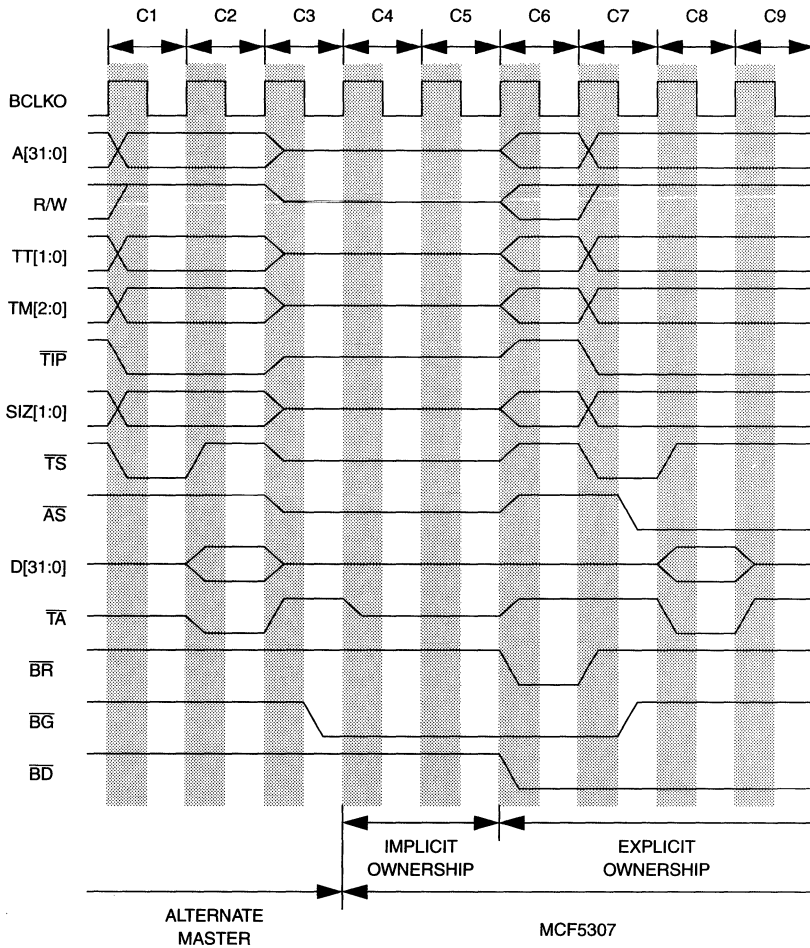


Figure 7-21. Three-Wire Implicit and Explicit Bus Ownership

In Figure 7-21, the alternate master has ownership of the external bus during C1 and C2. In C3, the alternate master releases control of the bus and the external arbiter asserts bus grant (\overline{BG}) to the MCF5307 processor. During C4 and C5, the MCF5307 device is implicit owner because an internal access is not pending. In C5, an internal bus request becomes pending, causing the MCF5307 device to take explicit ownership of the bus in Clock 6 (C6) by asserting \overline{BR} and \overline{BD} . In Clock 7 (C7), the alternate master removes the bus grant to the MCF5307 device. The MCF5307 processor does not relinquish the bus (the MCF5307 continues to assert \overline{BD}) until the end of the transfer.

NOTE

The MCF5307 processor can start a transfer during the CLK cycle after \overline{BG} is asserted. The external arbiter should not assert \overline{BG} to the MCF5307 processor until the previous alternate master has stopped driving the bus. \overline{BG} cannot be asserted while another alternate master transfer is still in progress or damage to the part could occur.

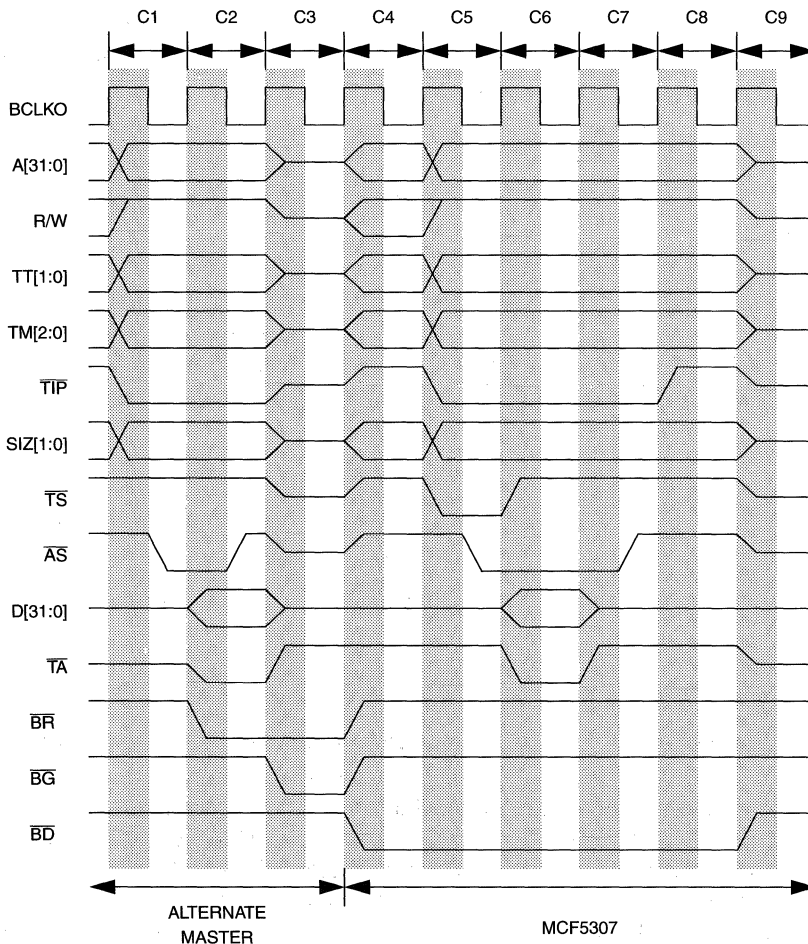


Figure 7-22. Three-Wire Bus Arbitration with External to Master Control Bit Asserted

In Figure 7-22, the alternate master is owner of the external bus during C1 and C2. During C2, the MCF5307 processor requests the external bus because of a pending internal transfer. On Clock C3, the alternate master relinquishes control of the bus and the external arbiter grants the bus to the MCF5307 device by asserting \overline{BG} . At this point, there is an internal access pending so the MCF5307 device asserts \overline{BD} during Clock C4, and begins the access. Thus, the MCF5307 processor becomes the explicit master of the external bus. Also during C4, the external arbiter removes the grant from the MCF5307 processor by negating \overline{BG} . Because the MCF5307 device is the current bus master, it continues to assert \overline{BD} until the current transfer has completed. Because \overline{BG} is negated, the MCF5307 device negates \overline{BD} during C9 and three-states the external bus, thereby passing ownership of the external bus to an alternate master.

\overline{BR} can be used by the external arbiter as an indication that the MCF5307 processor needs the bus. However, there is no guarantee that when the bus is granted to the MCF5307 processor that a bus cycle will be performed. At best, \overline{BR} must be used as a status output that indicates when the MCF5307 processor needs the bus, but not as an indication that the MCF5307 processor is in a certain bus arbitration state.

Figure 7-23 is a high-level bus arbitration state diagram for the MCF5307 bus arbitration protocol, which can be used by external arbiters to predict how the MCF5307 processor operates as a function of external signals. Table 7-10 lists conditions that cause a change to and from the various states. Table 7-11 describes the MCF5307 bus ownership, bus driving, and assertion of \overline{BD} for each state of the bus arbitration state machine.

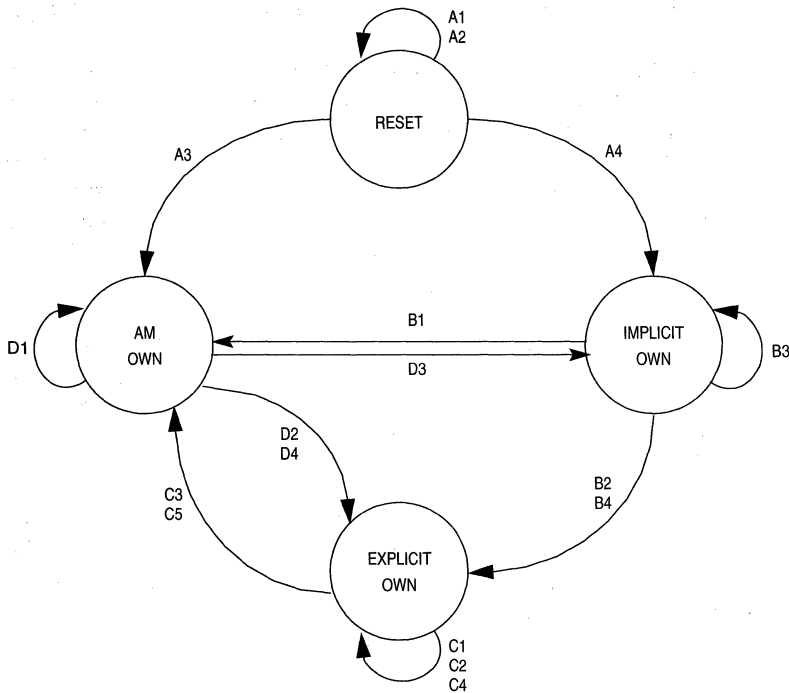


Figure 7-23. MCF5307 Three-Wire Bus Arbitration Protocol State Diagram

Table 7-10. MCF5307 Three-Wire Bus Arbitration Protocol Transition Conditions

PRESENT STATE	CONDITION LABEL	RSTI	SOFTWARE WATCHDOG RESET	BG	INTERNAL BUS REQUEST (IBR)	TRANSFER IN PROGRESS	END OF CYCLE	NEXT STATE
RESET	A1	A	-	-	-	-	-	Reset
	A2	N	A	-	-	-	-	Reset
	A3	N	N	N	-	-	-	Am Own
	A4	N	N	A	-	-	-	Implicit Own
IMPLICIT OWN	B1	N	N	N	-	-	-	Am Own
	B2	N	N	A	-	-	-	Explicit Own
	B3	N	N	A	N	-	-	Implicit Own
	B4	N	N	A	A	-	-	Explicit Own
EXPLICIT OWN	C1	N	N	A	-	-	-	Explicit Own
	C2	N	N	N	-	-	-	Explicit Own
	C3	N	N	N	-	N	-	Am Own
	C4	N	N	N	-	Y	N	Explicit Own
	C5	N	N	N	-	Y	Y	Am Own

Table 7-10. MCF5307 Three-Wire Bus Arbitration Protocol Transition Conditions

AM OWN	D1	N	N	N	-	-	-	Am Own
	D2	N	N	A	-	-	-	Explicit Own
	D3	N	N	A	N	-	-	Implicit Own
	D4	N	N	A	A	-	-	Explicit Own

1)“N” means negated; “A” means asserted; “AM” means alternate master. 2)End of Cycle: Whatever terminates a bus transaction whether it is normal or bus error. Note that bus cycles that result from a burst inhibited transfer are considered part of that original transfer. 3)IBR refers to an internal bus request. The output signals BR is a registered version of IBR when \overline{BG} is negated and \overline{BD} is negated. There is an internal bus request when the ColdFire core requires the external bus for an operand transfer

Table 7-11. MCF5307 Three-Wire Arbitration Protocol State Diagram

STATE	OWN	BUS STATUS	BD
Reset	No	Not Driven	Negated
Implicit Own	Yes	Not Driven	Negated
Explicit Own	Yes	Driven	Asserted
Am Own	No	Not Driven	Negated

The MCF5307 processor can be in any one of four arbitration states during bus operation: reset, alternate master own, implicit ownership, and explicit ownership.

The **reset state** is entered whenever \overline{RSTI} or software watchdog reset is asserted in any bus arbitration state. When \overline{RSTI} and the software watchdog reset are negated, the MCF5307 processor proceeds to the implicit ownership state or alternate master ownership state, depending on \overline{BG} .

The **alternate master ownership state** denotes the MCF5307 processor does not have ownership (\overline{BG} negated) of the bus and the MCF5307 device does not drive the bus. The MCF5307 device can assert memory control signals (i.e., $\overline{CS}[7:0]$, $\overline{WE}[3:0]$, $\overline{RAS}[1:0]$ or $\overline{CAS}[3:0]$) \overline{TA} and \overline{BR} during this state.

The implicit ownership state indicates that the MCF5307 device owns the bus because \overline{BG} is asserted to it. The MCF5307 device, however, is not ready to begin a bus cycle, and it keeps the bus three-stated until an internal bus request occurs.

The MCF5307 processor explicitly owns the bus when the bus is granted to it (\overline{BG} asserted) and at least one bus cycle has initiated. The MCF5307 device asserts \overline{BD} in this state to indicate that it has explicit ownership of the bus. Until \overline{BG} is negated, the MCF5307 device regains explicit ownership of the bus whether or not active bus cycles are being executed. Once \overline{BG} is negated, the MCF5307 processor will relinquish the bus at the end of the current bus cycle. When it is ready to relinquish the bus, the MCF5307 processor negates \overline{BD} and three-states the bus signals.

You can use the bus arbitration state diagram for the MCF5307 three-wire bus arbitration protocol to approximate the high-level behavior of the MCF5307 device. It is assumed that

all \overline{TS} or \overline{AS} signals in a system are tied together and each bus master's \overline{BD} and \overline{BR} signals are connected individually to the external bus arbiter. The external bus arbiter must make sure any alternate bus master has relinquished the bus or will be relinquishing the bus after the next rising edge of CLK before asserting \overline{BG} to the MCF5307 processor. The MCF5307 does not monitor external bus master operation regarding bus arbitration.

NOTE

The MCF5307 processor can start a transfer on the rising edge of CLK the cycle after \overline{BG} is asserted. The external arbiter should not assert \overline{BG} to the MCF5307 processor until the previous alternate master has stopped driving the bus. \overline{BG} cannot be asserted while another alternate master transfer is still in progress or damage to the part could occur.

7.8 RESET OPERATION

7

The MCF5307 processor supports two types of reset: the external master reset input (\overline{RSTI}), which resets the entire MCF5307 device (including the internal PLL module), and the Software Watchdog reset, which resets everything except the internal PLL module.

7.8.1 MASTER RESET

To perform a master reset, an external device asserts the reset input pin (\overline{RSTI}). When power is applied to the system, external circuitry should assert \overline{RSTI} for a minimum of six CLKIN cycles after Vcc is within tolerance. Figure 7-24 is a functional timing diagram of the master reset operation, illustrating relationships among Vcc, \overline{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 processor caused by the random manner in which internal flip-flops power up. \overline{RSTI} is internally synchronized for two CLKINs before being used and must meet the specified setup and hold times to CLKIN only if recognition by a specific CLKIN rising edge is required.

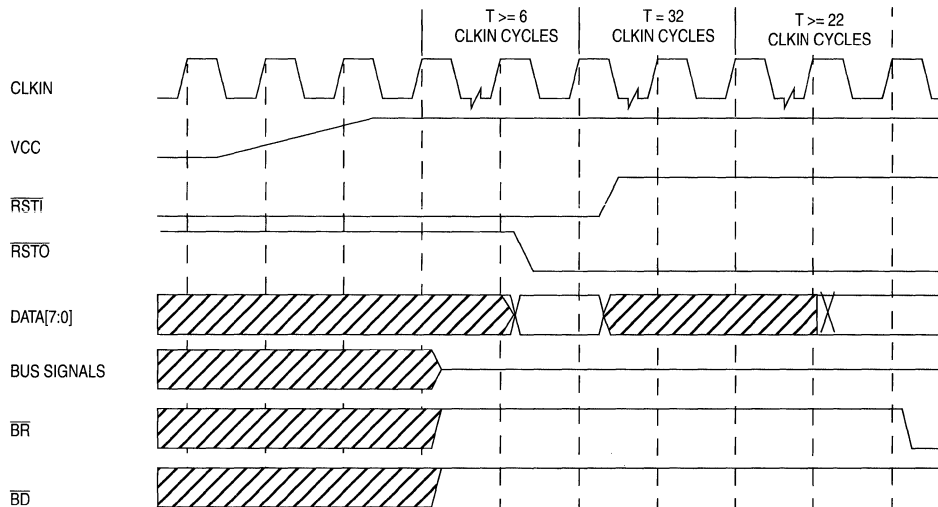


Figure 7-24. Master Reset Timing

During the master reset period, all signals that can be driven to a high-impedance state and all those that cannot are driven to their negated states. Once \overline{RSTI} negates, all bus signals continue to remain in a high-impedance state until the MCF5307 device is granted the bus and the ColdFire core begins the first bus cycle for reset exception processing. A master reset causes any bus cycle (including DRAM refresh cycles) to terminate. In addition, master reset initializes registers appropriately for a reset exception.

7.8.2 SOFTWARE WATCHDOG RESET

The software watchdog reset is performed anytime the executing software does not provide the correct write data sequence with the enable-control bit set. This reset helps prevent runaway software or nonterminated bus cycles. Figure 7-25 is a functional timing

diagram of the software watchdog reset operation, illustrating relationships among $\overline{\text{RSTO}}$ and bus signals.

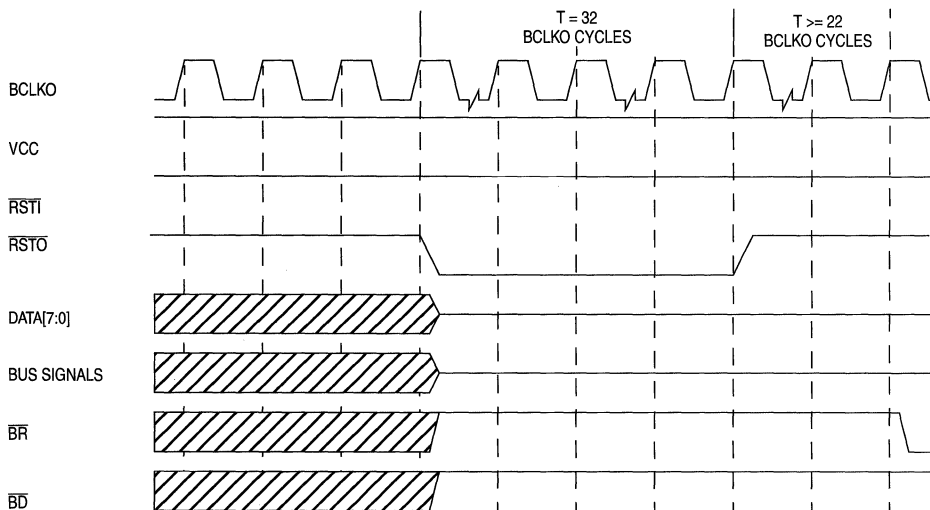


Figure 7-25. Software Watchdog Reset Timing

During the software watchdog reset period, all signals that can be driven to a high-impedance state and all those that cannot are driven to their negated states. Once $\overline{\text{RSTO}}$ negates, all bus signals continue to remain in a high-impedance state until the MCF5307 device is granted the bus and the ColdFire core begins the first bus cycle for reset exception processing.

SECTION 8

SYSTEM INTEGRATION MODULE

8.1 INTRODUCTION

This section details the operation and programming model of the System Integration Module (SIM) registers, including the interrupt controller and system-protection functions for the MCF5307 device. The SIM provides overall control of the internal and external buses and serves as the interface between the ColdFire core processor and the internal peripherals or external devices. The SIM also configures the parallel port and enables the CPU STOP instruction.

8.1.1 Features

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
- **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 with primary and secondary interrupt levels
- **System Protection and Reset Status**
 - Reset status to indicate cause of last reset
 - Software watchdog timer with optional secondary bus monitor functionality
- **Phase Locked Loop Clock Control Register for CPU STOP instruction (PLLCCR)**
 - Control for turning off clocks to core and interrupt levels that turn clocks back on.
- **Bus Arbitration Control Register (MPARK)**
 - Controls default bus master after arbitration
 - Enables display of internal accesses on the external bus for debug
- **Parallel Port Pin Assignment Register (PAR)**

8.2 PROGRAMMING MODEL

8.2.1 SIM Registers Memory Map

Table 8-1 shows the memory map of all the SIM registers. The internal registers in the SIM are memory-mapped registers offset from the MBAR address pointer. The following list addresses several key notes regarding the programming model table:

- You can access the Module Base Address Register in supervisor mode only using the MOVEC instruction with an Rc value of \$C0F.

Table 8-1. SIM Memory Map

MBAR + \$000	System Control Reg	RSR	SYPCR	SWIVR	SWSR
MBAR + \$004	Pin Assignment Reg	PAR		IRQPAR	Reserved
MBAR + \$008	PLL Control Reg	PLLCR	Reserved		
MBAR + \$00C	Bus Master Control Reg	MPARK	Reserved		
MBAR + \$010	-	Reserved			
MBAR + \$014	-				
MBAR + \$018	-				
MBAR + \$01C	-				
MBAR + \$020	-				
MBAR + \$024	-				
MBAR + \$028	-				
MBAR + \$02C	-				
MBAR + \$030	-				
MBAR + \$034	-				
MBAR + \$038	-				
MBAR + \$03C	-				
MBAR + \$040	Interrupt Pending Reg	IPR			
MBAR + \$044	Interrupt Mask Reg	IMR			
MBAR + \$048	Auto Vector Control Reg	Reserved			AVR
MBAR + \$04C	Interrupt Control Reg	ICR0	ICR1	ICR2	ICR3
MBAR + \$050	Interrupt Control Reg	ICR4	ICR5	ICR6	ICR7
MBAR + \$054	Interrupt Control Reg	ICR8	ICR9	ICR10	ICR11

8.3 SIM PROGRAMMING AND CONFIGURATION

8.3.1 Module Base Address Register (MBAR)

The MBAR determines the base address of all internal peripherals as well as the definition of access types allowed for these registers.

The MBAR is a 32-bit write-only supervisor control register that physically resides in the SIM. It is accessed in the CPU address space \$C0F via the MOVEC instruction. (Refer to the *ColdFire Family Programmer's Reference Manual Rev. 1.0* for use of MOVEC instruction). The MBAR can be read when in Debug mode using background debug commands.

At system reset, the MBAR valid bit is cleared to prevent incorrect references to resources before the MBAR is written. The remainder of the MBAR bits are uninitialized. To access the internal peripherals, you should write MBAR with the appropriate base address and set the valid bit after system reset.

All internal peripheral registers occupy a single relocatable memory block along 1-kbyte boundaries. If the MBAR valid bit is set, the base address field is compared to the upper 22 bits of the full 32-bit internal address to determine if an internal peripheral is being accessed. The MBAR masks specific address spaces using the address space fields. Any attempt to access a masked address space will generate an access on the external bus.

8.3.2 Module Base Address Register (MBAR)

The MBAR determines the base address location of all internal module resources such as registers as well as the definition of the types of accesses that are allowed for these resources.

The MBAR is a 32-bit write-only supervisor control register that physically resides in the SIM. It is accessed in the CPU address space \$C0F via the MOVEC instruction. (Refer to the *ColdFire Family Programmer's Reference Manual Rev. 1.0* for use of MOVEC instruction).

At system reset, the MBAR valid bit is cleared to prevent incorrect references to resources before the MBAR is written. The remainder of the MBAR bits are uninitialized. The MBAR can be written to when in Background Debug Mode (BDM). To access the internal modules, MBAR should be written with the appropriate base address after system reset.

All internal peripheral registers occupy a single relocatable memory block along 4Kbyte boundaries. If the MBAR valid bit is set, the base address field is compared to the upper 20 bits of the full 32-bit internal address to determine if an internal peripheral is being accessed. The MBAR masks specific address spaces using the address space fields. Any attempt to access a masked address space will result in an access being generated on the external bus.

Module Base Address Register(MBAR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BA31	BA30	BA29	BA28	BA27	BA26	BA25	BA24	BA23	BA22	BA21	BA20	BA19	BA18	BA17	BA16
RESET:	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA15	BA14	BA13	BA12	-	-	-	WP	-	AM	C/I	SC	SD	UC	UD	V
RESET:	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

Module Base Address Register

BA[31:12] - Base Address

This field defines the base address for a minimum 4Kbyte address range.

WP, AM, SC, SD, C/I, UC and UD - CPU/IACK Mask Bits

- SC - Mask Supervisor Code space in address range
- SD - Mask Supervisor Data space in address range
- UC - Mask User Code space in address range
- UD - Mask User Data space in address range
- C/I - Mask CPU Space and Interrupt Acknowledge Cycle

This field masks specific address spaces, placing the MBAR mapped registers in a specific address space or spaces. If an address space mask bit is cleared, an access to a location in that address space can activate the corresponding MBAR mapped register. If an address space mask bit is set, an access to a location in that address space becomes a regular external bus access.

The address space mask bits are:

8

WP - Write Protect

This bit is the 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

AM - Alternate Master Mask

When AM=0 and an alternate master actually accesses the MBAR mapped registers; SC, SD, UC, and UD are “don't cares” in the address decode.

For each address space mask bit (AM, C/I, SC, SD, UC, UD):

- 0 = Do not mask this address space for MBAR mapped register. Allow access to these registers.
- 1 = Mask this address space from the chip-select activation. Disallow access to these registers; this forces transfer to become a regular external bus cycle.

V - Valid

This bit defines when the base address is valid:

- 0 = Contents of MBAR are not valid
- 1 = Contents of MBAR are valid

The following example shows how to set the MBAR to location \$10000000 using the D0 register. In the example, a “1” in the least significant bit validates the MBAR location. The example assumes all accesses are valid:

```
move.l #10000001,DO
movec DO,MBAR
```

8.3.3 Interrupt Controller

The SIM provides a centralized interrupt controller for all MCF5307 interrupt sources, which includes:

- External interrupts
- Software watchdog timer
- Timer modules
- MBUS module
- UART modules
- DMA module

8.3.4 Interrupt Registers (ICR[11:0], IPR, IMR, AVR, and IRQPAR)

All interrupt sources have their own ICR, as shown in the location and the description of each ICR is detailed in Table 8-2. There are four external interrupt pins with seven possible external interrupts settings at a fixed interrupt level and priority. The IRQ PAR register determines these settings. You can program the external interrupts to supply an autovector or run an external interrupt acknowledge cycle. Refer to the Autovector Control Register (AVR).

Table 8-2. Interrupt Control Register Memory Map

ADDRESS	NAME	WIDTH	DESCRIPTION	RESET VALUE	ACCESS
MBAR + \$04C	ICR0	8	SWT	\$00	R/W
MBAR + \$04D	ICR1	8	Timer 1	\$00	R/W
MBAR + \$04E	ICR2	8	Timer 2	\$00	R/W
MBAR + \$04F	ICR3	8	MBUS	\$00	R/W
MBAR + \$050	ICR4	8	UART 1	\$00	R/W
MBAR + \$051	ICR5	8	UART 2	\$00	R/W
MBAR + \$052	ICR6	8	DMA 0	\$00	R/W
MBAR + \$053	ICR7	8	DMA 1	\$00	R/W
MBAR + \$054	ICR8	8	DMA 2	\$00	R/W
MBAR + \$055	ICR9	8	DMA 3	\$00	R/W
MBAR + \$056	ICR10	8	Reserved	-	-
MBAR + \$057	ICR11	8	Reserved	-	-

Internal interrupts are programmed to a level and priority. All internal interrupts have a unique Interrupt Control Register. There are 35 possible priority levels, including internal and external interrupts.

	7	6	5	4	3	2	1	0
AVEC	-	-	IL[2]	IL[1]	IL[0]	IP[1]	IP[0]	
RESET:	0	-	-	0	0	0	0	0

Interrupt Control Register

AVEC - Autovector Enable

This bit determines whether the interrupt-acknowledge cycle input (for the internal interrupt level indicated in IL[2:0] for each interrupt) requires an autovector response.

- AVEC = 0; Interrupting source returns vector during interrupt-acknowledge cycle
- AVEC = 1; SIM generates auto vector during interrupt acknowledge cycle

IL[2:0] - Interrupt Level

These bits indicate the interrupt level assigned to each interrupt input.

IP[1:0] - Interrupt Priority

These bits indicate the interrupt priority within the interrupt level assignment.

- IP[1] = 0; Lower priority than external interrupt at this level
- IP[1] = 1; Higher priority than external interrupt at this level
- IP[0] = 0; Lower priority than internal interrupt at this level with same IP[1]
- IP[0] = 1; Higher priority than internal interrupt at this level with same IP[1]

Table 8-3. Interrupt Priority Scheme

INTERRUPT LEVEL	INTERNAL MODULE ICR REG			INTERRUPT SOURCE	EXTERNAL INTERRUPT PAR REG		
	IL[2:0]	IP[1]	IP[0]		IRQPAR[2]	IRQPAR[1]	IRQPAR[0]
7	111	1	1	Internal Module	-	-	-
7	111	1	0	Internal Module	-	-	-
7	---	-	-	External Interrupt Pin IRQ7	-	-	-
7	111	0	1	Internal Module	-	-	-
7	111	0	0	Internal Module	-	-	-
6	110	1	1	Internal Module	-	-	-
6	110	1	0	Internal Module	-	-	-
6	---	-	-	External Interrupt Pin IRQ3	-	1	-
6	110	0	1	Internal Module	-	-	-
6	110	0	0	Internal Module	-	-	-
5	101	1	1	Internal Module	-	-	-
5	101	1	0	Internal Module	-	-	-
5	---	-	-	External Interrupt Pin IRQ5	0	-	-
5	101	0	1	Internal Module	-	-	-
5	101	0	0	Internal Module	-	-	-
4	100	1	1	Internal Module	-	-	-
4	100	1	0	Internal Module	-	-	-
4	---	-	-	External Interrupt Pin IRQ5	1	-	-
4	100	0	1	Internal Module	-	-	-
4	100	0	0	Internal Module	-	-	-
3	011	1	1	Internal Module	-	-	-
3	011	1	0	Internal Module	-	-	-
3	---	-	-	External Interrupt Pin IRQ3	-	0	-
3	011	0	1	Internal Module	-	-	-
3	011	0	0	Internal Module	-	-	-
2	010	1	1	Internal Module	-	-	-
2	010	1	0	Internal Module	-	-	-
2	---	-	-	External Interrupt Pin IRQ1	-	-	1
2	010	0	1	Internal Module	-	-	-
2	010	0	0	Internal Module	-	-	-
1	001	1	1	Internal Module	-	-	-
1	001	1	0	Internal Module	-	-	-
1	---	-	-	External Interrupt Pin IRQ1	-	-	0
1	001	0	1	Internal Module	-	-	-
1	001	0	0	Internal Module	-	-	-

Note: Motorola recommends that you not assign multiple internal modules to the same interrupt level and same interrupt priority when configuring the ICR registers.

AVEC[7:1] - Auto Vector Control

These bits determine whether the external interrupt at that level is auto vectored.

AVECx = 0; Interrupting source returns vector during interrupt-acknowledge cycle
 AVECx = 1; SIM generates auto vector during interrupt-acknowledge cycle

BLK - Block address strobe (\overline{AS}) for external AVEC access.

BLK = 0; Do not block

BLK = 1; Block address strobe

7	6	5	4	3	2	1	0
AVEC7	AVEC6	AVEC5	AVEC4	AVEC3	AVEC2	AVEC1	BLK
RESET:							
0	0	0	0	0	0	0	0

Autovector Control Register (AVR)

IM[16:1] - Interrupt Mask

Each bit corresponds to an interrupt source defined by the Interrupt Control Register (ICR) or Autovector Register (AVR). You mask an interrupt by setting the corresponding bit in the IMR. At system reset, all defined bits are initialized to a logic one.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IM[16]
RESET:															
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM[15]	IM[14]	IM[13]	IM[12]	IM[11]	IM[10]	IM[9]	IM[8]	IM[7]	IM[6]	IM[5]	IM[4]	IM[3]	IM[2]	IM[1]	-
RESET:															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-

Interrupt Mask Register (IMR)

IP[16:1] - Interrupt Pending

Each bit corresponds to an interrupt source defined by the Interrupt Control Register. This register contains a registered copy of the interrupt signal the interrupting source generates. The corresponding bit in this register reflects the state of the interrupt signal even if the corresponding mask bit is set. The IPR is a read-only longword register.

1 = The corresponding interrupt source has an interrupt pending
 0 = The corresponding interrupt source does not have an interrupt pending

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-	-	-	-	-	-	-	DMA3	DMA2
RESET:															
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMA1	DMA0	UART2	UART1	MBUS	TIMER 2	TIMER 1	SWT	EINT7	EINT6	EINT5	EINT4	EINT3	EINT2	EINT1	-
RESET:															
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Interrupt Pending Register

8.3.4.1 INTERRUPT ASSIGNMENT REGISTER (IRQPAR). The IRQPAR lets you change the level the external interrupts IRQ5, IRQ3, and IRQ1. The setting of the IRQPAR[2:0] bits determine the interrupt level the external interrupt pins.

For example:

- Note: External IRQ[7] pin is always internal interrupt level 7
- Setting the IRQPAR2 bit to “1” will set the IRQ[5] pin to internal interrupt level 4
- Setting the IRQPAR2 bit to “0” will set the IRQ[5] pin to internal interrupt level 5
- Setting the IRQPAR1 bit to “1” will set the IRQ[3] pin to internal interrupt level 6
- Setting the IRQPAR1 bit to “0” will set the IRQ[3] pin to internal interrupt level 3
- Setting the IRQPAR0 bit to “1” will set the IRQ[1] pin to internal interrupt level 2
- Setting the IRQPAR0 bit to “0” will set the IRQ[1] pin to internal interrupt level 1

Additionally, the IRQPAR2 and IRQPAR1 have been assigned to interrupts so as to allow the IRQ5 and IRQ3 external pins to raise priority over each other by becoming IRQ4 and IRQ6 internal interrupts, respectively. See the IRQPAR programming model that follows.

Interrupt Assignment Register (IRQPAR)								Address MBAR + \$06
7	6	5	4	3	2	1	0	-
IRQPAR[2]	IRQPAR[1]	IRQPAR[0]	-	-	-	-	-	-
RESET:								
0	0	0	0	0	0	0	0	

Interrupt Assignment Register (IRQPAR)

IRQPAR[2:0]- Interrupt Port Assignment bits

Refer to Table 8-4.

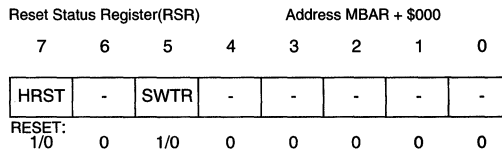
Table 8-4. Signals Selected with IRQPAR

IRQPAR	IRQPAR=0	IRQPAR=1	EXTERNAL PIN CONNECTION
IRQPAR[2]	IRQ[5]	IRQ[4]	IRQ[5]
IRQPAR[1]	IRQ[3]	IRQ[6]	IRQ[3]
IRQPAR[0]	IRQ[1]	IRQ[2]	IRQ[1]

8.3.5 System Protection And Reset Status

8.3.5.1 RESET STATUS REGISTER (RSR). The RSR contains a bit for each reset source to the SIM. A bit set to 1 indicates the last type of reset that occurred. The RSR is updated by the reset control logic when the reset is complete. Only one bit will be set at any one time in the RSR. The register will reflect the cause of the most recent reset. If a reset occurs and you have failed to clear this register, reset control logic will clear any uncleared bits and set the bit for the correct cause of reset. You clear a bit by writing a one to that bit location; writing a zero has no effect. The illustration that follows shows the RSR programming model.

The RSR is an 8-bit supervisor read-write register.



Reset Status Register (RSR)

HRST - Hard Reset or System Reset

- 1 = An external device driving \overline{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.

SWTR - Software Watchdog Timer Reset

- 1 = The last reset was caused by the software watchdog timer. If SWRI in the SYPCR is set and the software watchdog timer times out, a hard reset occurs.

8.3.5.2 SOFTWARE WATCHDOG TIMER (SWT). The software watchdog timer prevents system lockup should the software become trapped in loops with no controlled exit. The SWT can be enabled or disabled via the SWE bit in the SYPCR. If enabled, the SWT requires the execution of a software watchdog servicing sequence periodically. If

this periodic servicing action does not occur, the SWT times out resulting in a SWT IRQ or hardware reset, as programmed by the SWRI bit in the SYPCR. If the SWT times out and SWTA enable bit is set, the SWT IRQ will assert. If after another timeout, the SWT IACK cycle has not occurred, the SWT TA signal will assert in an attempt to terminate the bus cycle and allow IACK cycle to proceed. The setting of the SWTAVAL flag bit in the SYCPR register indicates that the SWT TA signal was asserted. Figure 8-2 visually depicts the SWTA functionality when terminating a locked bus.

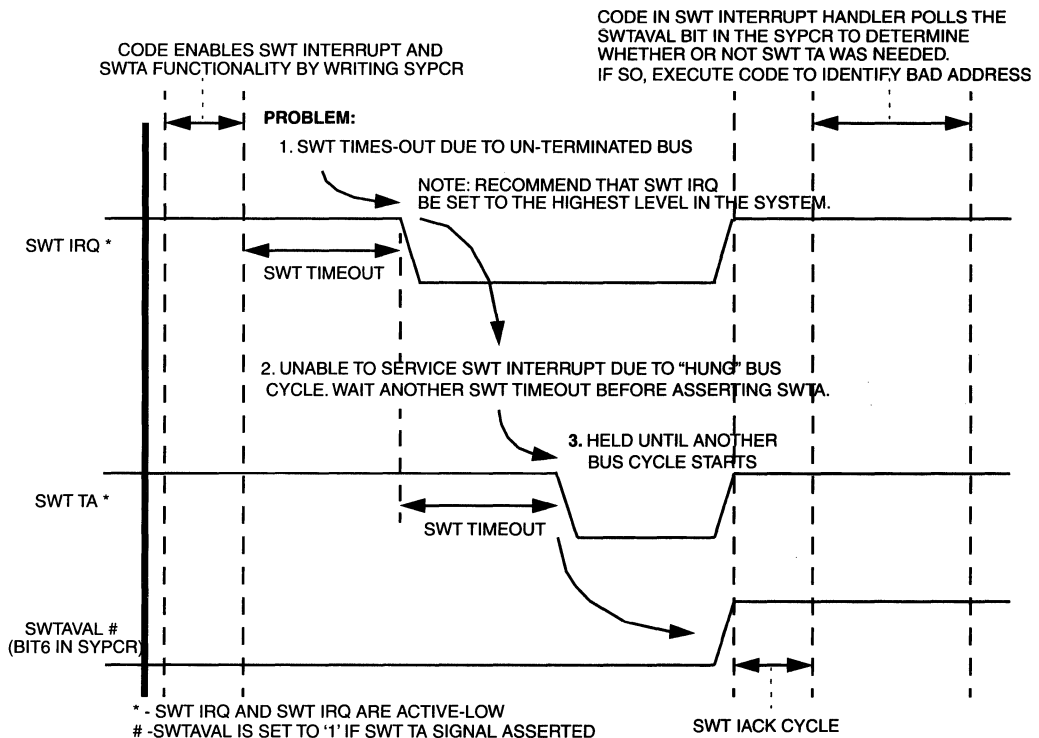


Figure 8-1. MCF5307 Embedded System Recovery from Unterminated Access

When the SWT times out and SWRI register bit is programmed for a hardware reset, an internal reset will be asserted, and the SWTR register bit will be set in the RSR.

To prevent SWT from interrupting or resetting, you must service the SWSR register. The SWT service sequence consists of the following steps:

1. Write \$55 to SWSR
2. Write \$AA to the SWSR

Both writes must occur in the order listed prior to the SWT timeout, but any number of instructions or accesses to the SWSR can be executed between the two writes. This order allows interrupts and exceptions to occur, if necessary, between the two writes.

Be careful when changing SYPCR values after the SWT has been enabled with the setting of the SWE register bit as it is difficult to determine the state of the SWT while running. The SWP and SWT[1:0] bits in SYPCR determine the SWT timeout period. Note that a function of the SWT[1:0] and SWP registers data is compared against the SWT timer versus the timer being “pre-loaded” with some particular countdown value determined by the SWP and SWT[1:0] bits. If you want to change one of these values in the SYPCR, take the following steps:

1. Disable SWT by writing a 0 to the SWE bit in SYPCR.
2. Service the SWSR, write \$55, then write \$AA to SWSR. This action resets the counter.
3. Re-write new SWT[1:0] and SWP values to SYPCR register.
4. Re-enable SWT by writing a 1 to SWE bit in SYPCR. You can do this in step 3.

8.3.5.3 SYSTEM PROTECTION CONTROL REGISTER (SYPCR). The SYPCR controls the software watchdog timer, timeout periods, and software watchdog timer transfer acknowledge.

8

The SYPCR is an 8-bit supervisor read-write register. The register can be read at any time, but can be written only if the SWT isn't pending. At system reset, the software watchdog timer is disabled

System Protection Control Register(SYPCR): Address MBAR + \$01

	7	6	5	4	3	2	1	0
	SWE	SWRI	SWP	SWT[1]	SWT[0]	SWTA	SWTAVAL	-
RESET:	0	0	0	0	0	0	0	0

System Protection Control Register (SYPCR)

SWE - Software Watchdog Enable

- 0 = SWT disabled
- 1 = SWT enabled

SWRI - Software Watchdog Reset/Interrupt Select

- 0 = If SWT timeout occurs, SWT generates an interrupt to the core processor at the level programmed into the IL bits of ICR0 (see Table 8-2.).
- 1 = SWT 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).

SWP - Software Watchdog Prescaler

- 0 = SWT clock not prescaled
- 1 = SWT clock prescaled by a value of 512

SWT[1:0] - Software Watchdog Timing Delay

These bits (along with the SWP bit) select the timeout period for the SWT as shown in Table 8-5. At system reset, the software watchdog timer is set to the minimum timeout period.

Table 8-5. SWT Timeout Period

SWP	SWT[1:0]	SWT TIMEOUT PERIOD
0	00	2^9 / System Frequency
0	01	2^{11} / System Frequency
0	10	2^{13} / System Frequency
0	11	2^{15} / System Frequency
1	00	2^{18} / System Frequency
1	01	2^{20} / System Frequency
1	10	2^{22} / System Frequency
1	11	2^{24} / System Frequency

NOTE

If the SWP and SWT bits are modified to select a new software timeout, you must perform the software service sequence (\$55 followed by \$AA written to the SWSR) before the new timeout period takes effect.

SWTA - Software Watchdog Transfer Acknowledge Enable

0 = SWTA Transfer Acknowledge disabled

1 = SWTA Assert Transfer Acknowledge enabled. After 1 SWT timeout period of the unacknowledged assertion of the SWT interrupt, the Software Watchdog Transfer Acknowledge will assert, which allows SWT to terminate a bus cycle and allow the IACK to occur

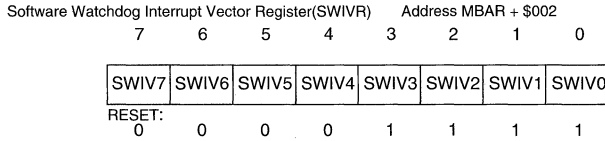
SWTAVAL - Software Watchdog Transfer Acknowledge Valid

0 = SWTA Transfer Acknowledge has NOT occurred

1 = SWTA Transfer Acknowledge has occurred. Write a 1 to clear this flag bit

8.3.5.4 SOFTWARE WATCHDOG INTERRUPT VECTOR REGISTER (SWIVR). The SWIVR contains the 8-bit interrupt vector the SIM returns during an interrupt-acknowledge cycle in response to a SWT-generated interrupt. The following register illustrates the SWIVR programming model.

The SWIVR is an 8-bit supervisor write-only register. This register is set to the uninitialized vector \$0F at system reset.

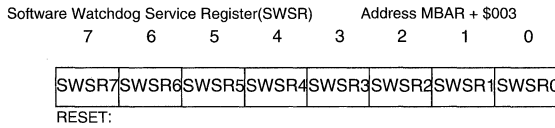


Software Watchdog Interrupt Vector Register (SWIVR)

8.3.5.5 SOFTWARE WATCHDOG SERVICE REGISTER (SWSR). The SWSR is where the SWT servicing sequence should be written. To prevent an SWT timeout, you should write a \$55 followed by a \$AA to this register. Both writes must be performed in the order listed prior to the SWT timeout, but any number of instructions or accesses to the SWSR can be executed between the two writes. If the SWT has already timed out, writing to this register will have no effect in negating the SWT interrupt. The following register illustrates the SWSR programming model.

8

The SWSR is an 8-bit supervisor write-only register. At system reset, the contents of SWSR are uninitialized.



Software Watchdog Service Register (SWSR)

8.3.6 Phase-Locked-Loop Clock Control for CPU STOP Instruction

8.3.6.1 PHASE-LOCKED LOOP CONTROL REGISTER (PLLCR). The ENBSTOP bit must be set to 1 in order for the ColdFire CPU STOP instruction to be acknowledged. After reset, this bit will be cleared to 0 and must be set to 1 for the MCF5307 to enter low-power modes. Only clocks to the core processor are turned off because of the CPU STOP instruction. All internal modules remain clocked and can generate interrupts to restart the ColdFire core.

The PLLCR control register bits PLLIPL[2:0] determine the minimum level at which an interrupt (decoded as an Interrupt Priority Level or IPL) must occur to awaken the PLL. The PLL will then turn clocks back on to the core processor and interrupt exception processing will take place.

Table 8-6 outlines settings of PLLIPL[2:0] to be compared against the interrupt ranges that awaken the core processor from a CPU STOP instruction.

The PLLCR is an 8-bit supervisor read/write register.

PLL Control Register (PLLCR)				Address MBAR + \$08			
7	6	5	4	3	2	1	0
ENBSTOP	PLLIPL[2]	PLLIPL[1]	PLLIPL[0]	-	-	-	-
RESET:	0	0	0	0	0	0	0

PLL Control Register (PLLCR)

ENBSTOP- Enable CPU STOP instruction

0 = ENBSTOP disabled

1 = ENBSTOP enabled, STOP instruction will turn off clocks to the ColdFire core

PLLIPL[2:0]- PLL Interrupt Priority Level to awaken from CPU STOP

Refer to Table 8-6.

Table 8-6. Settings of PLLIPL[2:0] and Interrupts that Awaken Core

PLLIPL[2:0]	INTERRUPTS THAT WILL TURN ON CLOCKS TO THE CORE PROCESSOR
000	Any interrupts will awaken core
001	Interrupts[7:2]
010	Interrupts[7:3]
011	Interrupts[7:4]
100	Interrupts[7:5]
101	Interrupts[7:6]
110	Interrupts[7]
111	No interrupts will awaken core

8.3.7 Bus Arbitration Control

8.3.7.1 DEFAULT BUS MASTER REGISTER (MPARK). The MPARK determines the Default Bus Master internal bus arbitration. Additionally, the MPARK configures internal to external arbitration for internal transfers. Table 8-7 discusses the MPARK bit encoding.

The MPARK is an 8-bit read-write register.

Default Bus Master Register (MPARK): Address MBAR + \$0C

	7	6	5	4	3	2	1	0
	PARK[1]	PARK[0]	IARBCTRL	EARBCTRL	SHOWDATA	-	-	-
RESET:	0	0	0	0	0	0	0	0

Default Bus Master Register (MPARK)

The IARBCTRL bit controls access for external masters to the MCF5307 internal bus.

In a single master system, the IARBCTRL bit should remain cleared (0), disabling internal arbitration by external masters. In this scenario, the PARK[1:0] bits will only apply to the priority of internal masters over one another. Note that the internal DMA (master 3) has a higher priority over the ColdFire core (master 2) if the internal DMA has its bandwidth BWC[2:0] bits set to 000 (maximum bandwidth).

8

In multiple master systems that expect to use internal resources like the DRAM controller or chip-selects, internal arbitration should be enabled by setting IARBCTRL to 1. Additionally, the PARK[1:0] bits must be set to 00 to make the external master the highest priority master. This prevents a potential locked bus situation. This condition arises when the EARBCTRL is set to 1 and an external master attempts to gain access to the internal bus. Unfortunately, the external master will lose arbitration because it is not the highest priority bus master (PARK[1:0] not set to 00). Meanwhile, an internal master of higher priority tries to run an external bus cycle; however the external arbitration logic will not issue the BG signal to the MCF5307 causing the internal bus master to wait indefinitely. In situations where the internal resources of the DRAM controller or chip-selects aren't needed by the external master, the IARBCTRL bit may be cleared to 0 and the PARK[1:0] can then be set to values that apply to the priority of the internal masters only.

The EARBCTRL bit determines whether or not the internal bus masters, core processor and internal DMA, will have to arbitrate for the external bus for transfers that hit internal register spaces (MBAR+register offset).

In a single master system, having the EARBCTRL bit set or cleared has no impact on performance from an arbitration standpoint. It is likely that the system designer has the BG signal tied low, wherein the MCF5307 device always owns the external bus and internal register transfers are going to the external bus already. In a system where the MCF5307 device is the only master, this bit may remain cleared to 0. If the system needs external visibility of the internal register transfers data bus values (for system debug purposes) then both EARBCTRL and the SHOWDATA bits must both be set to 1. When an internal register transfer is driven externally it is important to note that the XTA signal 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 AS signal and all chip-select-related strobe signals will not assert.

In multiple master systems, disabling arbitration with the EARBCTRL bit will allow performance improvement because internal register bus transfer cycles aren't interfering with the external bus. Having internal transfers go external hurts performance in two possible ways. If the internal master doesn't get the bus right away, the core is stalled until it wins arbitration of the external bus; or if the core does win the arbitration instantly, it may kick the external master off the external bus unnecessarily for a transfer that didn't need the external bus in the first place. For debug, the EARBCTRL and SHOWDATA bits can be set to 1, to gain external visibility of the internal bus cycles where this performance penalty isn't a concern.

Table 8-7. Default Bus Master Selected with PARK[1:0]

PARK[1:0]	DEFAULT BUS MASTER NUMBER
00	Park on master 1 (External)
01	Park on alternate master 2 (ColdFire Core)
10	Park on alternate master 3 (Internal DMA)
11	Park on current master

Table 8-8. Park on Master 1 Priority (PARK[1:0] = 00)

MASTER NUMBER	PRIORITY	BUS MASTER NAME
1	Highest	External
2	-	ColdFire Core
3	Lowest	Internal DMA

Table 8-9. Park on Alternate Master 2 Priority (PARK[1:0] = 01)

MASTER NUMBER	PRIORITY	BUS MASTER NAME
2	Highest	ColdFire Core
1	-	External
3	Lowest	Internal DMA

Table 8-10. Park on Alternate Master 3 Priority (PARK[1:0] = 10)

MASTER NUMBER	PRIORITY	BUS MASTER NAME
3	Highest	Internal DMA
1	-	External
2	Lowest	ColdFire Core

Table 8-11. Park on Current Master Priority (PARK[1:0] = 11)

MASTER NUMBER	PRIORITY	BUS MASTER NAME
1	Highest	External
2	-	ColdFire Core
3	Lowest	Internal DMA

Park on Current Master Note: When using the park on current master setting, the first master to arbitrate for the bus becomes the current master. The corresponding priority scheme should be interpreted as the priority of the next current master once the current master finishes.

Important Note: PARK[1:0] must be set to 00 and IARBTRL must be set to 1 if external masters are using internal resources like the DRAM controller or chip selects.

Additional Note: The Internal DMA (master 3) will have higher priority than the ColdFire Core (master 2) if the internal DMA has its bandwidth BWC[2:0] bits set to 000 (maximum bandwidth).

IARBCTRL - External to internal bus arbitration enable.

- 0 = Arbitration disabled
- 1 = Arbitration enabled

EARBCTRL - Enable internal register memory space to external bus arbitration.

- 0 = Arbitration disabled
- 1 = Arbitration enabled

EARBCTRL Note: Internal register memory space is considered all registers mapped off the MBAR (MBAR+offset registers). These include the programming models for the SIM, DMA, Chip Selects, Timers, UART, I2C, Parallel Port, etc.). It is important to note that these registers don't include the MBAR itself- only the core can access the MBAR.

SHOWDATA - Enable to drive Internal register data bus to external bus.

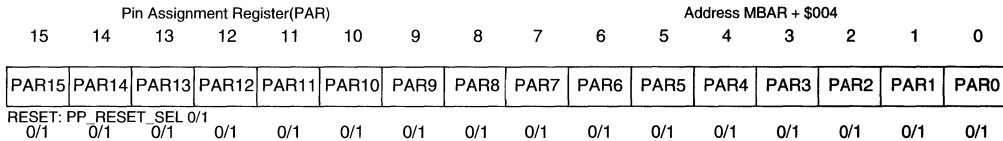
Note: The EARBCTRL bit must be set to 1 for this function to work.

- 0 = Don't drive internal register data bus values to external bus
- 1 = Drive internal register data bus values to external bus

8.3.8 Parallel Port Pin Assignment Register (PAR)

8.3.8.1 PIN ASSIGNMENT REGISTER (PAR). The PAR lets you select certain signal pin assignments. You can select between upper addresses, transfer in progress, DMA requests, transfer modifier, and transfer type or the parallel port signals. The following register illustrates the PAR programming model. Table 8-12 describes which bits in PAR to write for a particular output parallel port setting. The PAR register is 16 bits wide.

IMPORTANT NOTE: The system designer controls the reset value of this register by driving the external data bus bit 4 (Data[4]) with a 1 or 0 when the RSTI (Reset input to MCF5307) negates. The logic level on Data[4] pin that is latched at negation of RSTI is loaded into all bit positions of PP_RESET_SEL. The system is configured as PP[15:0] if Data[4] is 0; otherwise alternate pin functions selected by PAR=1 will be used. This value may be driven with a weak pullup or pulldown resistor.



Parallel Port Pin Assignment Register (PAR)

Table 8-12. Signals Selected with PAR

PAR	PAR=0	PAR=1
PAR15	PP[15]	ADDR[31]
PAR14	PP[14]	ADDR[30]
PAR13	PP[13]	ADDR[29]
PAR12	PP[12]	ADDR[28]
PAR11	PP[11]	ADDR[27]
PAR10	PP[10]	ADDR[26]
PAR9	PP[9]	ADDR[25]
PAR8	PP[8]	ADDR[24]
PAR7	PP[7]	XTIP
PAR6	PP[6]	DREQ[0]
PAR5	PP[5]	DREQ[1]
PAR4	PP[4]	TM[2]
PAR3	PP[3]	TM[1]

Table 8-12. Signals Selected with PAR (Continued)

PAR	PAR=0	PAR=1
PAR2	PP[2]	TM[0]
PAR1	PP[1]	TT[1]
PAR0	PP[0]	TT[0]

SECTION 9

CHIP-SELECT MODULE

9.1 INTRODUCTION

This section details the specification of the Chip-Select Module (CS) for the MCF5307 device. The Chip-Select Module provides user-programmable control of the eight chip-select outputs, four byte/byte-write enable outputs, and one output-enable signal.

This section also addresses the operation and programming model of the CS registers, including the Chip-Select Address, Mask and Control Registers.

9.1.1 Features

The following list summarizes the key chip-select features:

- Eight chip select signals; two with programmable addresses ($\overline{CS0}$ and $\overline{CS1}$); six with fixed offset from a programmable base address
- Address masking for memory block sizes from 64K to 2G for $\overline{CS0}$ and $\overline{CS1}$
- Fixed memory block sizes of 2M for $\overline{CS2}$ - $\overline{CS7}$
- Programmable wait states and port sizes
- External master access to chip-selects

9.2 CHIP-SELECT SIGNALS

The MCF5307 device provides eight programmable chip-selects that can directly interface with SRAM, EPROM, EEPROM, and peripherals.

9.2.1 Chip-Selects - ($\overline{CS}[7:0]$)

You can program each chip-select for an address location as well as for masking capabilities, port size and burst-capability indication, wait-state generation, and internal/external termination. A reset disables all chip-selects. $\overline{CS}[0]$ is also unique because it can function at reset as a global chip-select that allows you to select boot ROM at any defined address space. $\overline{CS}[0]$ is the only chip-select initialized out of reset. Port size and termination (internal vs. external) for $\overline{CS}[0]$ are configured by the levels on D[7:0] when \overline{RSTI} negates.

9.2.2 Output Enable - (\overline{OE})

This signal is sent to the interfacing memory and/or peripheral to enable a read transfer. It is asserted and negated on the falling edge of the clock. This signal is asserted only when there is a match of one of the chip-select for the current address decode.

9.2.3 Byte Enables/Byte Write Enables - ($\overline{BE}[3:0]/\overline{BWE}[3:0]$)

The four byte-enable pins are multiplexed with the byte-write-enable pins of the MCF5307 device. You can program each of these four multiplexed pins through the Chip-Select Control Register on an individual chip-select basis.

These generated signals provide byte data select signals, which are decoded from the SIZx A1 and A0 signals in addition to the programmed port size and burstability of the memory being accessed, as shown in Table 9-1.

Table 9-1. Data Bus Byte Write Enable Signals

TRANSFER SIZE	PORT SIZE	BURST	A1	A0	BWE0	BWE1	BWE2	BWE3
					D[31:24]	D[23:16]	D[15:8]	D[7:0]
Byte	8-bit	Disabled	0	0	0	1	1	1
			0	1	0	1	1	1
			1	0	0	1	1	1
			1	1	0	1	1	1
		Enabled	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	Disabled	0	0	0	1	1	1
			0	1	1	0	1	1
			1	0	0	1	1	1
			1	1	1	0	1	1
		Enabled	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	Disabled	0	0	0	1	1	1
			0	1	1	0	1	1
			1	0	1	1	0	1
			1	1	1	1	1	0
		Enabled	0	0	0	1	1	1
			0	1	1	0	1	1
			1	0	1	1	0	1
			1	1	1	1	1	0

Table 9-1. Data Bus Byte Write Enable Signals

TRANSFER SIZE	PORT SIZE	BURST	A1	A0	BWE0	BWE1	BWE2	BWE3	
					D[31:24]	D[23:16]	D[15:8]	D[7:0]	
Word	8-bit	Disabled	0	0	0	1	1	1	
			0	1	0	1	1	1	
			1	0	0	1	1	1	
			1	1	0	1	1	1	
		Enabled	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	Disabled	0	0	0	0	1	1	
			1	0	0	0	1	1	
		Enabled	0	0	0	0	1	1	
			1	0	0	0	1	1	
	32-bit	Disabled	0	0	0	0	1	1	
			1	0	1	1	0	0	
		Enabled	0	0	0	0	1	1	
			1	0	1	1	0	0	
	Longword	8-bit	Disabled	0	0	0	1	1	1
				0	1	0	1	1	1
				1	0	0	1	1	1
				1	1	0	1	1	1
			Enabled	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		Disabled	0	0	0	0	1	1	
			1	0	0	0	1	1	
		Enabled	0	0	0	0	1	1	
			1	0	0	0	1	1	
32-bit		Disabled	0	0	0	0	0	0	
		Enabled	0	0	0	0	0	0	

Table 9-1. Data Bus Byte Write Enable Signals

TRANSFER SIZE	PORT SIZE	BURST	A1	A0	BWE0	BWE1	BWE2	BWE3
					D[31:24]	D[23:16]	D[15:8]	D[7:0]
Line	8-bit	Disabled	0	0	0	1	1	1
			0	1	0	1	1	1
			1	0	0	1	1	1
		Enabled	1	1	0	1	1	1
			0	0	0	1	1	1
			0	1	0	1	1	1
	16-bit	Disabled	1	0	0	1	1	1
			0	0	0	0	1	1
		Enabled	0	0	0	0	1	1
			1	0	0	0	1	1
	32-bit	Disabled	0	0	0	0	0	0
		Enabled	0	0	0	0	0	0

9.3 CHIP-SELECT OPERATION

9.3.1 Chip-Select Module

The Chip-Select Module provides a glueless interface to many types of external memory. The Chip-Select Module includes the needed external control signals to interface to SRAM, PROM, EPROM, EEPROM, FLASH and peripherals.

Each of the eight chip-select outputs has an associated mask register and control register.

Chip-Selects ($\overline{CS0}$ and $\overline{CS1}$)

- Have individual 16-bit base address registers
- Have individual 32-bit mask register which provide for 16-bit address masking.
- Have individual 16-bit control register which provides port size and burst capability indication, wait state generation, and automatic acknowledge generation features.

Chip-Selects ($\overline{CS2}$ through $\overline{CS7}$)

- All have a common 8-bit base address register (CSBAR)
- Each has a 16-bit control register which provides port size and burst capability indication, wait state generation, and automatic acknowledge generation features.
- Each has a 32-bit mask register which provides for the selection of specific address space accesses.
- Address blocks for each chip select is 2MB.

Chip-select 0 provides special functionality. It is a “global” chip-select after reset and provides relocatable boot ROM capability. You can program $\overline{CS0}$, $\overline{CS1}$ and $\overline{CS7}$ to assert during specific CPU space accesses such as interrupt-acknowledge cycles.

9.3.1.1 GENERAL CHIP-SELECT OPERATION. The general-purpose chip-selects are controlled by the Chip-Select Mask Register (CSMR), the Chip-Select Control Register (CSCR), and by the Chip-Select Address Registers (CSAR & CSBAR). There is one CSMR and one CSCR for each of the chip-selects ($\overline{CS0}$ - $\overline{CS7}$).

Chip-select $\overline{CS0}$ and $\overline{CS1}$ have individual CSAR. Chip-selects $\overline{CS2}$ - $\overline{CS7}$ use a common base address register (CSBAR).

Chip-Selects $\overline{CS0}$ and $\overline{CS1}$

- The chip-select address register controls the base address space of the chip-select.
- The chip-select mask register controls the memory block size and addressing attributes of the chip-select.
- The chip-select control register programs the features of the chip-select signals.

Chip-Selects $\overline{CS2}$ - $\overline{CS7}$

- The chip-select base address register CSBAR defines the base address for chip-selects 2 through 7.
- The individual chip-select mask registers control the addressing attributes of the chip-select.
- The individual chip-select control registers program the assertion features of the chip-select signals.

The MCF5307 processor will compare the address and mask in chip-select 0 - 7 control registers. If the address and attributes do not match in a single chip-select register, the MCF5307 will run an external bus cycle with external termination on a 32-bit port with burst-inhibited transfers. Should an address and attribute match in multiple chip-select registers, the matching chip-select signals will be driven; however, the MCF5307 will run an external bus cycle with external termination on a 32-bit port with burst-inhibited transfers. Table 9-2 shows the type of access depending on what matches are made in the CS control registers.

Table 9-2. Accesses by Matches in CS Control Registers

NUMBER OF CHIP SELECTS REGISTER MATCHES	TYPE OF ACCESS
None	External ¹
Single	As defined by chip select control register
Multiple	External ^{1,2}

1. For the cases represented in the table, EXTERNAL refers to an external bus cycle with external termination on a 32-bit port with burst-inhibited transfers
2. For the case of multiple chip selects matching, all of the matching chip selects will be asserted even though external termination will be required as defined in NOTE 1.

9.3.1.1.1 8-, 16-, and 32-Bit Port Sizing. The general-purpose chip-selects support static bus sizing. You can program the size of the port controlled by a chip-select. Defined 8-bit ports are connected to data bus bits 31:24, defined 16-bit ports are connected to data bus bits 31:16, and defined 32-bit ports are connected to data bus bits 31:0. The port size is specified by the PS bits in the CSCR. Figure 9-1 illustrates the byte lanes that external chip select memory should be connected to.

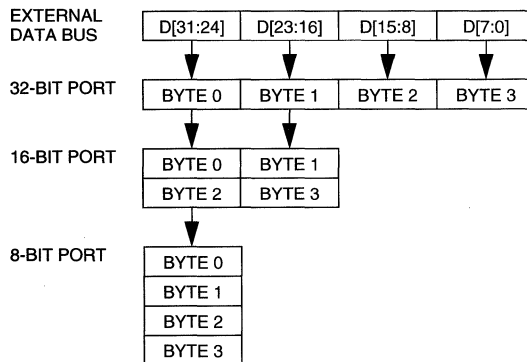


Figure 9-1. Connections for External Memory Port Sizes

9

9.3.2 Global Chip-Select Operation

$\overline{CS0}$ is the global (boot) chip-select and as such, allows address decoding for boot ROM before system initialization occurs. Its operation differs from the other external chip-select outputs following a system reset.

After system reset, $\overline{CS0}$ is asserted for every external access. Internal accesses can be made to go external by setting IARBCTRL bit of the MPARK register in the SIM. Refer to the **Default Bus Master Register (MPARK)** section for more information. No other chip-select should be used while $\overline{CS0}$ is a global chip-select. $\overline{CS0}$ operates in this manner until the valid bit is set in CSMR0, at which point $\overline{CS7}$ - $\overline{CS1}$ may 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 the D[7:5] signals.

The reset values of AA, PS1, and PS0 in CSCR0 are determined by the logic levels driven on D[7:5] at the rising edge of \overline{XRSTI} . Table 9-3 shows how the logic levels of D[7:5] correspond to the port sizes and enable/disable auto acknowledge for $\overline{CS0}$.

Table 9-3. D[6] and D[5] Selection of $\overline{CS0}$ Port Size

D[6]	D[5]	BOOT $\overline{CS0}$ PORT SIZE
0	0	32-bit port
0	1	8-bit port
1	0	16-bit port
1	1	16-bit port

Table 9-4. D[7] Selection of $\overline{CS0}$ Automatic Acknowledge

D[7]	BOOT $\overline{CS0}$ AA
0	Disabled
1	Enabled with 15 wait states

Provided the required address range is first loaded into CSAR, $\overline{CS0}$ can be programmed to continue to decode for a range of addresses after the V-bit is set. After the V-bit is set for $\overline{CS0}$, global chip-select can be restored only with another system reset.

9.4 PROGRAMMING MODEL

9.4.1 Chip-Select Registers Memory Map

Table 9-5 below shows the memory map of all the chip-select registers.

Table 9-5. Memory Map of Chip-Select Registers

ADDRESS	NAME	WIDTH	DESCRIPTION	RESET VALUE ²	ACCESS ³
MBAR+\$080	CSAR0	16	Chip-Select Address Register - Bank 0	uninitialized	R/W
MBAR+\$084	CSMR0	32	Chip-Select Mask Register - Bank 0	uninitialized (except V=0)	R/W
MBAR+\$08A	CSCR0	16	Chip-Select Control Register - Bank 0	BEM=1; BSTR=BSTW=0; AA=D[7]; PS1=D[6]; PS0=D[5]; WS3=WS2=WS1=WS0=1	R/W
MBAR+\$08C	CSAR1	16	Chip-Select Address Register - Bank 1	uninitialized	R/W
MBAR+\$090	CSMR1	32	Chip-Select Mask Register - Bank 1	uninitialized (except V=0)	R/W
MBAR+\$096	CSCR1	16	Chip-Select Control Register - Bank 1	uninitialized (except BEM=BSTR=BSTW=0)	R/W
MBAR+\$098	CSBAR	8	Chip-Select Base Address Register - Banks 2-7	uninitialized	R/W
MBAR+\$9C	CSMR2	16	Chip-Select Mask Register - Bank 2	uninitialized (except V=0)	R/W
MBAR+\$0A2	CSCR2	16	Chip-Select Control Register - Bank 2	uninitialized (except BEM=BSTR=BSTW=0)	R/W
MBAR+\$0A4	--	32	Reserved ¹	--	--
MBAR+\$AA	CSMR3	16	Chip-Select Mask Register - Bank 3	uninitialized (except V=0)	R/W
MBAR+\$0AE	CSCR3	16	Chip-Select Control Register - Bank 3	uninitialized (except BEM=BSTR=BSTW=0)	R/W
MBAR+\$0B0	--	32	Reserved ¹	--	--
MBAR+\$0B6	CSMR4	16	Chip-Select Mask Register - Bank 4	uninitialized (except V=0)	R/W
MBAR+\$0BA	CSCR4	16	Chip-Select Control Register - Bank 4	uninitialized (except BEM=BSTR=BSTW=0)	R/W
MBAR+\$0BC	--	32	Reserved ¹	--	--
MBAR+\$0C2	CSMR5	16	Chip-Select Mask Register - Bank 5	uninitialized (except V=0)	R/W
MBAR+\$0C6	CSCR5	16	Chip-Select Control Register - Bank 5	uninitialized (except BEM=BSTR=BSTW=0)	R/W
MBAR+\$0C8	--	32	Reserved ¹	--	--
MBAR+\$0CE	CSMR6	16	Chip-Select Mask Register - Bank 6	uninitialized (except V=0)	R/W
MBAR+\$0D2	CSCR6	16	Chip-Select Control Register - Bank 6	uninitialized (except BEM=BSTR=BSTW=0)	R/W
MBAR+\$0D4	--	32	Reserved ¹	--	--
MBAR+\$0DA	CSMR7	16	Chip-Select Mask Register - Bank 7	uninitialized (except V=0)	R/W

Table 9-5. Memory Map of Chip-Select Registers (Continued)

ADDRESS	NAME	WIDTH	DESCRIPTION	RESET VALUE ²	ACCESS ³
MBAR+\$0DE	CSCR7	16	Chip-Select Control Register - Bank 7	uninitialized (except BEM=BSTR=BSTW=0)	R/W

1. Addresses not assigned to a register and undefined register bits are reserved for future expansion. Write accesses to these reserved address spaces and reserved register bits have no effect.
2. The reset value column indicates the register initial value at reset. Certain registers may be uninitialized upon reset, i.e., they may contain random values.
3. The access column indicates whether the corresponding register allows both read/write functionality (R/W), read-only functionality (R), or write-only functionality (W). A read access to a write-only register will return zeros. A write access to a read-only register will have no effect.

9.4.2 Chip-Select Module Registers

9.4.2.1 CHIP-SELECT ADDRESS REGISTER (CSAR0, CSAR1 AND CSBAR). Each CSAR and CSBAR determines the base address of the corresponding chip-select pin. CSAR0 and CSAR1 determine the base addresses from which chip-selects 0 and 1 will be offset, respectively. CSBAR determines the base address from which chip-selects 2 through 7 will be offset.

- CSAR0 and CSAR1 are 16-bit read/write registers. CSBAR is a 8-bit read/write register. The value stored in each CSAR register corresponds to A[31:16]. The value stored in the CSBAR register corresponds to A[31:24].
- CSAR0, CSAR1 and CSBAR are uninitialized by reset

Chip Select Address Register for $\overline{CS0}$ - $\overline{CS1}$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA31	BA30	BA29	BA28	BA27	BA26	BA25	BA24	BA23	BA22	BA21	BA20	BA19	BA18	BA17	BA16

RESET: - - - - -

BA31-BA16 - Base Address

This field defines the base address location of memory dedicated to chip select $\overline{CS0}$ and $\overline{CS1}$. These bits are compared to bits 31-16 on the internal core address bus to determine if the chip-select memory is being accessed.

Chip Select Base Address Register(CSBAR) for $\overline{CS2}$ - $\overline{CS7}$

7	6	5	4	3	2	1	0
BA31	BA30	BA29	BA28	BA27	BA26	BA25	BA24

RESET: - - - - -

CHIP-SELECT (CS) MODULE

BA31-BA24 - Base Address

This field defines the base address location of memory dedicated to chip-selects $\overline{CS2}$ through $\overline{CS7}$. These bits are compared to bits 31-24 on the internal core address bus to determine if the chip-select memory is being accessed.

For chip-selects 2-7, only the upper 8 bits are implemented (in CSBAR) allowing for memory block sizes of 2M for each chip-select. For chip-selects 2-7, the offset from CSBAR is decoded from bits 23-21 on the internal core address bus to determine which chip-select signal is being used. See Table 9-6 for the chip-select decoding.

For example, if \$F0 is written into the Chip-Select Base Register (CSBAR), chip-select 5 will be asserted when the address range is from \$F0A00000 to \$F0BFFFFF. Chip-select 6 will be asserted from address range \$F0C00000 to \$F0DFFFFF.

Table 9-6. Chip-Select 2-7 Decoding

A23	A22	A21	CHIP SELECT	MAXIMUM BLOCK SIZE
0	0	0	0	* 2 Megabytes
0	0	1	1	* 2 Megabytes
0	1	0	2	2 Megabytes
0	1	1	3	2 Megabytes
1	0	0	4	2 Megabytes
1	0	1	5	2 Megabytes
1	1	0	6	2 Megabytes
1	1	1	7	2 Megabytes

* This only applies when the parallel port, PP[15:8], signals are selected on the external pins and an external master requires access to the chip select resources. In this case, the internal address bus bits, A[31:24] are all 0. To avoid overlap of chip select space, CSBAR should be set to \$0, CSAR0 should be set to \$0 and CSAR1 should be set to \$0020. CSMR0 and CSMR1 should be set with the appropriate mask bits such that the maximum space is 2M.

9.4.2.2 CHIP-SELECT MASK REGISTER (CSMR0 - CSMR7). CSMR0 and CSMR1 determine the address mask for CS[0:1], respectively. Also, CSMR[7:0] determines the definition of which types of accesses are allowed for these signals. Each CSMR is a 32-bit read/write control register that physically resides in the Chip-Select Module. CSMR7 - CSMR0 are uninitialized by reset, except for bit 0 (V-bit) which is initialized to 0. Note that CPU or IACK space masking is only possible in $\overline{CS0}$, $\overline{CS1}$ or $\overline{CS7}$. For an IACK cycle to occur in $\overline{CS7}$, CSBAR would have to be set to \$FF.

Chip Select Mask Register(CSMR0-CSMR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BAM31	BAM30	BAM29	BAM28	BAM27	BAM26	BAM25	BAM24	BAM23	BAM22	BAM21	BAM20	BAM19	BAM18	BAM17	BAM16
RESET:															
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	WP	-	AM	C/I	SC	SD	UC	UD	V
RESET:															
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

Chip Select Mask Register(CSMR2-CSMR7)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	WP	-	AM	C/I*	SC	SD	UC	UD	V
RESET:															
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

* This bit (C/I) is not implemented in registers CSMR2 - CSMR6; it is only implemented in registers CSMR0, CSMR1 and CSMR2.

BAM31-BAM16 - Base Address Mask

This field defines the chip-select block size through the use of address mask bits. Any set bit masks the corresponding base address register (CSAR) bit (the base address bit becomes a don't care 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 $\overline{CS}[0:1]$ is equal to 2^n , where n =(number of bits set in the base address mask field of the respective CSMR)+16. The block size for $\overline{CS}[2:7]$ is equal to 2MB in all cases.

For example, if CSAR0 was set at \$0000 and CSMR0 was set at \$0008, then chip select \overline{CS}_0 would address two 64K spaces: from \$00000000 to \$0000FFFF and from \$00080000 to \$0008FFFF.

WP, AM, C/I, SC, SD, UC, UD - Write Protect, Alternate Master, CPU/IACK, Supervisor Code, Supervisor Data, User Code, User Data Address Space Mask

1. This field masks specific address spaces, placing the chip-select in a specific address space or spaces. If an address space mask bit is cleared, an access to a location in that address space can activate the corresponding chip-select. If an address space

CHIP-SELECT (CS) MODULE

mask bit is set, an access to a location in that address space becomes a regular external bus access, and no chip-select is activated.

For each address space mask bit (AM, C/I, SC, SD, UC, UD):

- 0 = Do not mask this address space for the chip select. An access using the chip select can occur for this address space.
- 1 = Mask this address space from the chip select activation. If this address space is accessed, no chip select activation will occur on the external cycle.

The address space mask bits are:

WP= Write Protect

The WP bit can restrict write accesses to the address range in a CSAR. An attempt to write to the range of addresses specified in a CSAR that has this bit set will result in the appropriate chip-select not being selected.

- 1 = Only read accesses are allowed
- 0 = Either read or write accesses are allowed

AM= Alternate Master mask

When AM=0 and an alternate master access occurs, SC, SD, UC, and UD are “don’t cares” in the chip-select decode. Refer to **Bus Arbitration** in **Section 7 Bus Operations** section for information on alternate bus masters.

- C/I = CPU space and Interrupt Acknowledge Cycle mask ($\overline{CS0}$, $\overline{CS1}$, and $\overline{CS7}$)
- SC = Supervisor Code address space mask
- SD = Supervisor Data address space mask
- UC = User Code address space mask
- UD = User Data address space mask

NOTE

Even if the C/I bit is set in CSMR[6:2] they will not assert for CPU space and IACK cycles. These chip selects will assert for normal accesses only.

V - Valid bit

The Valid bit indicates that the contents of its address register, mask register, and control register are valid. The programmed chip selects do not assert until the V-bit is set (except for $\overline{CS0}$ which acts as the global (boot) chip select - see 9.3.2 Global Chip-Select Operation.) A reset clears the V-bit in each CSMR.

- 0 = Chip-select invalid
- 1 = Chip-select valid

9.4.2.3 CHIP-SELECT CONTROL REGISTER (CSCR0 - CSCR7). Each CSCR controls the auto acknowledge, external master support, port size, burst capability, and activation of each of the chip-selects.

For CSCR1 - CSCR7, bits BSTR, BSTW, and V are initialized to 0 by reset while all other bits are uninitialized by reset. For CSCR0, bits BSTR, BSTW, and V are initialized to 0 by reset while bits WS[3:0] are initialized to 1 by reset.

$\overline{CS0}$ is the global (boot) chip-select and as such, allows address decoding for boot ROM before system initialization occurs. Its operation differs from the other external chip-select outputs following a system reset. Refer to **9.3.2 Global Chip-Select Operation** for more information concerning the global chip-select.

Chip Select Control Register(CSCR0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	WS3	WS2	WS1	WS0	-	AA	PS1	PS0	BEM	BSTR	BSTW	-	-	-
RESET:	-	1	1	1	1	-	D[7]	D[6]	D[5]	1	0	0	-	-	-

Chip Select Control Register(CSCR1-7)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	WS3	WS2	WS1	WS0	-	AA	PS1	PS0	BEM	BSTR	BSTW	-	-	-
RESET:	-	-	-	-	-	-	-	-	-	0	0	0	-	-	-

WS[3:0] - Wait States

This field defines the number of wait states that will be inserted before an internal transfer acknowledge is generated. If the AA bit is set to 0, \overline{TA} is asserted by the external system regardless of the number of wait states generated. In that case the external transfer acknowledge will end the cycle.

BSTR - Burst Read Enable

This field specifies the read burst capability of the memory associated with each chip-select. If BSTR=1, all reads from port sizes smaller than the requested transfer size will be bursted, including longword reads from 8 and 16-bit ports, word reads from 8-bit ports as well as line reads from 8-, 16-, and 32-bit ports. If BSTR=0, all reads from port sizes smaller than the requested transfer size will be broken into individual reads that are no larger than the specified port size. For example, a longword read from an 8-bit port would be broken into four individual byte reads.

- 0 = Break all reads that are larger than the specified port size into individual nonburst reads that are no larger than the specified port size
- 1 = Allow burst read by chip-selected address space for all reads that are larger than the specified port size

BSTW - Burst Write Enable

This field specifies the write burst capability of the memory associated with each chip-select. If $BSTW=1$, all writes to port sizes smaller than the requested transfer size will be bursted, including longword writes to 8 and 16-bit ports, word writes to 8-bit ports as well as line writes to 8-, 16-, and 32-bit ports. If $BSTW=0$, all writes to port sizes smaller than the requested transfer size will be broken into individual writes that are no larger than the specified port size. For example, a longword write to an 8-bit port would be broken into four individual byte writes.

- 0 = Break all writes that are larger than the specified port size into individual nonburst writes that are no larger than the specified port size
- 1 = Allow burst write to chip-selected address space for all writes that are larger than the specified port size

AA - Auto-Acknowledge Enable

This field controls the assertion of the internal transfer-acknowledge during all accesses that hit in the corresponding chip-select address space. If $AA=1$, the internal transfer-acknowledge will be asserted at the time determined by the value of $WS[3:0]$. If $AA=0$, the Chip-Select Module will not cause the internal transfer acknowledge to be asserted and the cycle will have to be terminated by the external system.

- 0 = Wait for external transfer acknowledge
- 1 = Wait for internal acknowledge specified by $WS[3:0]$

PS[1:0] - Port Size

This field specifies the width of the data associated with each chip-select. It determines where data will be driven during write cycles and where data will be sampled during read cycles.

- 00 = 32-bit port size - Data sampled and driven on $D[31:0]$
- 01 = 8-bit port size - Data sampled and driven on $D[31:24]$ only
- 10 = 16-bit port size - Data sampled and driven on $D[31:16]$ only
- 11 = 16-bit port size - Data sampled and driven on $D[31:16]$ only

BEM - Byte Enable Module

This field specifies the mode of functionality for byte enables. Certain SRAMs have byte enables that must be asserted during reads (in addition to writes.) The BEM bit may be set in the relevant CSCR to provide the appropriate mode of byte enable in support of these SRAMs. The default mode after reset is 0 for $\overline{CS7} - \overline{CS1}$ and 1 for $\overline{CS0}$.

- 1 = $\overline{BE}/\overline{BWE}$ signals generated for data reads and writes
- 0 = \overline{BWE} signals generated for data writes only

Note

Even if selected, \overline{BE} is not asserted for writes.

9.4.2.4 CODE EXAMPLE. The code below provides an example of how to initialize the chip-selects. MBARx defines the base of the module address space

```

CSAR0 EQU MBARx+$080 ;Chip Select 0 address register
CSMR0 EQU MBARx+$084 ;Chip Select 0 mask register
CSCR0 EQU MBARx+$08A ;Chip Select 0 control register

CSAR1 EQU MBARx+$08C ;Chip Select 1 address register
CSMR1 EQU MBARx+$090 ;Chip Select 1 mask register
CSCR1 EQU MBARx+$096 ;Chip Select 1 control register

CSBAR EQU MBARx+$098 ;Chip Select 2-7 base address register

CSMR2 EQU MBARx+$09C ;Chip Select 2 mask register
CSCR2 EQU MBARx+$0A2 ;Chip Select 2 control register

CSMR3 EQU MBARx+$0A8 ;Chip Select 3 mask register
CSCR3 EQU MBARx+$0AE ;Chip Select 3 control register

CSMR4 EQU MBARx+$0B4 ;Chip Select 4 mask register
CSCR4 EQU MBARx+$0BA ;Chip Select 4 control register

CSMR5 EQU MBARx+$0C0 ;Chip Select 5 mask register
CSCR5 EQU MBARx+$0C6 ;Chip Select 5 control register

CSMR6 EQU MBARx+$0CC ;Chip Select 6 mask register
CSCR6 EQU MBARx+$0D2 ;Chip Select 6 control register

CSMR7 EQU MBARx+$0D8 ;Chip Select 7 mask register
CSCR7 EQU MBARx+$0DE ;Chip Select 7 control register

    move.w  #$0000,D0      ;CSAR0 starts at address $00000000
    move.w  DO,CSAR0      ;it is usually best to set up the cscr0 before exiting
                          ;global chip select
    move.w  #$0C18,D0      ;3 wait states, AA = 0, PS = 32-bit
    move.w  DO,CSCR0      ;BEM = 0, BURSTREAD = 1, BURSTWRITE = 1; This will exit
                          ;global chip select mode
    move.l  #$00000001,D0  ;Addresses range from $00000000 to $0000FFFF
    move.l  DO,CSMR0      ; WP,AM,C/I,SC,SD,UC,UD =0; V = 1

    move.w  #$0001,D0      ;CSAR1 starts at address $00010000
    move.w  DO,CSAR0

    move.w  #$000F0001,D0  ;Addresses range from $00010000 to $000FFFFF
    move.w  DO,CSMR1      ; WP,AM,C/I,SC,SD,UC,UD =0; V = 1

    move.w  #$0C18,D0      ;3 wait states, AA = 0, PS = 32-bit
    move.w  DO,CSCR1      ;BEM = 0, BURSTREAD = 1, BURSTWRITE = 1

    move.w  #$0F00,D0      ;CSBAR addresses $0F000000
    move.w  DO,CSBAR      ;This is for Chip Selects 2-7

```


9.5 TIMING DIAGRAMS

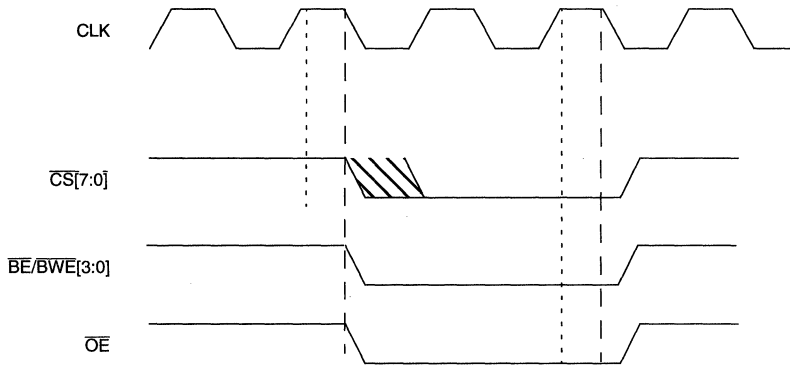


Figure 9-2. Chip-Select Module Outputs Timing Diagram

SECTION 10

PARALLEL PORT (GENERAL-PURPOSE I/O)

10.1 INTRODUCTION

This subsection describes the operation and programming model of the parallel port registers and the direction-control data registers.

10.2 SIGNAL DESCRIPTIONS

The MCF5307 parallel port module has 16 signals that you can select as inputs or outputs on a pin-by-pin basis. These pins are multiplexed with the upper address pins A[31:24], \overline{TIP} , \overline{DREQ} [1:0], and TM[2:0], and TT[1:0] pins. The Pin Assignment Register (PAR) defines how these pins function (see Table 10-1).

For example, if address pins A[25:0] were needed for external addressing, you can configure the remaining pins (PP[15:10]/A[31:26]) for general-purpose I/O. Note that PP[7:0] are multiplexed with the \overline{TIP} , \overline{DREQ} [1:0], TM[2:0], and TT[1:0]. PP[15:8] are multiplexed with A[31:24] (see Table 10-1).

Table 10-1 lists the signals associated with the PAR. The PAR only enables those pins that function as parallel I/O and has no other effect on the functionality of the parallel port (such as logic levels) or which pins are inputs or outputs.

Pin Assignment Register(PAR)																Address MBAR + \$04	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
PAR15	PAR14	PAR13	PAR12	PAR11	PAR10	PAR9	PAR8	PAR7	PAR6	PAR5	PAR4	PAR3	PAR2	PAR1	PAR0		
RESET: PP	RESET	SEL	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1		
0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1		

Table 10-1: Signals Selected with PAR

PAR	PAR=0	PAR=1
PAR15	PP[15]	ADDR[31]
PAR14	PP[14]	ADDR[30]
PAR13	PP[13]	ADDR[29]
PAR12	PP[12]	ADDR[28]
PAR11	PP[11]	ADDR[27]
PAR10	PP[10]	A[26]

Table 10-1: Signals Selected with PAR (Continued)

PAR	PAR=0	PAR=1
PAR9	PP[9]	A[25]
PAR8	PP[8]	A[24]
PAR7	PP[7]	$\overline{\text{TIP}}$
PAR6	PP[6]	$\overline{\text{DREQ}}[0]$
PAR5	PP[5]	$\overline{\text{DREQ}}[1]$
PAR4	PP[4]	TM[2]
PAR3	PP[3]	TM[1]
PAR2	PP[2]	TM[0]
PAR1	PP[1]	TT[1]
PAR0	PP[0]	TT[0]

(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 Pin Assignment Register (PAR) in the System Integration Module (SIM) determines the function of each of these eight multiplexed pins. You can program these on a bit-by-bit basis.

10

When any of these pins are enabled in the PAR (logic 1), they represent the most significant bits of the address bus. As much as 4 Gbytes of memory is available when all of these pins are programmed as address signals.

When any of these pins are disabled in the PAR (logic 0), they represent the most significant bits of the parallel port.

($\overline{\text{TIP}}$ /PP[7])

This multiplexed pin can serve as the transfer in progress output or as one bit of the parallel port. Programming the Pin Assignment Register (PAR) in the SIM determines the function of this pin.

This output is asserted to indicate that a bus transfer is in progress and is negated during idle bus cycles if the bus is still granted to the processor. Note that $\overline{\text{TIP}}$ is held asserted on back-to-back bus cycles.

DMA Request ($\overline{\text{DREQ}}[1:0]$ /PP[6:5])

These multiplexed pins can serve as the DMA request inputs or as two bits of the parallel port. Programming the Pin Assignment Register (PAR) in the SIM determines the function of each of these three multiplexed pins. The pins are programmable on a bit-by-bit basis.

These active-low inputs are asserted by a peripheral device to request an operand transfer between that peripheral and memory.

Transfer Modifier - (TM[2:0]/PP[4:2])

These multiplexed pins can serve as the transfer type outputs or as three bits of the parallel port. Programming the Pin Assignment Register (PAR) in the SIM determines the function of each of these three multiplexed pins, which are programmable on a bit-by-bit basis.

These input signals provide supplemental information for each transfer type (see Table 2-4, Transfer Modifier Encodings for TT=00). For emulation transfers, the TM signals indicate user or data transfer types (see Table 2-5, Transfer Modifier Encodings for TT=10). For CPU space transfers, the TM signals are low (see Table 2-6, Transfer Modifier Encodings for TT=11). For interrupt acknowledge transfers, the TM signals carry the interrupt level being acknowledged (see Table 2-6).

Transfer Type - (TT[1:0]/PP[1:0])

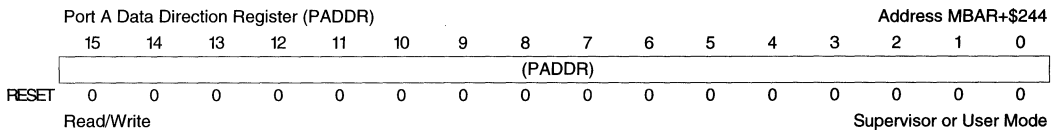
These multiplexed pins can serve as the transfer type outputs or as two bits of the parallel port. Programming the Pin Assignment Register (PAR) in the SIM determines the function of each of these two multiplexed pins. The pins are programmable on a bit-by-bit basis.

When the MCF5307 is the bus master, it outputs these signals. They indicate the type of access for the current bus access. Table 2-3 (Bus Cycle Transfer Type Encoding) shows the encoding definitions.

10.3 PARALLEL PORT OPERATION

10.3.1 PORT A DATA DIRECTION REGISTER (PADDR)

The PADDR allows you to select the signal direction of each parallel port signal. The PADDR register contains one data direction bit for each parallel port signal. The data direction control bits will only affect the direction of the associated pin if you program that pin as a general-purpose I/O signal.



PADDR[15:0] - Data Direction Bits [15:0]

For each of the data direction bits, you can select the direction of the signal as follows:

- 0 = Signal is defined as an input
- 1 = Signal is defined as an output

10.3.1.1 PORT A DATA REGISTER (PADAT). The parallel port data register reflects the current status of the parallel port signals. If you configure a parallel port as an input, the value in the register corresponds to the logical voltage present at the pin. If you configure the parallel port signal as an output, the value in the register corresponds to the logical level driven onto the pin.

Port A Data Register (PADAT)														Address MBAR+\$248		
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	(PADAT)															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Read or Write												Supervisor or User Mode			

- Bits in the PADAT are valid for the pins configured as general-purpose I/O only. If you configure a pin to be an output, the logic level shown in the register will be the logic level on the corresponding pin.
- The PADAT has no effect on a pin that is not configured as a general-purpose I/O.
- You can write to the PADAT register at any time. A write to a bit corresponding to an input signal will seemingly have no effect. However, if a pin changes from an input to an output, the value most recently written into the PADAT will be the value driven.

PADAT[15:0] - Parallel Port Data Register Bits[15:0]

10

- For parallel port signals programmed to outputs:
 - For PADAT read: register bit indicates logical voltage level at the pin
 - For PADAT write: drive indicated logical voltage level onto associated pin
- For parallel port signals programmed to inputs:
 - For PADAT read: register bit indicates current logical voltage level at the pin
 - For PADAT write: has no effect unless pin direction is changed to output

10.3.1.2 EXAMPLE CODE. The code example that follows demonstrates how to set up the parallel port. This example sets PP[7:0] as general-purpose I/O. PP[3:0] are defined as inputs and PP[7:4] are defined as outputs. The example assumes that MBAR was defined at location \$00010000.

```
MBARx EQU $00010000
PAR EQU MBARx+$04
PADDR EQU MBARx+$244
PADAT EQU MBARx+$248

move.l #MBARx,D0
movec D0,MBAR ;since MBAR is an internal register, MBARx
;was used as variable for the memory map

move.w #$00FF,D0
move.w D0,PAR ;set up the PAR. PP[7:0] set up as I/O
move.w #$00F0,D0
move.w D0,PADDR ;set PP[7:4] as outputs;PP[3:0] as inputs
move.b #$A0,D0
move.b D0,PADAT ;set PP pin 7 and 5, which are outputs, to a
;logic '1'. PP 6 and 4 are logic '0'
```


SECTION 11

SYNCHRONOUS/ASYNCHRONOUS DRAM CONTROLLER MODULE

11.1 INTRODUCTION

The Synchronous/Asynchronous DRAM Controller (SADRAMC) module provides for glueless integration of DRAM with the ColdFire product.

11.2 FEATURES

The key features of the DRAM controller include:

- Support for 2 banks of DRAM
- Support for External Masters
- Programmable Wait States and Refresh Timer
- Support for Page Mode
- Support for 8-, 16-, and 32-bit wide DRAM banks
- Support for EDO DRAMs
- Support for synchronous DRAMs

11.3 BLOCK DIAGRAM

The basic blocks of the DRAM Controller are shown in Figure 11-1. A brief description of all the blocks follows.

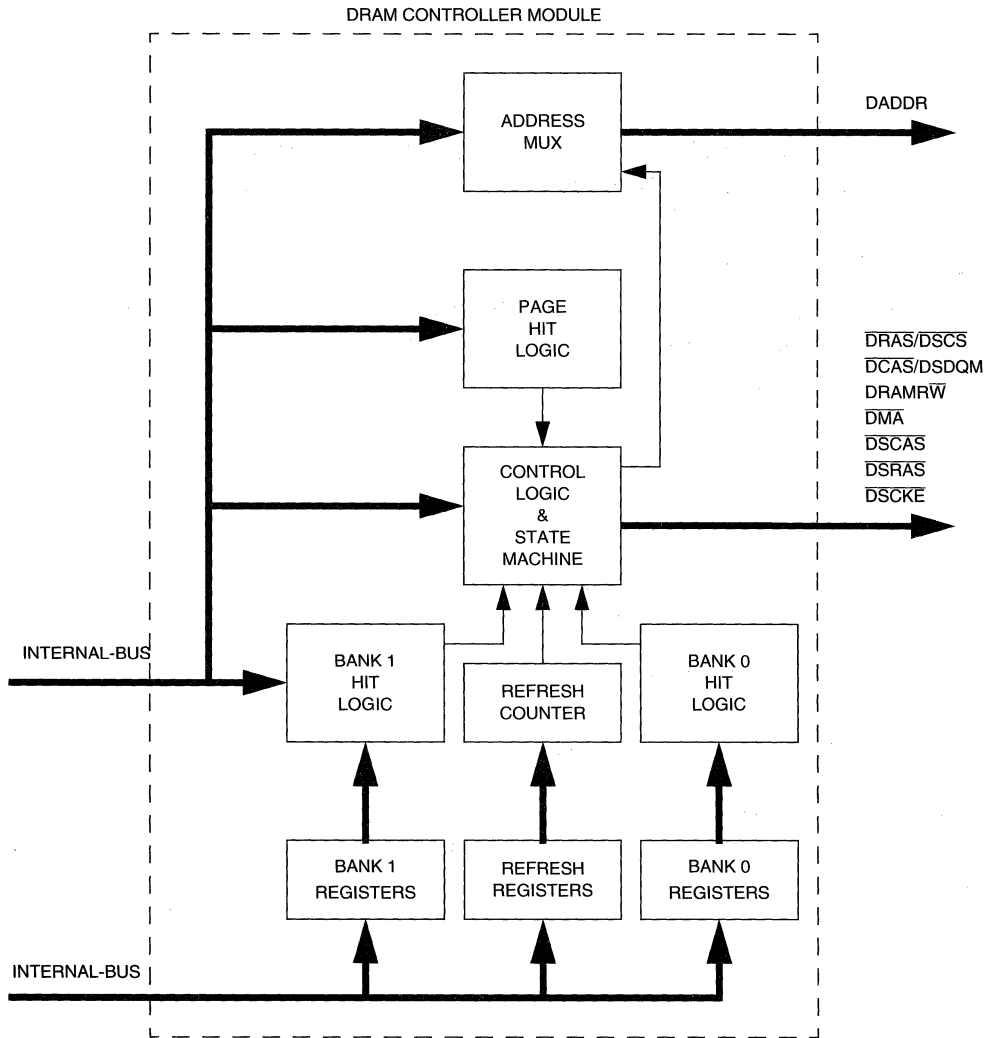


Figure 11-1. DRAM Controller Block Diagram Bank Register Blocks

The SADRAMC consists of two banks, each of which contains an associated Register Block. The registers are accessed as memory locations. The register information is passed on to the Hit Logic.

11.3.1 Refresh Register Block

This block contains data to control refresh operation of the DRAM Controller. All refresh for all banks will be controlled by the same register bits and refreshed concurrently.

11.3.2 Refresh Counter

This block contains the counter that determines when refresh should take place. It generates a refresh request to the control block

11.3.3 Hit Logic

The Hit Logic is responsible for comparing the address and attribute signals from the requested location to the register information to determine if a DRAM bank is being accessed. If a match occurs, the Hit Logic passes this hit to the Control Logic block along with the characteristics of the bus cycle to be generated. This is or'ed at the Control Logic with information from the other bank to determine what the bus cycle characteristics are.

11.3.4 Control Logic and State Machine

This block is responsible for generating \overline{RAS} , \overline{CAS} , and DRAM/ \overline{RW} signals. It takes the bus cycle characteristic data from the bank logic, along with hit information to generate accesses to DRAM. It is also responsible for taking refresh requests from the refresh counter to generate \overline{CAS} before \overline{RAS} refresh.

11.3.5 Page Hit Logic

This block is responsible for determining if the next access is within the same DRAM page as the previous access. This information is passed on to the control logic.

11.3.6 Address Mux

This block multiplexes the addresses to allow column addresses to be driven on the same pins as the row addresses. This allows glueless interface to DRAMs.

11.4 SIGNAL LIST

The following DRAM signals provide a seamless interface to external DRAM. DRAM widths of 8-, 16-, and 32- bits are supported and can access as much as 256 MBytes of memory.

11.4.1 Row Address Strokes - ($\overline{RAS}[1:0]$)

These active-low pins provide a seamless interface to Row Address Strobe inputs on industry standard DRAMs. These pins provide RAS for a given DRAM bank. Banks correspond to specific Base Address and Control information programmed in the DCM registers (see **Section 11 DRAM Controller** for a description). $\overline{RAS}[0]$ corresponds to bank 0 and $\overline{RAS}[1]$ corresponds to Bank 1.

11.4.2 Column Address Strobes - ($\overline{\text{CAS}}[3:0]$)

These active-low pins provide a seamless interface to Column Address Strobe inputs on industry standard DRAMs. These provide CAS for a given DRAM bank. $\overline{\text{CAS}}[3:0]$ controls access to the most significant to least significant byte of data, respectively.

11.4.3 DRAM Read/Write - ($\overline{\text{DRAMW}}$)

This active-low pin is asserted to signify that a DRAM write cycle is underway.

11.4.4 Synchronous DRAM Column Address Strobe - ($\overline{\text{SCAS}}$)

This active-low output signal is registered during synchronous mode to route directly to the $\overline{\text{SCAS}}$ signal of SDRAMs.

11.4.5 Synchronous DRAM Row Address Strobe - ($\overline{\text{SRAS}}$)

This active-low output signal is registered during synchronous mode to route directly to the $\overline{\text{SRAS}}$ signal of external SDRAMs.

11.4.6 Synchronous DRAM Clock Enable - (SCKE)

This active high output signal is registered during synchronous mode to route directly to the SCKE signal of external SDRAMs.

11.4.7 Synchronous Edge Select - (EDGESEL)

This input signal helps select additional output hold time for signals that interface to external SDRAMs.

This signal can provide three modes of operation for the SDRAM interface signals. When this signal is tied high, the interface signals to the SDRAM will change on the rising edge of the clock (BCLKO). When this signal is tied low, the interface signals to the SDRAM will change on the falling edge of the clock (BCLKO). When this signal is tied to the SDRAM clock, the interface signals to the SDRAM will change on the rising edge of the SDRAM clock. The various configurations of this signal can provide additional output hold time for the interface signals provided to the SDRAM.

11.4.7.1 Edge Select Output Cell. While the MCF5307 supports SDRAM with a glueless interface, as shown in Figure 11-2, a special I/O module has been developed to ease the system level timings. This output interface cell monitors the clock at the input to the SDRAM and holds data until the next edge of the bus clock, adding required output hold time to the address, data and control signals. The buffer shown is optional, but may be required in some instances where additional clock delay is needed.

To generate the needed timing required to interface to SDRAMs, the address, data and control signals required for the SDRAM interface are clocked through a two stage shift register. The first stage is clocked on the rising edge of the bus clock (BCLKO) and the second stage is clocked on the falling edge. This method allows the signal to be available for up to an additional half bus clock cycle, as shown in Figure 11-3, of which only a small

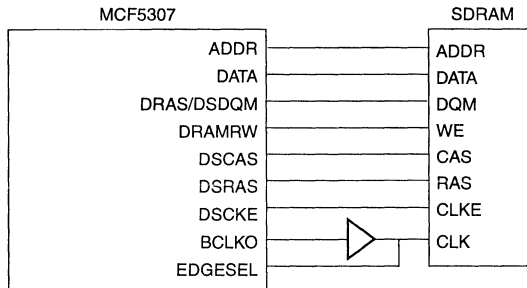


Figure 11-2. MCF5307 SDRAM Interface

percentage is deemed to insure proper timing is met. A feedback path was added to control which of the flipflops is driving the external pin. Using the clock signal feeding the SDRAM as the feedback signal insures that data is held until after the falling edge of the SDRAM clock input signal by a minimum specified by spec B16, EDGESEL to Invalid (hold). This also insures minimum delay unchanging to valid data or the next SDRAM clock cycle.

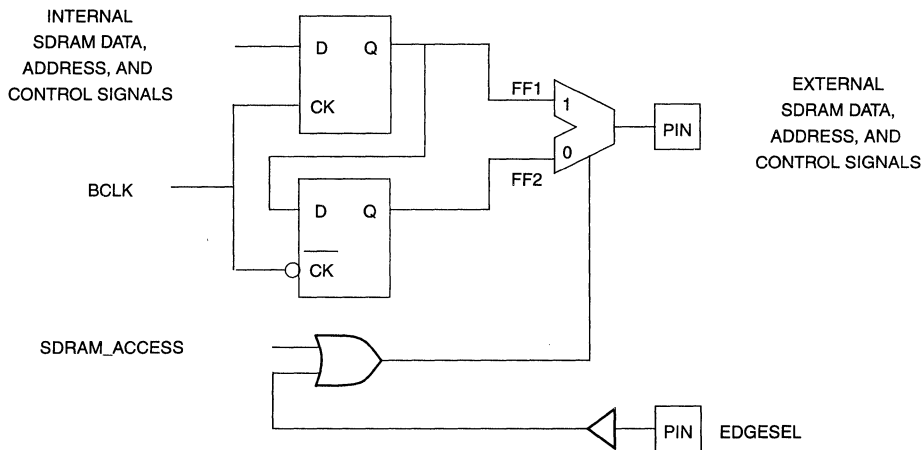


Figure 11-3. Edge-Select Cell

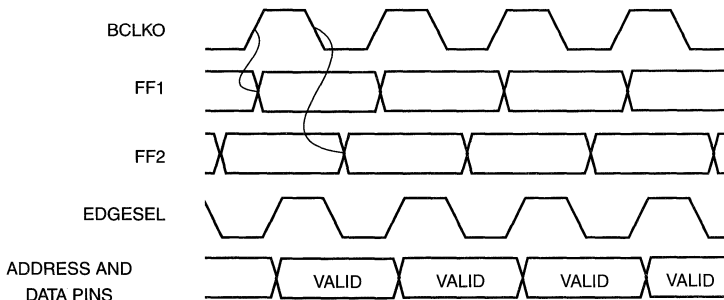


Figure 11-4. Edge-Select Cell Output Waveforms

11.5 ASYNCHRONOUS OPERATION

The SADRAMC has two major modes of operation. One is asynchronous operation which includes Page mode and EDO mode. The other is synchronous mode, which is designed to work with industry standard SDRAMs. These two modes act very different, and cause different use of the memory map and the pins. Both banks of the SADRAMC will be in the same mode of operation based on the SO bit in the DCR.

11.5.1 Asynchronous Memory Map

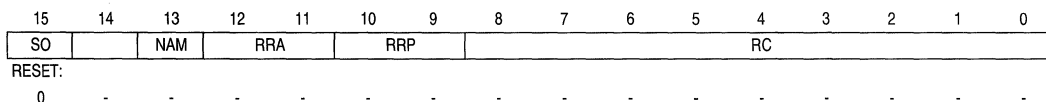
The SADRAMC memory map is shown in Table 11-1. The SADRAMC is responsible only for decoding the offset from the Module Enable.

Table 11-1. DRAM Controller Memory Map

ADDRESS	NAME	BYTE0	BYTE1	BYTE2	BYTE3
Base + \$100	DRAMC Control Register	DCR		Reserved	
Base + \$104	Reserved	--			
Base + \$108	Address & Control Bank 0	DACR0			
Base + \$10C	Mask Register Bank 0	DMR0			
Base + \$110	Address & Control Bank 1	DACR1			
Base + \$114	Mask Register Bank 1	DMR1			

11.5.1.1 DRAM Control Register -DCR.

MBAR + \$100



This register 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.

11.5.1.1.1 Synchronous Operation - SO

This bit determines if the SADRAMC is in synchronous operation mode. For asynchronous DRAMs, this bit must be set to zero. Note that once the SADRAMC has entered synchronous operation, it cannot be returned to asynchronous operation except through a reset of the MCF5307. See subsection 11.6.1.1

11.5.1.1.2 No Address Multiplexing - NAM

Some implementations will require external multiplexing support. For instance, if there is an external master accessing the DRAM or if a linear addressing scheme is required. In these cases, it would be advantageous to prevent the SADRAMC from multiplexing the addresses on a DRAM access. If this bit is set, the SADRAMC will not multiplex the external address bus to provide column addresses.

11.5.1.1.3 Refresh RAS Asserted - RRA[1:0]

These bits determine how long the RAS signal is asserted during a refresh operation. Table 11-2 shows the encoding of RRA.

Table 11-2. RRA Encoding

RRA[1:0]	RAS ASSERTED
00	2 Clocks
01	3 Clocks
10	4 Clocks
11	5 Clocks

11.5.1.1.4 Refresh RAS Precharged - RRP[1:0]

This field controls how many clocks the RAS signal is precharged after a refresh operation before accesses are allowed to DRAM. Table 11-3 shows the encoding for RRP.

Table 11-3. RRP Encoding

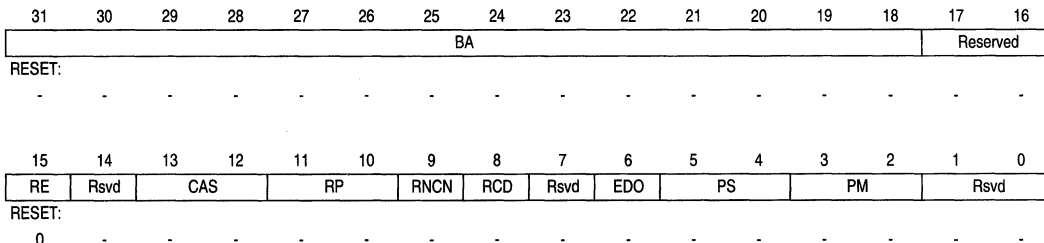
RRP[1:0]	RAS PRECHARGED
00	1 Clock
01	2 Clocks
10	3 Clocks
11	4 Clocks

11.5.1.1.5 Refresh Count - RC[8:0]

This field controls the frequency of refresh performed by the SDRAMC. One is added to the value stored in these register locations and multiplied by 16 bus clocks to determine the refresh period. This allows the refresh period to range from 16 to 8192 bus clocks. This will accommodate both standard and low-power DRAMs with bus clock operation from less than 2MHz to greater than 50MHz.

11.5.1.2 Address & Control Registers - DACR0 & DACR1

MBAR+\$108, 110.



These registers contain the base address compare value and the control bits for both banks 0 and 1 of the DRAM controller. Address and timing are also controlled by bits in the DACRx registers. Care must be taken to ensure that the memory areas defined for each bank are unique and do not overlap. In the event the two banks should have a common memory area, operation is undefined for accesses in the overlapping region.

11

11.5.1.2.1 Base Address Register - BA[31:18]

These register bits are used in conjunction with the BAM bits in the DCMR to determine the address range in which the associated bank of DRAM will be located. Each bit in the Base Address Register 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 bank's DRAM block.

11.5.1.2.2 Refresh Enable - RE

This bit determines if the SDRAMC will generate a refresh cycle to the associated DRAM bank. Table 11-4 shows the encoding of the RE bit. This bit is reset to zero to ensure that the refresh function is disabled at reset. The contents of DRAM are not preserved during RESET.

Table 11-4. RE Encoding

RE	FUNCTION
0	Do Not Refresh Associated DRAM Bank
1	Refresh Associated DRAM Bank

11.5.1.2.3 Column Address Strobe Timing - CAS[1:0]

These bits determine how long the $\overline{\text{CAS}}$ signal is asserted during a DRAM access. Table 11-5 shows the encoding of the CAS bits.

Table 11-5. CAS Encoding

CAS[1:0]	# OF SYSTEM CLOCKS $\overline{\text{CAS}}$ ACTIVE
00	1
01	2
10	3
11	4

11.5.1.2.4 Row Address Strobe Precharge Timing - RP[1:0]

These two bits determine how long the $\overline{\text{RAS}}$ signal is precharged between accesses. Table 11-6 shows the encoding of the RP bits.

Table 11-6. $\overline{\text{RAS}}$ Precharge (RP) Encoding

RP[1:0]	# OF SYSTEM CLOCKS $\overline{\text{RAS}}$ PRECHARGED
00	1
01	2
10	3
11	4

11

11.5.1.2.5 $\overline{\text{RAS}}$ Negate to $\overline{\text{CAS}}$ Negate - RNCN

This signal controls whether $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ are negated concurrently or one clock apart. Encoding of the RNCN is described in Table 11-7. This bit is ignored if $\overline{\text{CAS}}$ is asserted for only one clock. In this case $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ will be negated concurrently. This bit is only used for non-page mode accesses and single accesses in page mode.

Table 11-7. RNCN Encoding

RNCN	FUNCTION
0	$\overline{\text{RAS}}$ Negated Concurrently with $\overline{\text{CAS}}$
1	$\overline{\text{RAS}}$ Negated One Clock Before $\overline{\text{CAS}}$

11.5.1.2.6 $\overline{\text{RAS}}$ to $\overline{\text{CAS}}$ Delay - RCD

This bit determines the number of system clocks between the assertion of $\overline{\text{RAS}}$ and the assertion of $\overline{\text{CAS}}$. Operation of the RCD bit is shown in Table 11-8.

Table 11-8. RCD Encoding

RCD	# OF SYSTEM CLOCKS BETWEEN $\overline{\text{RAS}}$ AND $\overline{\text{CAS}}$
0	1
1	2

11.5.1.2.7 Extended Data Out - EDO

This register bit determines if the associated bank of DRAM operates in a mode to take advantage of industry standard EDO DRAMs. The EDO register encoding is shown below in Table 11-9.

Table 11-9. EDO Encoding

EDO	FUNCTION
0	EDO Operation Disabled
1	EDO Operation Enabled

11.5.1.2.8 Port Size - PS

These two bits will determine the port size of the associated bank of DRAM. This will allow for dynamic sizing of the associated accesses. Table 11-10 shows the encoding of the PS bits. Note that both 11 and 10 decode to a 16-bit port.

Table 11-10. PS Encoding

PS[1:0]	PORT SIZE
00	32-Bit Port
01	8-Bit Port
10	16-Bit Port
11	16-Bit Port

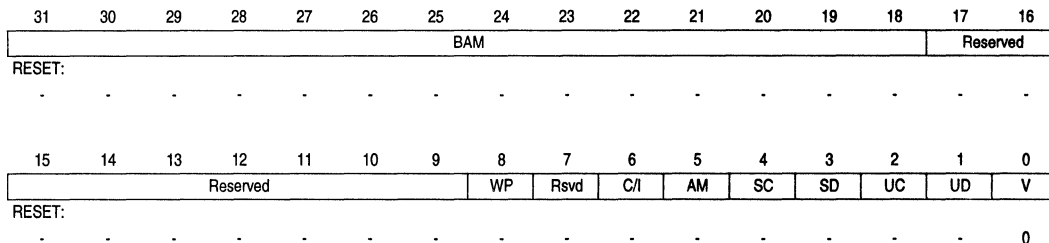
11.5.1.2.9 Page Mode - PM

These two bits determine if the DRAM bank supports page mode operation, and if so, to what level. Table 11-11 shows the encoding of the PM bits.

Table 11-11. PM Encoding

PM[1:0]	FUNCTION
00	No Page Mode
01	Page Mode on Bursts Only
10	Reserved
11	Continuous Page Mode

11.5.1.3 DRAM Controller Mask Registers - DMR0 & DMR1 MBAR+\$10C,114.



These registers contain bits to mask off the compare function for addresses as well as to control response to certain Transfer Type/Transfer Modifier combinations.

11.5.1.3.1 Base Address Mask - BAM[31:18]

This register bits are provided to allow the user to mask or “don’t care” the associated bit in the Base Address (BA) register. If the bit is zero, the associated address is ignored to generate the DRAM hit. This lets you connect various size DRAMs to the SDRAMC. The function of each BAM bit is summarized in Table 11-12.

Table 11-12. Base Address Mask Encoding

BAM	FUNCTION
0	Associate Address Bit Is Compared In DRAM Hit
1	Associated Address Bit Is Ignored In DRAM Hit

11.5.1.3.2 Write Protect - WP

This bit determines if the associated bank of DRAM is write protected. If the bit is set, write accesses to the block of memory space occupied by the DRAM bank will be ignored by the DRAM Controller. Table 11-13 summarizes the WP bit operation.

Table 11-13. WP Encoding

WP	FUNCTION
0	Allow Write Accesses
1	Ignore Write Accesses

11.5.1.3.3 Address Modifier Masks - C/I, AM, SC, SD, UC, UD

These bits allow the associated type of access to be allowed to access DRAM. Table 11-14 shows the definition of the bits. If the bit is one, the associated access type is ignored. If the bit is zero, the associated access type is allowed to hit in the DRAM space. Table 11-15 summarizes the function of each bit.

Table 11-14. Address Modifier Bit Definitions

ADDRESS MODIFIER BIT	ASSOCIATED ACCESS TYPE
C/I	CPU Space / Interrupt Acknowledge
AM	Alternate Master
SC	Supervisor Code
SD	Supervisor Data
UC	User Code
UD	User Data

Table 11-15. Address Modifier Bit Encoding

ADDRESS MODIFIER BIT	FUNCTION
0	Allow Access Type To Hit In DRAM
1	Do Not Allow Access Type to Hit In DRAM

11.5.1.3.4 Valid - V

This bit is set to show that the registers that control the associated bank of DRAM have been initialized, and that the DRAM Controller can begin to decode DRAM accesses. Table

11-16 shows the encoding of the V bit. This bit is reset to zero to assure that DRAM banks are not erroneously decoded at reset.

Table 11-16. Valid Bit Encoding

V	FUNCTION
0	Do Not Decode DRAM Accesses
1	Registers Valid -- Decode Accesses

11.5.2 Asynchronous Operation

The SDRAMC provides a high level of functionality and flexibility, and at the same time remaining cost efficient and simple. There are 4 basic modes of asynchronous operation of the SDRAMC:

- Nonpage Mode
- Burst-Page Mode
- Continuous-Page Mode
- Extended Data-Out Mode

11.5.2.1 General Operation Guidelines. The SDRAMC provides control for the DRAM signals \overline{RAS} , \overline{CAS} , and $\overline{DRAMR/\overline{W}}$, as well as muxing the addresses, and providing bus cycle termination. The exact control of the signals and termination are determined by the mode of operation. The muxing scheme remains the same for all modes of operation, reducing the complexity of the muxing function. The address muxing scheme is shown in Table 11-17.

Table 11-17. Address Muxing Scheme

ADDRESS PIN	ROW ADDRESS	COLUMN ADDRESS
17	17	0 (8-bit)
16	16	1
15	15	2
14	14	3
13	13	4
12	12	5
11	11	6
10	10	7
9	9	8
17	17	16(16 & 32-bit)

Table 11-17. Address Muxing Scheme (Continued)

ADDRESS PIN	ROW ADDRESS	COLUMN ADDRESS
18	18	17
19	19	18
20	20	19
21	21	20
22	22	21
23	23	22
24	24	23
25	25	24

This muxing scheme allows various sized memories of differing width to be connected to the address bus. Each combination will have a fixed page size of 512 bytes for page mode operations. The addresses will be connected differently for the various size width combinations. A summary of how 8-bit, 16-bit, and 32-bit memories are connected to the address bus is shown in Table 11-18, Table 11-19, and Table 11-20 respectively. In each of these summaries, the Address pin refers to the external processor pin. The Row Address refers to the processor address that is valid for the Row Address Strobe. It always corresponds to the address pin. The Column Address is the processor address bit that is valid on the corresponding address pin when the column address strobe is driven. The memory sizes show what DRAM size will be 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 muxing. Notice also that Address Pin 17 is treated differently in byte-wide operations.

11

Table 11-18. DRAM Addressing for Byte-Wide Memories

ADDRESS PIN	ROW ADDRESS	COLUMN ADDRESS	MEMORY SIZE
17	17	0	Base Memory Size of 256 KB
16	16	1	
15	15	2	
14	14	3	
13	13	4	
12	12	5	
11	11	6	
10	10	7	
9	9	8	

Table 11-18. DRAM Addressing for Byte-Wide Memories (Continued)

ADDRESS PIN	ROW ADDRESS	COLUMN ADDRESS	MEMORY SIZE
19	19	18	1 MB
21	21	20	4 MB
23	23	22	16 MB
25	25	24	64 MB

Table 11-19. DRAM Addressing for 16-Bit Wide Memories

ADDRESS PIN	ROW ADDRESS	COLUMN ADDRESS	MEMORY SIZE
16	16	1	Base Memory Size of 128 KB
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 KB
20	20	19	2 MB
22	22	21	8 MB
24	24	23	32 MB

Table 11-20. DRAM Addressing for 32-Bit Wide Memories

ADDRESS PIN	ROW ADDRESS	COLUMN ADDRESS	MEMORY SIZE
15	15	2	Base Memory Size of 64 KB
14	14	3	
13	13	4	
12	12	5	
11	11	6	
10	10	7	
9	9	8	
17	17	16	256 KB
19	19	18	1 MB
21	21	20	4 MB
23	23	22	16 MB
25	25	24	64 MB

In the asynchronous mode, \overline{RAS} and \overline{CAS} will always transition at the falling edge of the clock. Accesses by external masters will be treated in the SDRAMC the same as internal accesses. The operation and timing of each bank of DRAM is controlled by separate registers, but refresh is treated the same for both banks. All DRAM accesses should be terminated by the SDRAMC. There is no priority encoding between the banks. If the user programs the banks to overlap or to overlap with other internal resources, undefined behavior may result.

11.5.2.2 Nonpage Mode-Operation. The simplest operation mode is nonpage mode. In this mode, the SDRAMC will provide termination and run a separate bus cycle for each transfer of data. Figure 11-5 shows the operation of the non-page mode access. This case shows a DRAM read followed by a write. Addresses for a new bus cycle are driven at the rising edge of the clock. If there is a hit in the DRAM bank, the associated \overline{RAS} signal is driven at the next falling edge. In this case, with $RCD = 0$, 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. \overline{CAS} remains asserted for the period programmed in the CAS bits. In this case, $RNCN$ is equal to one, 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 the \overline{CAS} signal is valid.

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. Since $RNCN = 0$, \overline{RAS} and \overline{CAS} are negated at the same time. The next bus cycle is initiated, but since the RP bits require \overline{RAS} to be

precharged for two clocks, the $\overline{\text{RAS}}$ is delayed for a clock in the bus cycle. Note that this does not delay the address signals, only $\overline{\text{RAS}}$.

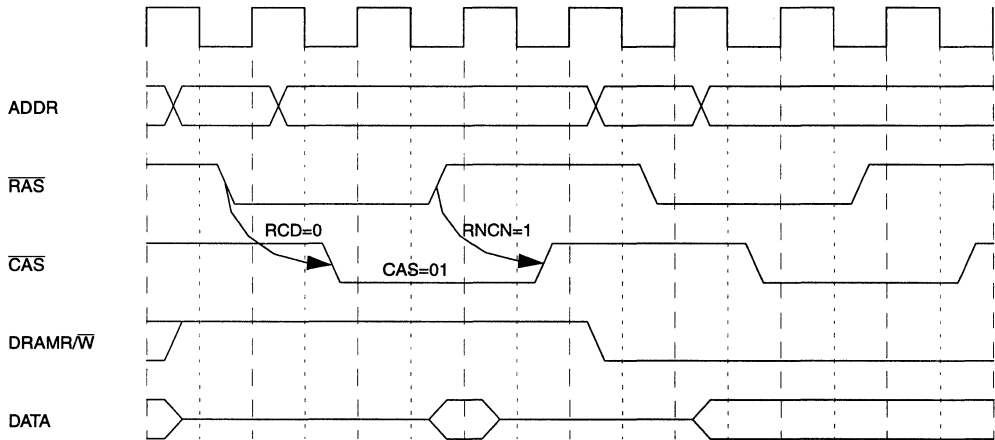


Figure 11-5. Basic Nonpage-Mode Operation RCD=0, RNCN=1

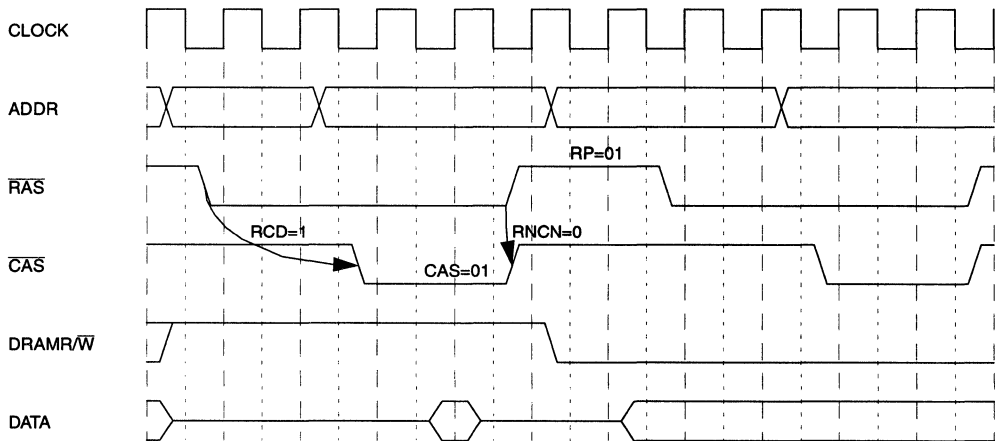


Figure 11-6. Basic Nonpage-Mode Operation RCD = 1, RNCN = 0

11.5.2.3 Burst Page-Mode Operation. Page mode operation permits optimization of memory accesses by allowing the row address to remain registered in the DRAM, while

accessing data in different columns. This relieves the necessity to precharge and assert $\overline{\text{RAS}}$ with the associated setup and hold times. Therefore, the first bus cycle in the page takes the full access time, but subsequent accesses are streamlined. Single accesses look the same as non-page mode accesses.

In the Burst Page Mode, any accesses, byte, word, long word or line, is assumed to reside in the same page. In Burst Page Mode, the SDRAMC will generate a burst transfer when the operand size is larger than the DRAM bank port size (e.g., line transfer to a 32-bit port, longword transfer to an 8-bit port, etc.). The primary cycle asserts $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$, and the subsequent secondary cycles will assert only $\overline{\text{CAS}}$. At the end of the access, $\overline{\text{RAS}}$ is precharged. The addresses will be incremented between cycles.

Figure 11-7 shows a read access in page mode. In this case, four accesses take place. These 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 size of the access and the port size. In those cases, timing will be similar with more or fewer accesses. Figure 11-8 shows the write operation with the same configuration

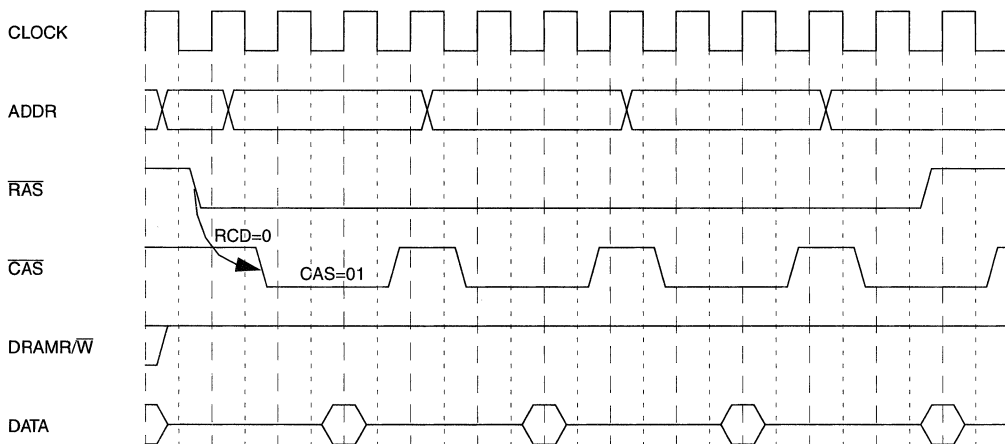


Figure 11-7. Burst Page-Mode Read Operation

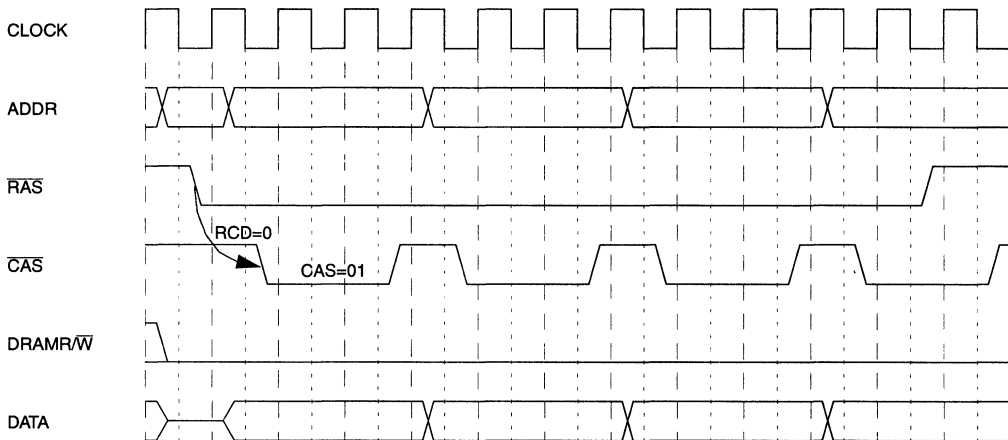


Figure 11-8. Burst Page-Mode Write Operation

11.5.2.4 Continuous-Page Mode. The SADRAMC incorporates a variation of page mode which tries to balance performance, complexity, and size. This is called Continuous Page Mode. In typical page mode implementations, the current address is compared with the previous address to determine if there is a hit in the DRAM bank. If the bus cycle hits, $\overline{\text{RAS}}$ remains asserted and $\overline{\text{CAS}}$ is asserted with the new column address. If there is a miss however, the $\overline{\text{RAS}}$ signal must be precharged again before the bus cycle begins.

Continuous Page Mode is a method to implement the page mode operation without requiring an address holding register per bank, as well as to eliminate the need to wait for a miss to precharge $\overline{\text{RAS}}$ for the upcoming bus cycle. Since the Internal 5307 Address bus implementation is a pipelined bus, addresses for the upcoming bus cycle are often available while the current bus cycle is being completed on the bus. Therefore at the end of the bus cycle the upcoming address is compared with the current address on the bus to determine if the upcoming address hits in the same page. If the access hits, $\overline{\text{RAS}}$ will remain asserted. If the next address misses, or there is no access pending, the $\overline{\text{RAS}}$ signal will be precharged before the next bus cycle becomes active on the external bus. The result is that there should never be a penalty paid for a page miss. Single accesses that are not followed by a hit in the page look the same as non-page mode accesses.

Figure 11-9 shows Continuous Page Mode operation. The first bus cycle is run, and the next bus cycle hits in the same page. The $\overline{\text{RAS}}$ signal is not negated before the next bus cycle is run. Note that in this case the row address does not appear on the pins for a bus cycle that hits in the page. The Column Address is immediately muxed onto the pins. In transitioning to the third bus cycle, a miss in the page is detected. The $\overline{\text{RAS}}$ line is precharged before the end of the bus cycle so that an extra delay is not incurred waiting for the line to precharge.

The example in Figure 11-9 shows a write cycle followed by a read cycle. If there is a hit in the page for a write cycle, CAS must be delayed by one clock to allow data to become valid, as shown in Figure 11-10.

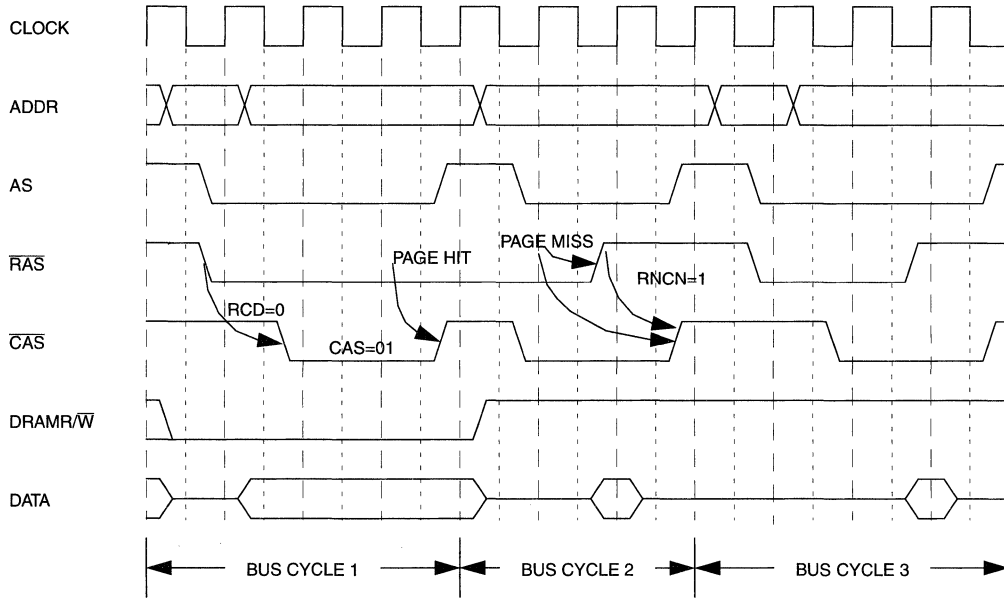


Figure 11-9. Continuous Page-Mode Operation

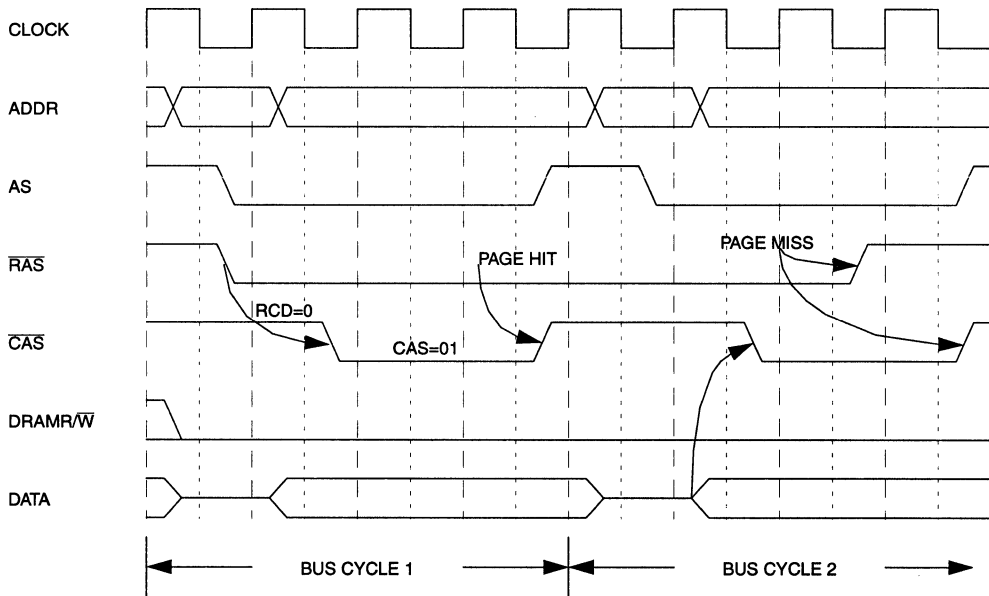


Figure 11-10. Write Hit in Continuous-Page Mode

11.5.2.5 Extended Data Out (EDO) Operation. Extended Data Out is a variation of page mode that allows the DRAM to continue driving data out of the device while $\overline{\text{CAS}}$ is being precharged. In order to support EDO DRAM's, the SDRAMC delays the internal termination of the cycle by one clock to capture data after $\overline{\text{CAS}}$ is being precharged. In order for data to be driven by the DRAM's the $\overline{\text{RAS}}$ signal is held after $\overline{\text{CAS}}$ is negated. EDO operation does not affect write operations. EDO DRAM's may be used in either the Full Page Mode or the Burst Page Mode. Single accesses that are not followed by a hit in the page look the same as non-page mode accesses.

Figure 11-11 shows 4 consecutive EDO accesses. Note that the data is sampled after the $\overline{\text{CAS}}$ has been negated. Note also that on the last page access, $\overline{\text{CAS}}$ is held until after the data is sampled to assure that the data is driven. This allows $\overline{\text{RAS}}$ to be precharged before the end of the cycle.

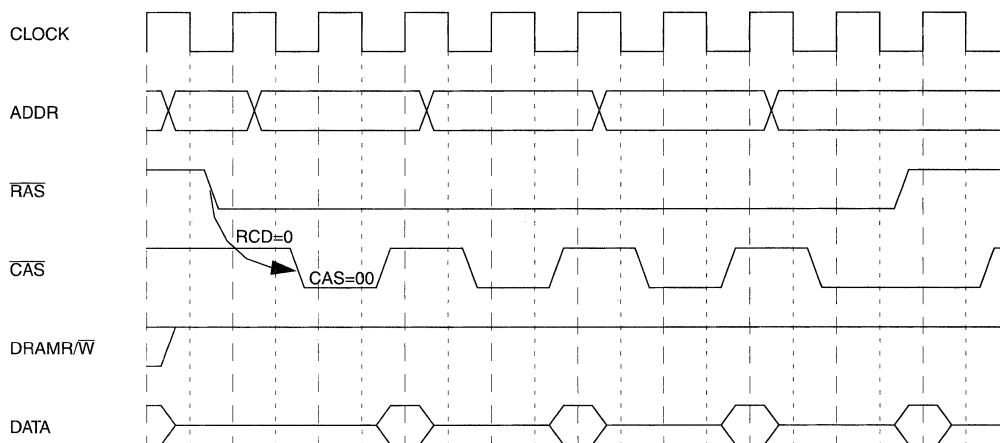


Figure 11-11. EDO Read Operation

11.5.2.6 Refresh Operation. Refresh support is provided in the SDRAMC by $\overline{\text{CAS}}$ before $\overline{\text{RAS}}$ refresh operation. Refresh is not synchronized to the bus activity. A special $\overline{\text{DRAMR/W}}$ pin is provided so that refresh can take place regardless of the state of the processor bus. When the Refresh Counter rolls over, it sets an internal flag to tell the DRAM controller that a refresh is pending. If refresh becomes pending during a Continuous Page Mode Access, the page will be closed (RAS precharged) when the operand transfer is completed. This will allow the refresh to take place. That flag is cleared as soon as the refresh cycle is run. The only event that will delay refresh is an active bus access operating in one of the DRAM banks. Refresh will take place in both banks at the same time as determined by the Refresh Control Register (RCR). Any DRAM accesses will be delayed while refresh takes place.

Figure 11-12 shows a bus cycle delayed by a refresh operation. Notice that the $\overline{\text{DRAMR/W}}$ signal is forced high during refresh. The row address is held until the DRAM access takes place.

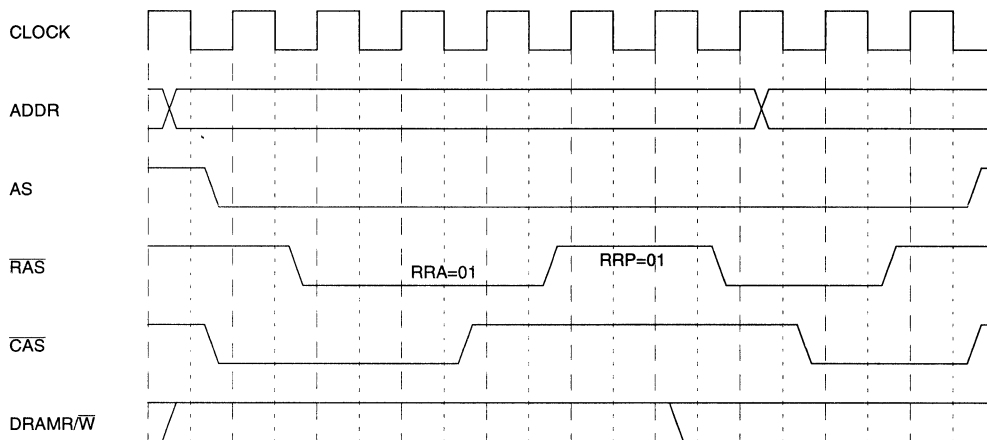


Figure 11-12. DRAM Access Delayed By Refresh

11.5.2.7 External Master Support. External masters will be supported with the SADRAMC. The SADRAMC will provide strobes and termination for the external master, but external address row and column multiplexing must be provided in this case.

11.6 SYNCHRONOUS OPERATION

In the synchronous mode of operation, signals, registers, and timing differ greatly from the asynchronous mode. Once synchronous operation is selected by setting the SO bit in the DCR, the memory map will reflect the synchronous operation, and there is no way to return to asynchronous operation without resetting the processor.

11.6.1 Synchronous Memory Map

The SADRAMC memory map shown in Table 11-1 is the same for both synchronous and asynchronous operation. The specific bits in the registers change function, and they are described below.

11.6.1.1 DRAM Control Register -DCR

MBAR + \$100.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SO	Rsvd	NAM	COC	IS	RTIM			RC							
RESET:															
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This register 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.

11.6.1.1.1 Synchronous Operation - SO

This bit determines if the SDRAMC is in synchronous operation mode. For synchronous DRAMs, this bit must be set to one. Note that once the part has entered into synchronous operation, it cannot be returned to asynchronous operation except by a reset. Table 11-21 summarizes the SO bit encoding.

Table 11-21. SO Bit Encoding

SO	FUNCTION
0	Asynchronous Operation
1	Synchronous Operation

11.6.1.1.2 No Address Multiplexing - NAM

Some implementations will require external multiplexing support. For instance, if there is an external master accessing the DRAM or if a linear addressing scheme is required the SDRAMC multiplexing may not be sufficient. In these cases, it would be advantageous to prevent the SDRAMC from multiplexing the addresses on a DRAM access. If this bit is set to a 1, the SDRAMC will not multiplex the external address bus to provide column addresses. Table 11-22 shows the NAM encoding.

Table 11-22. NAM Bit Encoding

NAM	FUNCTION
0	Address Bus is Multiplexed for DRAM Accesses
1	Address Bus is not Multiplexed for DRAM Accesses

11.6.1.1.3 Command on Clock Enable - COC

Implementations that utilize external multiplexing must have support for command information to be multiplexed onto the SDRAM address bus. This bit allows the command information to be driven out on what is normally the SDRAM clock enable (SCKE). In this case, the SDRAMC will not support self refresh operation, but external support may be generated. If the COC bit is set, the address command bit information will be generated on the SCKE pin. External multiplexing will be responsible for putting the command information on the proper address bit. Table 11-23 shows the COC encoding.

Table 11-23. COC Bit Encoding

COC	FUNCTION
0	Clock Enable on SCKE pin
1	Command on SCKE pin

11.6.1.1.4 Initiate Self-Refresh Command - IS

This bit tells the DRAM controller to send the SELF command to both banks to cause the SDRAMs to enter into low-power self-refresh state where they will remain until the IS bit is cleared. When the IS bit is cleared, the DRAM controller will send the SELFX command to the SDRAMs to tell them to exit the self-refresh state. The refresh counter is suspended while the SDRAMs are in self-refresh. Table 7-24 summarizes the IS bit encoding.

Table 11-24. IS Bit Encoding

IS	FUNCTION
0	Take no Action or Exit Self Refresh
1	Initiate Self Refresh or stay in Self Refresh

11.6.1.1.5 Refresh Timing - RTIM

These bits will determine the timing operation of Auto-Refresh in the SDRAMC. Specifically, it will determine the number of clocks inserted between the REF command and the next possible ACTV command. This same timing is used for both banks of the SDRAMC. This corresponds to t_{RC} in the SDRAM specifications. Table 11-25 shows the encoding of the RTIM bits.

Table 11-25. RTIM Encoding

RTIM	REFRESH TO ACTV (T_{RC})
00	3
01	6
10,11	9

11.6.1.1.6 Refresh Count - RC

This field controls the frequency of refresh performed by the SDRAMC. One is added to the value stored in these register locations and multiplied by 16 bus clocks to determine the refresh period. This operation is the same as in asynchronous mode.

11.6.1.2 Address & Control Registers - DACR0 & DACR1

MBAR+\$108, 110.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BA														Reserved	
RESET:															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RE	Rsvd	CASL		Rsvd	CBM			Rsvd	IMRS	PS		IP	PM	Rsvd	
RESET:															
0															

These registers contain the base address compare value and the control bits for both banks 0 and 1 of the DRAM controller. Address and timing are also controlled by bits in the ACRx registers.

11.6.1.2.1 Base Address Register - BA[31:18]

These register bits are used in conjunction with the BAM bits in the DCMR to determine the address range in which the associated bank of DRAM will be located. Each 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 hits in the associated bank's DRAM block. These bits function the same as in asynchronous operation.

11.6.1.2.2 Refresh Enable - RE

This bit determines if the SADRAMC will generate a refresh cycle to the associated DRAM bank. Table 11-26 shows the encoding of the RE bit. This bit is reset to zero to ensure that the refresh function is disabled at reset.

Table 11-26. RE Encoding

RE	FUNCTION
0	Do Not Refresh Associated DRAM Bank
1	Refresh Associated DRAM Bank

11.6.1.2.3 Column Address Strobe Latency - CASL[1:0]

These bits determine how long the data is delayed after the $\overline{\text{CAS}}$ signal (or the read command) is asserted during a SDRAM access. This data delay corresponds to the t_{RCD} specification in most SDRAMs. This also implies other timings with respect to the SDRAM. These include active command to precharge command (t_{RAS}), precharge command to active command (t_{RP}), last data input to precharge command (t_{RWL}), and last data out to

early precharge (t_{EP}). Check your SDRAM specs for greater detail. Table 11-27 summarizes the CASL[1:0] encoding and function.

Table 11-27. Synchronous CASL Encoding

CASL[1:0]	NUMBER OF BUS CLOCKS				
	T_{RCD}	T_{RAS}	T_{RP}	T_{RWL}	T_{EP}
00	1	2	1	1	0
01	2	4	2	1	-1
10	3	6	3	2	-2
11	3	6	3	2	-2

11.6.1.2.4 Initiate Mode Register Set Command - IMRS

This bit will generate the Mode Register Set (MRS) command to the associated SDRAMs. To use this feature, the base address and mask registers must be set. The associated CBM bits should also be initialized. After the IMRS bit is set, the next access to the address space of the SDRAM will cause the MRS command to that SDRAM to be generated. The address of the access should be selected in order to place the correct mode information on the address pins of the SDRAM. This bit is set to initiate a MRS command. The DRAM controller will clear the bit when the command is finished. Table 11-28 summarizes the function of the IMRS Bit. Any accesses via the IMRS bit should be restricted to be no wider than the port size programmed in the PS bits.

Table 11-28. IMRS Bit Encoding

IMRS	FUNCTION
0	Take no Action
1	Initiate MRS Command

11.6.1.2.5 Command and Bank Mux - CBM[2:0]

Because different SDRAM configurations will cause the bank and column bits to correspond to different addresses these resources have been made programmable. These bits will determine the addresses these onto which these functions will be multiplexed. Table 11-29 shows the encoding of these bits. This encoding along with the address multiplexing scheme handles common organizations of 16-Mbit SDRAMs, as well as some organizations of 4-Mbit, and allowing room for future 64-Mbit implementations. Note that the bank select bits include a base bit and all address bits above. This is to allow for future implementations of SDRAM that have more than one bank select bit.

Table 11-29. CBM Encoding

CBM[2:0]	COMMAND BIT	BANK SELECT BITS
000	17	18 & UP
001	18	19 & UP
010	19	20 & UP
011	20	21 & UP
100	21	22 & UP
101	22	23 & UP
110	23	24 & UP
111	24	25 & UP

11.6.1.2.6 Port Size - PS

These two bits will determine the port size of the associated bank of SDRAM which will allow for dynamic sizing of the associated accesses. Table 11-30 shows the encoding of the PS bits. This encoding is the same as used in asynchronous operation.

Table 11-30. PS Encoding

PS[1:0]	PORT SIZE
00	32-Bit Port
01	8-Bit Port
10	16-Bit Port
11	16-Bit Port

11.6.1.2.7 Initiate Precharge All Command - IP

This bit will cause a Precharge All (PALL) command to be generated to the associated SDRAM bank. This is useful in the power up sequence of SDRAMs. In order to use this feature, the base address and mask registers must be set. The associated CBM bits should also be initialized. After the IP bit is set, the next access to the address space of the SDRAM will cause the PALL command to that SDRAM bank to be generated. It is automatically cleared by the DRAM controller after the PALL command is finished. Any

accesses via the IP bit should be restricted to be no wider than the port size programmed in the PS bits. Table 7-31 summarizes the function of the IP Bit.

Table 11-31. IP Bit Encoding

IP	FUNCTION
0	Take no Action
1	Initiate PALL Command

11.6.1.2.8 Page Mode -- PM

This bit determines how the associated SDRAM bank supports page mode operation. Table 11-32 shows the encoding of the PM bit.

Table 11-32. Synchronous PM Encoding

PM	FUNCTION
0	Page Mode on Bursts Only
1	Continuous Page Mode

11.6.1.3 DRAM Controller Mask Registers - DMR0 & DMR1 MBAR+\$10C,114.

These registers are equivalent in synchronous and asynchronous operation. See subsection 11.5.1.3 for details.

11.6.2 Synchronous/Asynchronous DRAM Controller Synchronous Operation

The Synchronous/Asynchronous DRAM controller was designed to provide a high level of functionality and flexibility while remaining cost effective. There are 2 basic modes of operation of the SDRAMC in synchronous mode:

- Burst-Page Mode
- Continuous-Page Mode

11.6.2.1 Synchronous DRAM General Operation Guidelines. The SDRAMC provides control for the SDRAM signals \overline{SRAS} , \overline{SCAS} , \overline{DRAMRW} , \overline{SDQM} , and \overline{SCS} , as well as multiplexing the addresses to support common SDRAM implementations.

11.6.2.1.1 Address Multiplexing

The basic address multiplexing is the same as for the asynchronous operation as shown in Table 11-17 with the addition of support for a command bit and bank addresses as described in Table 11-29. This multiplexing scheme provides for a high level of flexibility with respect to which types of SDRAM are supported. Tables 11-33 through 11-38 give examples of how the address can be connected to support various organizations of SDRAM. This is not an exhaustive set of possibilities, but gives examples of the principle concepts. The first four examples show 16-Mbit SDRAMs. The next is a 4-Mbit SDRAM

implementation. The last is a 64-Mbit SDRAM implementation. Also note that the addressing scheme does not support all organizations of 4-Mbit SDRAM.

Table 11-33. Address Setup for 2M X 2 Bank X 4-Bit SDRAM X 2 as 8-Bit Port

SDRAM ADDRESS PIN	ADDRESS PIN	ACTV COMMAND FUNTION	READ/WRITE COMMAND FUNCTION
A0	A17	A17	A0
A1	A16	A16	A1
A2	A15	A15	A2
A3	A14	A14	A3
A4	A13	A13	A4
A5	A12	A12	A5
A6	A11	A11	A6
A7	A10	A10	A7
A8	A9	A9	A8
A9	A19	A19	A18
A10	A20	A20	Command
A11	A21	Bank Select	Bank Select

Table 11-34. Address Setup for 2M X 2Bank X 4-Bit SDRAM X 4 as 16-Bit Port

SDRAM ADDRESS PIN	ADDRESS PIN	ACTV COMMAND FUNCTION	READ/WRITE COMMAND FUNCTION
A0	A16	A16	A1
A1	A15	A15	A2
A2	A14	A14	A3
A3	A13	A13	A4
A4	A12	A12	A5
A5	A11	A11	A6
A6	A10	A10	A7
A7	A9	A9	A8
A8	A18	A18	A17
A9	A20	A20	A19

Table 11-34. Address Setup for 2M X 2Bank X 4-Bit SDRAM X 4 as 16-Bit Port

SDRAM ADDRESS PIN	ADDRESS PIN	ACTV COMMAND FUNCTION	READ/WRITE COMMAND FUNCTION
A10	A21	A21	Command
A11	A22	Bank Select	Bank Select

Table 11-35. Address Setup for 2M X 2 Bank X 4-Bit SDRAM X 8 as 32-Bit Port

SDRAM ADDRESS PIN	ADDRESS PIN	ACTV COMMAND FUNCTION	READ/WRITE COMMAND FUNCTION
A0	A15	A15	A2
A1	A14	A14	A3
A2	A13	A13	A4
A3	A12	A12	A5
A4	A11	A11	A6
A5	A10	A10	A7
A6	A9	A9	A8
A7	A17	A17	A16
A8	A19	A19	A18
A9	A21	A21	A20
A10	A22	A22	Command
A11	A23	Bank Select	Bank Select

Table 11-36. Address Setup for 512 K X 2 Bank X 16-Bit SDRAM X 1 as 16-Bit Port

SDRAM ADDRESS PIN	ADDRESS PIN	ACTV COMMAND FUNCTION	READ/WRITE COMMAND FUNCTION
A0	A16	A16	A1
A1	A15	A15	A2
A2	A14	A14	A3
A3	A13	A13	A4
A4	A12	A12	A5
A5	A11	A11	A6

Table 11-36. Address Setup for 512 K X 2 Bank X 16-Bit SDRAM X 1 as 16-Bit Port

SDRAM ADDRESS PIN	ADDRESS PIN	ACTV COMMAND FUNCTION	READ/WRITE COMMAND FUNCTION
A6	A10	A10	A7
A7	A9	A9	A8
A8	A17	A17	Not Used
A9	A18	A18	Not Used
A10	A19	A19	Command
A11	A20	Bank Select	Bank Select

Table 11-37. Address Setup for 128 K X 2 Bank X 16-Bits SDRAM X1 as 16-Bit Port

SDRAM ADDRESS PIN	ADDRESS PIN	ACTV COMMAND FUNCTION	READ/WRITE COMMAND FUNCTION
A0	A16	A16	A1
A1	A15	A15	A2
A2	A14	A14	A3
A3	A13	A13	A4
A4	A12	A12	A5
A5	A11	A11	A6
A6	A10	A10	A7
A7	A9	A9	A8
A8	A17	A17	Command
A9	A18	Bank Select	Bank Select

Table 11-38. Address Setup for 1M X 4 Bank X 16-Bits SDRAM X 2 as 32-Bit Port

SDRAM ADDRESS PIN	ADDRESS PIN	ACTV COMMAND FUNCTION	READ/WRITE COMMAND FUNCTION
A0	A15	A15	A2
A1	A14	A14	A3
A2	A13	A13	A4
A3	A12	A12	A5
A4	A11	A11	A6

Table 11-38. Address Setup for 1M X 4 Bank X 16-Bits SDRAM X 2 as 32-Bit Port

SDRAM ADDRESS PIN	ADDRESS PIN	ACTV COMMAND FUNCTION	READ/WRITE COMMAND FUNCTION
A5	A10	A10	A7
A6	A9	A9	A8
A7	A17	A17	A16
A8	A18	A18	not used
A9	A19	A19	not used
A10	A20	A20	not used
A11	A21	A21	Command
A12	A22	Bank Select	Bank Select
A13	A23	Bank Select	Bank Select

11.6.2.1.2 Power-On Sequence

Synchronous DRAMs have a prescribed power-on sequence. The SADRAMC supports this sequence with the following procedure:

- The SDRAM control signals are reset to the Idle state. Wait the prescribed period after reset before any action is taken on the SDRAMs.
- Set up the DCR, ACR, and DCMR registers in their operational configuration. Do not yet enable refresh commands.
- Issue the PALL command to the SDRAMs by setting the associated bit in the DCR, and accessing a memory space within the SDRAM.
- Enable refresh, and wait a period long enough for at least 8 refreshes to take place.
- Issue the MRS command by setting the IMRS bit in the ACRs, and accessing a memory space within the SDRAM.
- NOTE: Accesses for the PALL and MRS commands must be to an address which is appropriate. SDRAM registers (registers on the SDRAM, not in the MCF5307 SADRAMC) are accessed via the address bus. Use an address correct for the SDRAM configuration and muxing scheme.

11.6.2.1.3 Mode Register Settings

It is possible to configure the operation of SDRAMs, namely their burst operation and CAS latency. The CAS latency is a function of the speed of the SDRAM and the bus clock of the SADRAMC. The SADRAMC will operate at a CAS latency of 1, 2, or 3.

Although the SADRAMC supports bursting operations, it does not use the bursting features of the SDRAMs. Since operand sizes can be 1, 2, 4, or 16 bytes long, support of a fixed burst length in the SDRAMs mode register becomes problematic. Therefore, the

SADRAMC will generate a new address and read or write command for each transfer within the burst. Therefore, the SDRAM mode register should be either set to a burst length of one, or not to burst.

The mode register in the SDRAM is written by setting the IMRS bit in the associated bank's ACR register. First the base address and mask registers must be set. The associated CBM bits should also be initialized. After the IMRS bit in the ACR is set, the next access to the address space of the SDRAM will cause the MRS command to that SDRAM to be generated. The address of the access should be selected in order to place the correct mode information on the address pins of the SDRAM. The addresses will not be multiplexed for the MRS command. The MRS access should be a write access. Figure 11-13 shows the Mode Register Set command. The MRS takes place during the first clock of the bus cycle.

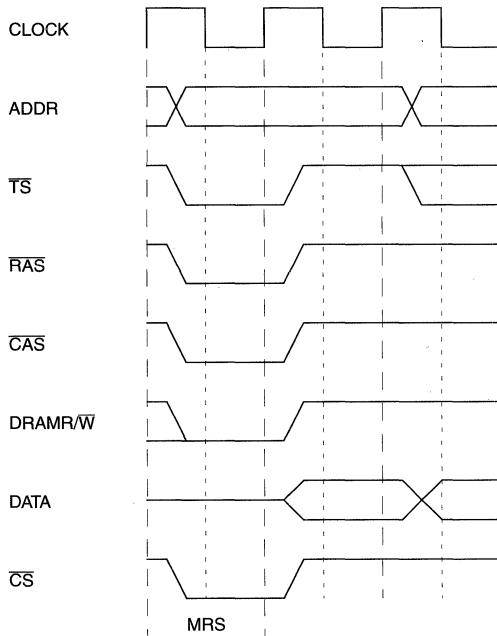


Figure 11-13. Mode Register Set Command

11.6.2.1.4 General Details

All signals generated in synchronous mode by the SADRAMC transition at the rising edge of the bus clock. External masters are handled by the SADRAMC the same as internal accesses. All SDRAM accesses should be terminated by the SADRAMC. There is no priority encoding between banks. If the user programs the banks to overlap, or to overlap with other internal resources, undefined behavior may result.

11.6.2.2 Burst-Page Mode. The advantage of SDRAM is the ability to quickly and efficiently provide data once the page of SDRAM has been opened. Once CAS has been issued the SDRAM will accept a new address and CAS every clock for as long as the accesses are in that page. In burst page mode, there will be multiple read or write operations for every ACTV command in the SDRAM if the requested transfer size is greater than the port size of the associated SDRAM. The primary cycle of the transfer will generate the ACTV and READ or WRITE commands and the secondary cycles will generate only READ or WRITE commands. As soon as the transfer is completed, the PALL command is generated to begin preparation for the next access.

In order to provide the ability to quickly transfer various sizes of data, the SDRAMC does not utilize the burst capability in the SDRAM architecture. Instead of using the SDRAM burst, addresses are incremented internally to the SDRAMC to allow fast accesses.

Figure 11-14 shows a burst read operation. Notice that the PALL command takes place before all of the data is read. Figure 11-15 shows the burst write operation. Notice that in the case of a write, the bus cycle is terminated sooner than with the read case. The next

bus cycle is initiated sooner, but is not allowed to begin an SDRAM cycle until the precharge to ACTV delay is completed.

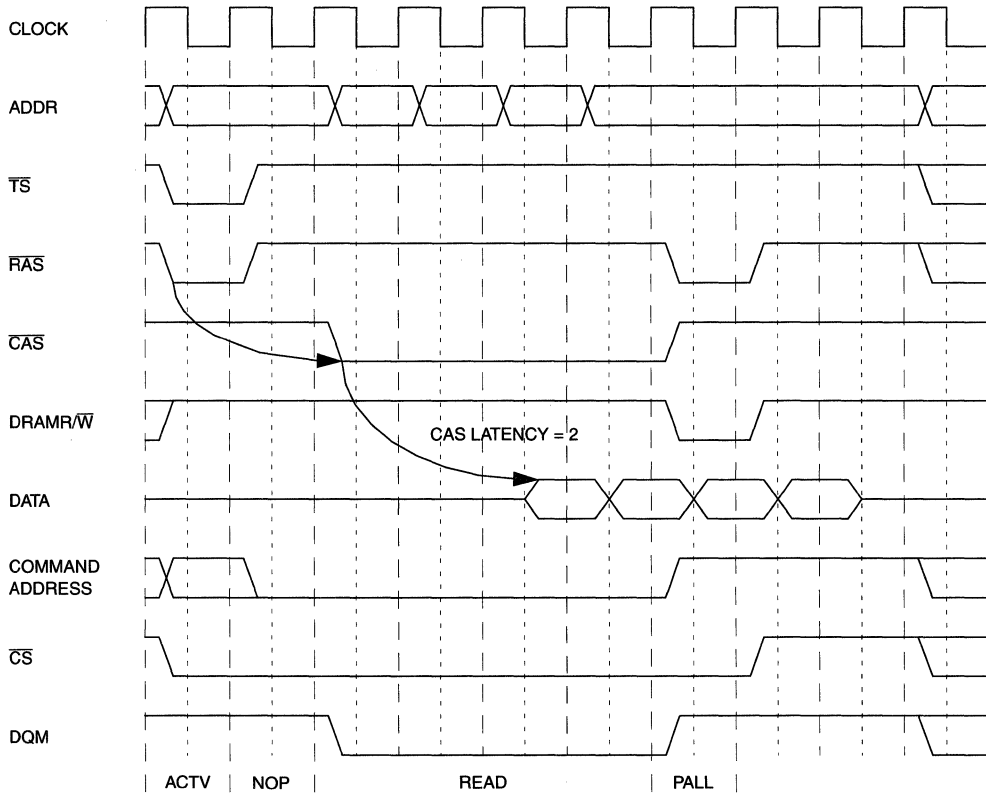


Figure 11-14. Burst Read SDRAM Access

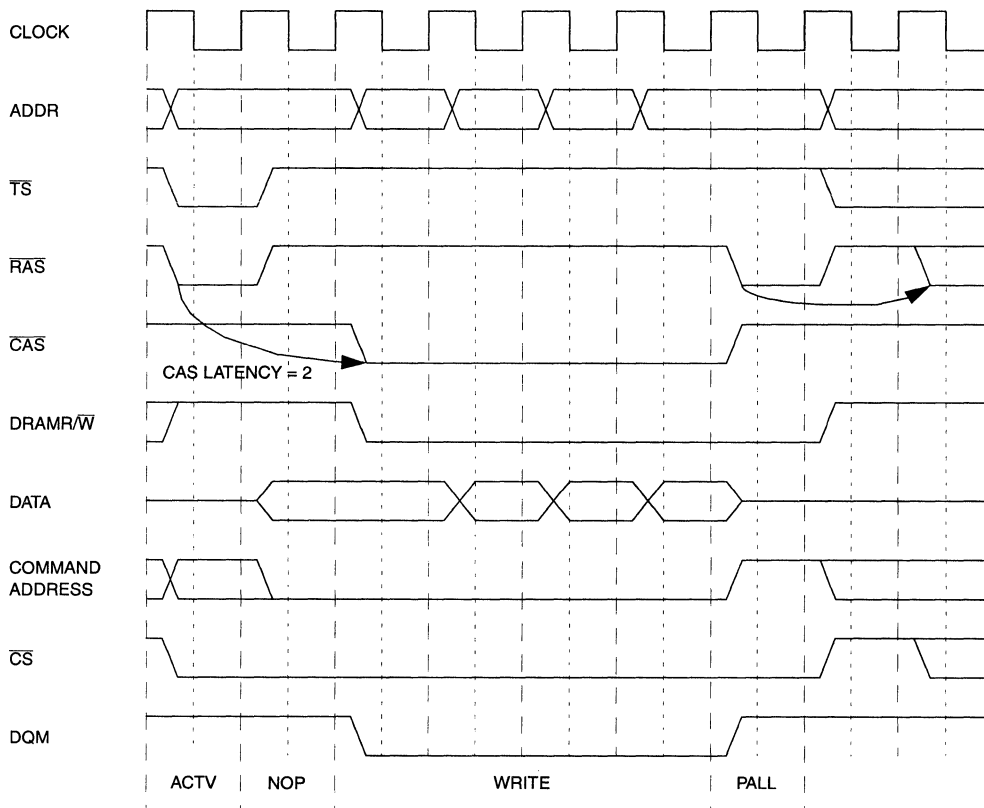


Figure 11-15. Burst Write SDRAM Access

Accesses in synchronous burst page mode will always cause the following sequence:

1. ACTV command
2. NOP commands to assure RAS to CAS delay (if CAS latency is 1, there are no NOP commands).
3. Required number of Read Commands or Write Commands to service the transfer size with the given port size.
4. Some transfers may require 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.

11.6.2.3 Continuous-Page Mode. The SDRAMC incorporates a variation in page mode that tries to balance performance complexity and size. This is called continuous-page mode.

Continuous-page mode operates with the internal pipelined bus of the MCF5307 CPU Core to predict whether the upcoming bus cycle will hit in the same page of SDRAM as it finishes the current bus cycle. If the next bus cycle is not pending or misses in the page, the PALL command is generated to the SDRAM. Issuing the PALL command in this manner allows the precharge to be hidden in the current cycle. If the next bus cycle is pending and hits in the page, the page is left open, and the next SDRAM access will begin with a read or write command. Figure 11-16 shows an example of a read access followed by a read 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.

11

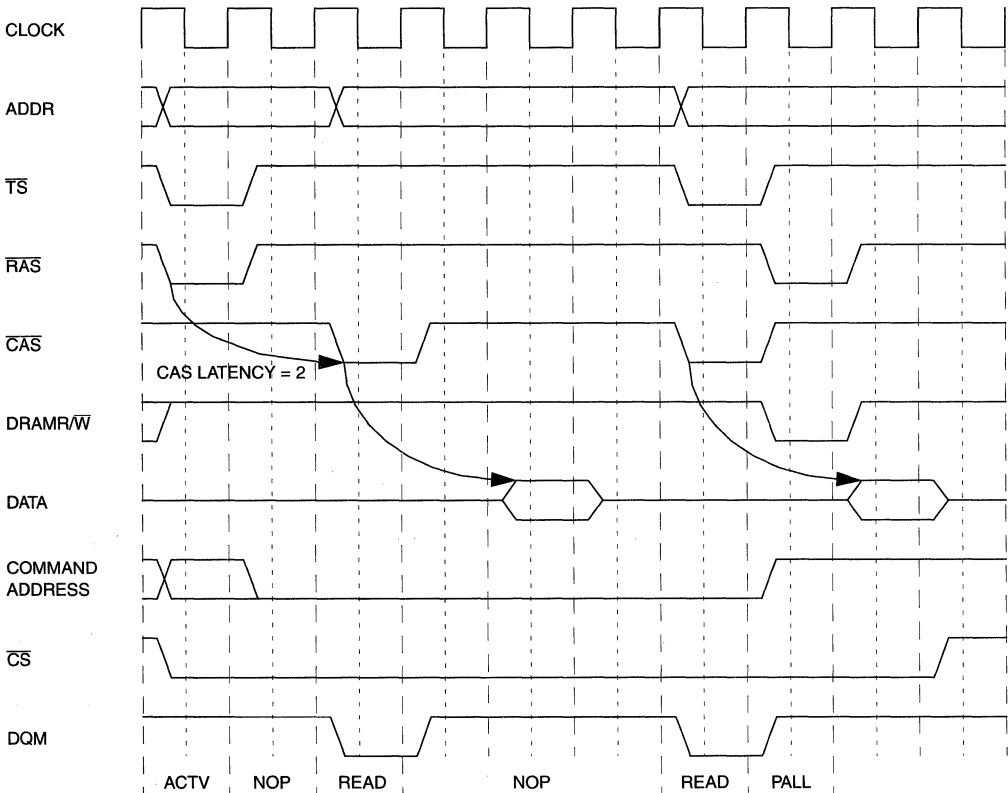


Figure 11-16. Synchronous Continuous Page Mode Access -- Read Followed by Read

Figure 11-17 shows an example of a write accesses followed by a read in continuous page mode. Note that the second cycle begins sooner after the write than after the read. This is because the bus cycle is terminated with the write command. A read requires data to be returned before the bus cycle may be terminated.

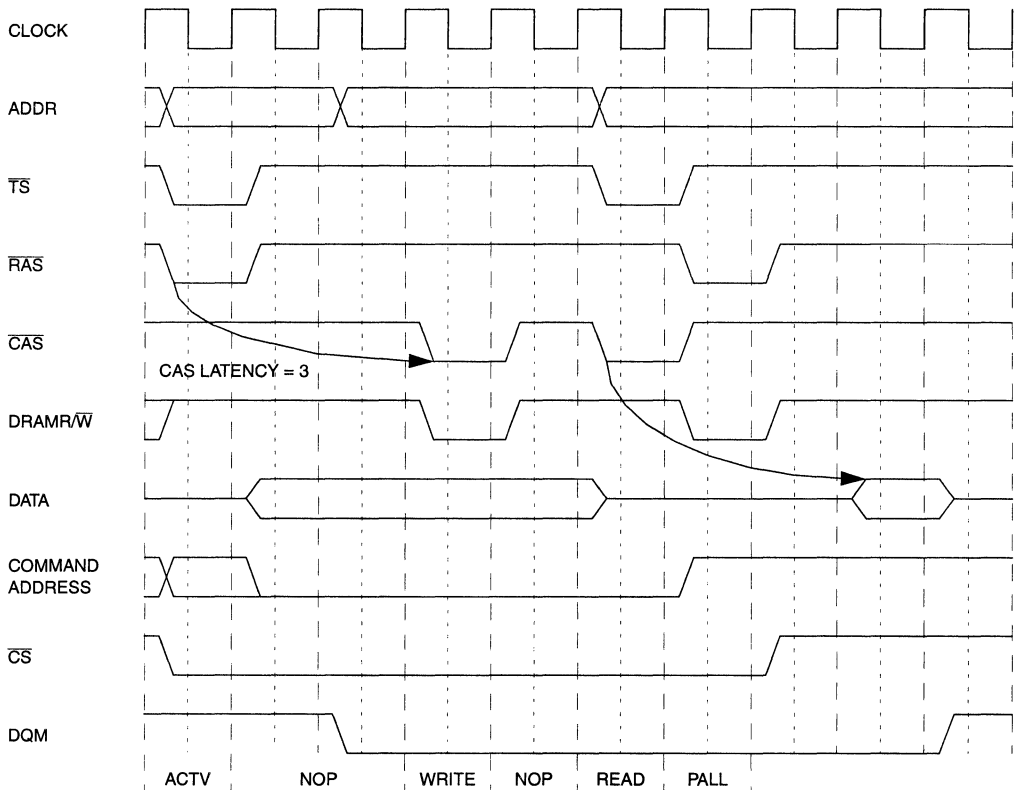


Figure 11-17. Synchronous Continuous Page Mode Access -- Write Followed by Read

Note that in continuous page mode operation, secondary accesses will not generate the physical address externally, but will present the multiplexed address.

11.6.2.4 Auto-Refresh Operation. The SADRAMC 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 SADRAMC completes any active burst operation and then performs a precharge all operation. The SADRAMC 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 that is initiated during the auto-refresh cycle will be delayed until the cycle is through.

Figure 11-18 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 the RTIM bits in the DCR 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.

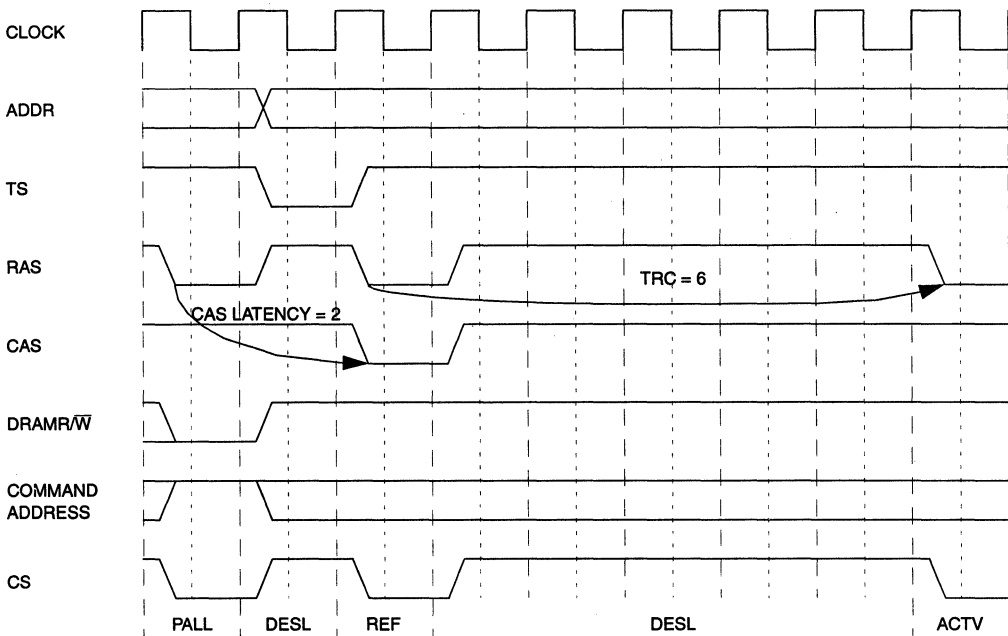


Figure 11-18. Auto-Refresh Operation

When a REF command is generated by the SADRAMC, it goes to both banks of external SDRAM at the same time. This is accomplished by both chip selects being active during the REF command.

11.6.2.5 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 SADRAMC supports self-refresh with the IS bit in the DCR. When the IS bit is set, the SELF command is sent to the SDRAM. When the bit is cleared the SELFX command is sent to the SADRAMC. Figure 11-19 shows the self refresh operation.

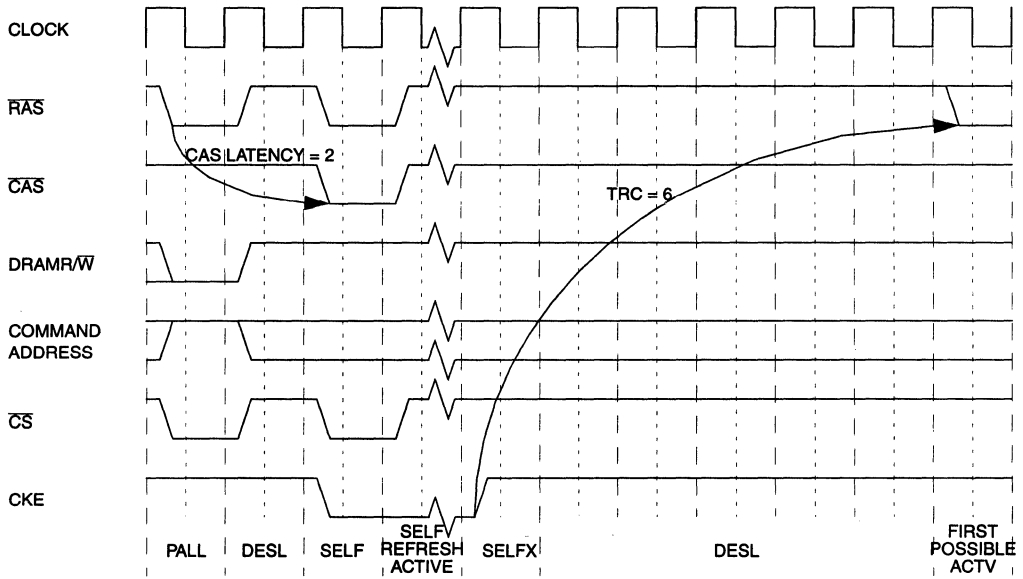


Figure 11-19. Self Refresh Operation

SECTION 12

TIMER MODULE

12.1 OVERVIEW

The timer module includes two independent general-purpose 16-bit timers. Timer references either timer 1 or timer 2 because both are functionally equivalent. The output of an 8-bit prescaler clocks each 16-bit timer.

The timers can operate from the system clock or an external clocking source on the TIN pin. If the system clock is selected, it can be divided by 16 or 1 in the Timer Mode Register (TMR). System bus clock is defined as $(CLKIN \times 2)/(2,3, \text{ or } 4)$. The PLL divides the 90 MHz internal clock by 2, 3, or 4 which yields a 45, 30, or 25MHz clocking source to the timers. See **Section 4 PLL** for details on defining bus clock speeds. Figure 12-1 is a block diagram of the timer module.

12.1.1 Key Features

The general-purpose 16-bit timer unit has the following features:

- Maximum period of 5.96 seconds at 45MHz, 8.95 seconds at 30MHz, and 11.93 seconds at 22.5MHz
- 22.2ns resolution at 45MHz, 33.3ns at 30MHz, 44.4ns at 22.5MHz
- 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

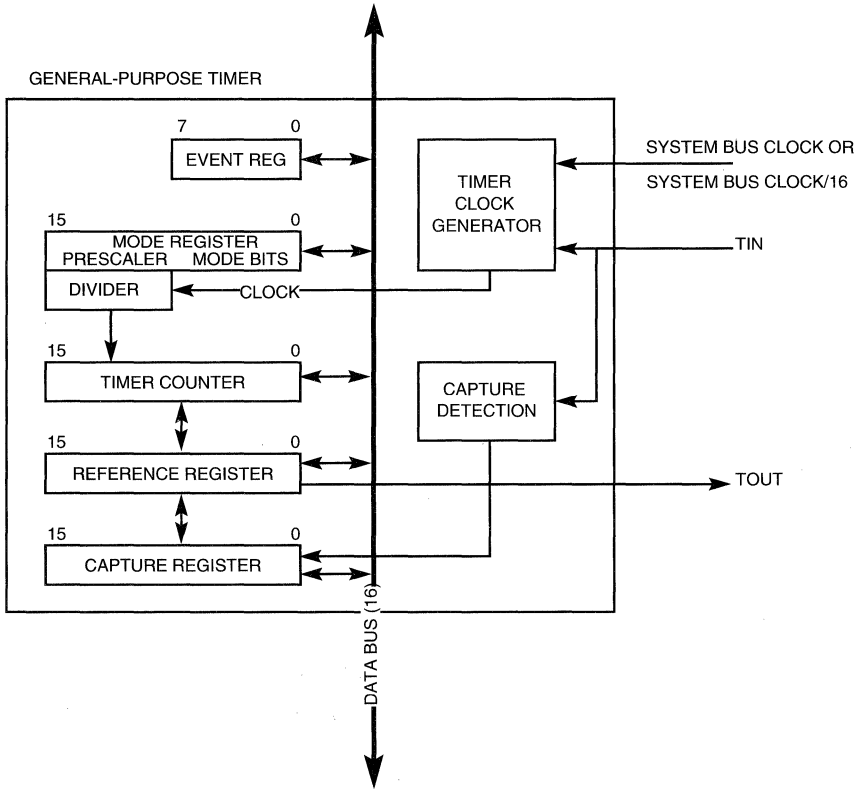


Figure 12-1. Timer Block Diagram

12.2 MODULE OPERATION

12.2.1 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 the TIN pin
- System bus clock can be divided by 16 or 1. This clock is input to the prescaler
- The TIN pin is fed directly into the 8 bit prescaler
- The 8 bit prescaler clock divides the clocking source and is user-programmable from 1 to 256
- Programmed events generate interrupts
- The TOUT pin can be configured to toggle or pulse on an event
- If the TIN pin is used as a clocking source, it can not be used as an input capture. If the internal clock is used, the TIN pin can be used as an input capture.

12.2.1.1 PRESCALER. The prescaler clock input may be selected from the system bus clock (divided by 1 or by 16), or from the corresponding timer input TIN pin. TIN is synchronized to the system bus clock. The synchronization delay is between two and three main clocks. The ICLK bits of the corresponding TMR select the clock input source. The prescaler is programmed to divide the clock input by values from 1 to 256. The prescaler output is used as an input to the 16-bit counter.

12.2.1.2 CAPTURE MODE. The timer has a 16-bit Timer Capture Register (TCR) that latches the counter value when the corresponding input capture edge detector senses a defined transition (of TIN). The Capture Edge (CE) bits in the TMR select the type of transition triggering the capture. A capture event sets the Timer Event Register (TER) bit 0 and issues a maskable interrupt.

12.2.1.3 REFERENCE COMPARE. The timer may be configured to count until it reaches a reference value. It then either starts a new time count immediately or continues to run. The Free Run/Restart (FRR) bit of the TMR selects either mode. A timer that reaches the reference value sets the TER bit 1 and issues an interrupt if the Output Reference Interrupt (ORI) enable bit in the TMR is set.

12.2.1.4 OUTPUT MODE. The timer may send an output signal on the Timer Output (TOUT) pin when it reaches the reference value as selected by the Output Mode (OM) bit in the TMR. This signal can be an active-low pulse or a toggle of the current output under program control.

12.3 PROGRAMMING MODEL

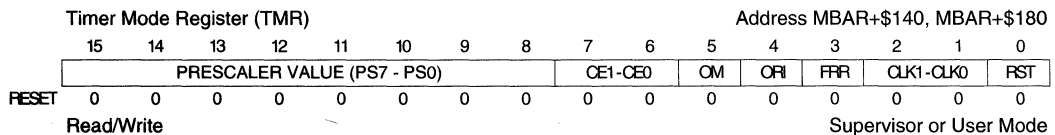
12.3.1 General-Purpose Timer Registers

You can modify the timer registers at any time. Table 12-1 illustrates the programming model.

Table 12-1. Programming Model for Timers

TIMER 1 ADDRESS	TIMER 2 ADDRESS	SIM MODULE-TIMER MODULE REGISTERS
MBAR+\$140	MBAR+\$180	Timer Mode Register (TMR)
MBAR+\$144	MBAR+\$184	Timer Reference Register (TRR)
MBAR+\$148	MBAR+\$188	Timer Capture Register (TCR)
MBAR+\$14C	MBAR+\$18C	Timer Counter (TCN)
MBAR+\$151	MBAR+\$191	Timer Event Register (TER)

12.3.1.1 TIMER MODE REGISTER (TMR). TMR is a 16-bit memory-mapped register. This register programs the various timer modes and is cleared by reset.



PS7-PS0 — Prescaler Value

The prescaler is programmed to divide the clock input by values (system bus clock/(16 or 1) or clock on TIN pin) from 1 to 256. The binary value 00000000 divides the clock by 1; the value 11111111 divides the clock by 256.

CE1-CE0 — Capture Edge and Enable Interrupt

- 11 = Capture on any edge and enable interrupt on capture event
- 10 = Capture on falling edge only and enable interrupt on capture event
- 01 = Capture on rising edge only and enable interrupt on capture event
- 00 = Disable interrupt on capture event

OM — Output Mode

- 1 = Toggle output
- 0 = Active-low pulse for one system bus clock cycle (22ns at 45MHz, 33ns at 30MHz, and 44ns at 22.5MHz)

ORI — Output Reference Interrupt Enable

- 1 = Enable interrupt upon reaching the reference value
- 0 = Disable interrupt for reference reached (does not affect interrupt on capture function)

NOTE

If ORI is set when the REF event is asserted in the Timer Event Register (TER), then an immediate interrupt will occur.

FRR — Free Run/Restart

- 1 = Restart: Timer count is reset immediately after reaching the reference value
- 0 = Free run: Timer count continues to increment after reaching the reference value

CLK1–CLK0 — Input Clock Source for the Timer

- 11 = TIN pin (falling edge)
- 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 in length
- 01 = System bus clock divided by 1
- 00 = Stop count

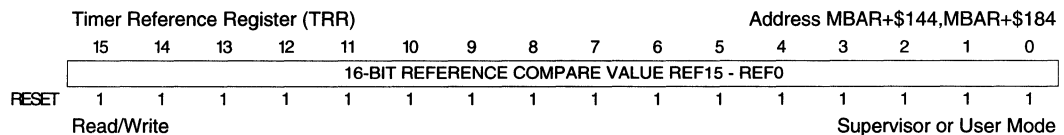
RST — Reset Timer

This bit performs a software timer reset similar to that of an external reset, although while this bit is zero, the other register values can still be written, if necessary. Effectively, a transition of this bit from one to zero is what resets the register values. The counter/timer/prescaler will not be clocked unless the timer is enabled.

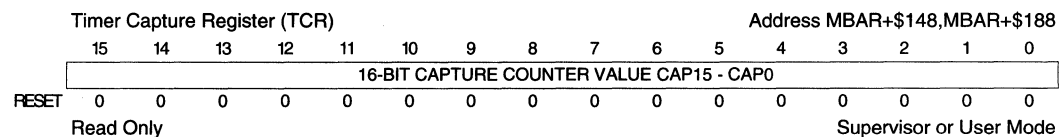
- 1 = Enable timer
- 0 = Reset timer (software reset)

12.3.1.2 TIMER REFERENCE REGISTER (TRR). The TRR is a 16-bit register containing the reference value that is compared with the free-running Timer Counter (TCN) as part of the output-compare function. TRR is a memory-mapped read/write register.

TRR is set to \$FFFF at reset. The reference value is not matched until TCN equals TRR.

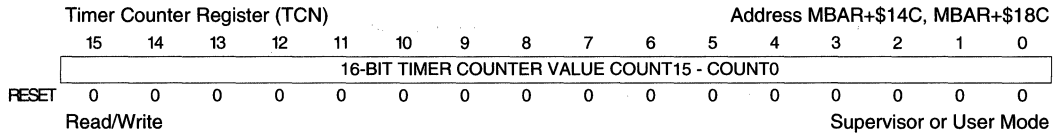


12.3.1.3 TIMER CAPTURE REGISTER (TCR). The TCR is a 16-bit register that latches the value of the Timer Counter (TCN) during a capture operation when an edge occurs on the TIN pin, as programmed in the TMR. TCR appears as a memory-mapped read-only register to users and is cleared to \$0000 at reset. This assumes that the system bus clock has been selected as the clocking source. The TIN pin cannot function as a clocking source and as an input capture pin simultaneously.



12.3.1.4 TIMER COUNTER (TCN). TCN is a memory-mapped 16-bit up-counter. Users can read it at any time. A read cycle to TCN yields the current timer value and does not affect the counting operation.

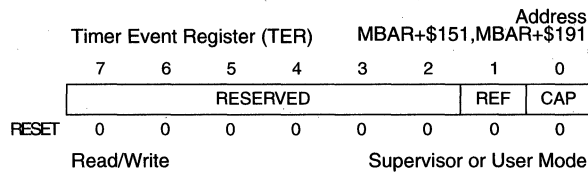
A write cycle to TCN causes it and the corresponding prescaler to be reset.



12.3.1.5 TIMER EVENT REGISTER (TER). The TER is an 8-bit register that reports events the timer recognizes. When the timer recognizes an event, it will set the appropriate bit in the TER, regardless of the corresponding interrupt-enable bits (ORI and CE) in the TMR.

TER, which appears to users as a memory-mapped register, may be read at any time.

Writing a 1 to any bit clears it (writing a zero does not affect bit value); more than one bit may be cleared at a time. The REF and CAP bits must be cleared before the timer will negate the IRQ to the interrupt controller. Reset clears this register.



Bits 7–2 — Reserved for future use.

CAP — Capture Event

The counter value has been latched into the TCR. The CE bit in the TMR enables the interrupt request caused by this event. Write a one to this bit to clear the event condition.

REF — Output Reference Event

The counter has reached the TRR value. The ORI bit in the TMR enables the interrupt request caused by this event. Write a one to this bit to clear the event condition.

12.3.1.6 TABLE OF TIMEOUT VALUES. Table 12-2 provides a table of time-out values for the timers. The values provided are the time it takes the 16-bit TCN register to count from \$0000 to \$FFFF. It assumes the clocking source is from the system bus clock and the Timer Reference Register is set to \$FFFF. TMR bits [2:1] define the clocking source to the timers. A binary value of 00 in TMR bits [2:1] divides the system bus clock by 1 and a value of 10 divides the system bus clock by 16.

TMR bits [2:1] define the clocking source to the timers. TMR bits [15:8] define the prescale value. Additionally, the system bus clock to the timers depends on the clock setting in the PLL. The PLL divides the 90 MHz internal clock by 2, 3, or 4 which yields a 45, 30, or 25MHz

clocking source to the timers (assuming a master clock of 45MHz on the CLKIN pin). See the PLL section for more details.

A time-out occurs when the value of the Timer Reference Register (TRR) is equal to the Timer Counter Register (TCN). Below is an example of how to define time-out period:

$$\text{Time-out Period} = (1/\text{clock}) \times (1 \text{ or } 16) \times (\text{TMR1 prescale value}+1) \times (\text{Timer ref. value})$$

For example, if the timer clock was 45 MHz, the clock was divided by 16 (TMR [2:1]=10), the TMR prescale was set to \$7F, and the timer reference was \$ABCD (43981 decimal), the time-out period would be:

$$\text{Time-out Period} = (1/45\text{E}6) \times (16) \times (127+1) \times (43981) = 2.00 \text{ seconds}$$

The TMR prescale value is always between and 1 and 256. When calculating time-out periods, add 1 to the prescale. This makes calculating easier since a prescale value of \$00 = 1 and \$FF=256. The table below assumes that the system clock is being used as the clocking source. If an external clock is used, the timeout values will need to be calculated

The table below shows the time-out values for various settings of the TMR. The time-out values shown (in seconds) are representative of the time it takes the counter to count from \$0000 to \$FFFF.

Table 12-2. Calculated Time-out Values (45MHz System Bus Clock).

TMR BITS [15:8]		45 MHz	30 MHz	22.5 MHz	45 MHz	30 MHz	22.5 MHz
DECIMAL	HEX	TMR REGISTER BITS [2:1]=10 (SYSTEM CLOCK/16)			TMR REGISTER BITS [2:1]=01 (SYSTEM CLOCK/1)		
Note: values below are in seconds							
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
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

Table 12-2. Calculated Time-out Values (45MHz System Bus Clock).

TMR BITS [15:8]		45 MHz	30 MHz	22.5 MHz	45 MHz	30 MHz	22.5 MHz
DECIMAL	HEX	TMR REGISTER BITS [2:1]=10 (SYSTEM CLOCK/16)			TMR REGISTER BITS [2:1]=01 (SYSTEM CLOCK/1)		
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
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

Table 12-2. Calculated Time-out Values (45MHz System Bus Clock).

TMR BITS [15:8]		45 MHz	30 MHz	22.5 MHz	45 MHz	30 MHz	22.5 MHz
DECIMAL	HEX	TMR REGISTER BITS [2:1]=10 (SYSTEM CLOCK/16)			TMR REGISTER BITS [2:1]=01 (SYSTEM CLOCK/1)		
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
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

Table 12-2. Calculated Time-out Values (45MHz System Bus Clock).

TMR BITS [15:8]		45 MHz	30 MHz	22.5 MHz	45 MHz	30 MHz	22.5 MHz
DECIMAL	HEX	TMR REGISTER BITS [2:1]=10 (SYSTEM CLOCK/16)			TMR REGISTER BITS [2:1]=01 (SYSTEM CLOCK/1)		
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
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

Table 12-2. Calculated Time-out Values (45MHz System Bus Clock).

TMR BITS [15:8]		45 MHz	30 MHz	22.5 MHz	45 MHz	30 MHz	22.5 MHz
DECIMAL	HEX	TMR REGISTER BITS [2:1]=10 (SYSTEM CLOCK/16)			TMR REGISTER BITS [2:1]=01 (SYSTEM CLOCK/1)		
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
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

Table 12-2. Calculated Time-out Values (45MHz System Bus Clock).

TMR BITS [15:8]		45 MHz	30 MHz	22.5 MHz	45 MHz	30 MHz	22.5 MHz
DECIMAL	HEX	TMR REGISTER BITS [2:1]=10 (SYSTEM CLOCK/16)			TMR REGISTER BITS [2:1]=01 (SYSTEM CLOCK/1)		
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
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

Table 12-2. Calculated Time-out Values (45MHz System Bus Clock).

TMR BITS [15:8]		45 MHz	30 MHz	22.5 MHz	45 MHz	30 MHz	22.5 MHz
DECIMAL	HEX	TMR REGISTER BITS [2:1]=10 (SYSTEM CLOCK/16)			TMR REGISTER BITS [2:1]=01 (SYSTEM CLOCK/1)		
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
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

12.3.1.7 CODE EXAMPLE. The following code provides an example of how to initialize TIMER 1 and how to use the timer for counting time-out periods.

TIMER MODULE

```
MBARx equ $10000      ;Defines the base of the module base
TMR1 EQU MBARx+$140   ;Timer 1 register
TMR2 EQU MBARx+$180   ;Timer 2 register
TRR1 EQU MBARx+$144   ;Timer 1 reference register
TRR2 EQU MBARx+$184   ;Timer 2 reference register
TCR1 EQU MBARx+$148   ;Timer 1 capture register
TCR2 EQU MBARx+$188   ;Timer 2 capture register
TCN1 EQU MBARx+$14C   ;Timer 1 counter register
TCN2 EQU MBARx+$18C   ;Timer 2 counter register
TER1 EQU MBARx+$151   ;Timer 1 event register
TER2 EQU MBARx+$191   ;Timer 2 event register

* TMR1 is defined as: *
*[15:8]=$FF,          divide clock by 256
*[7:6] = '11'         disable interrupt
*[5] = '1'           output=active-low pulse
*[4] = '0',          disable ref.interrupt
*[3] = '1',          restart mode enabled
*[2:1] = '10',       master clock/16
*[0] = '0',          timer1 enabled

    move.w  #$FF2C,D0
    move.w  D0,TMR1

    move.w  #$0000,D0      ;writing to the timer counter with any
    move.w  D0,TCN1       ;value resets it to zero

    move.w  #AFAF,D0      ;set the timer 1 reference to be
    move.w  #D0,TRR1      ;defined as $AFAF
```

The code below provides a simple example of using Timer 1. It simply counts time-out loops. A time-out occurs when the reference value is hit. In this example, the reference value was defined to be \$AFAF.

```
timer1_ex
    clr.l   D0
    clr.l   D1
    clt.l   D2

    move.w  #$0000,D0
    move.w  D0,TCN1      ;reset the counter to $0000

    move.b  #$03,D0      ;writing "1" to the "REF" & "CAP" bits of
    move.b  D0,TER1      ;the TER register clears the event flags

    move.w  TMR1,D0      ;save the contents of TMR1 while setting
    bset    #0,D0        ;the 0 bit. This enables Timer 1 and starts counting
    move.w  D0, TMR1     ;load the value back into the register

T1_LOOP
    move.b  TER1,D1      ;load the timer1 event register and
    btst   #1,D1        ;see if bit 1 (REF) has been set
    beq    T1_LOOP
```

```

addi.l    #1,D2        ;Increment D2
cmp.l     #5,D2        ;Has D2 reached 5 yet (i.e. timer ref has timed)
beq       T1_FINISH   ;If so, end timer1 example. Otherwise jump back

move.b    #$02,D0      ;writing "1" to the "REF" bit in
move.b    D0,TER1      ;the TER register clears the event flag
jmp       T1_LOOP

T1_FINISH
HALT      ;END PROCESSING. EXAMPLE IS FINISHED

```


SECTION 13

DMA CONTROLLER MODULE

13.1 INTRODUCTION

The Direct Memory Access Controller (DMA) Module provides a quick and efficient process for moving blocks of data with minimal processor overhead. The DMA module, shown in Figure 13-1, provides four channels that allow byte, word, or longword operand transfers. These transfers can be single or dual address to off-chip devices or dual address to on-chip devices.

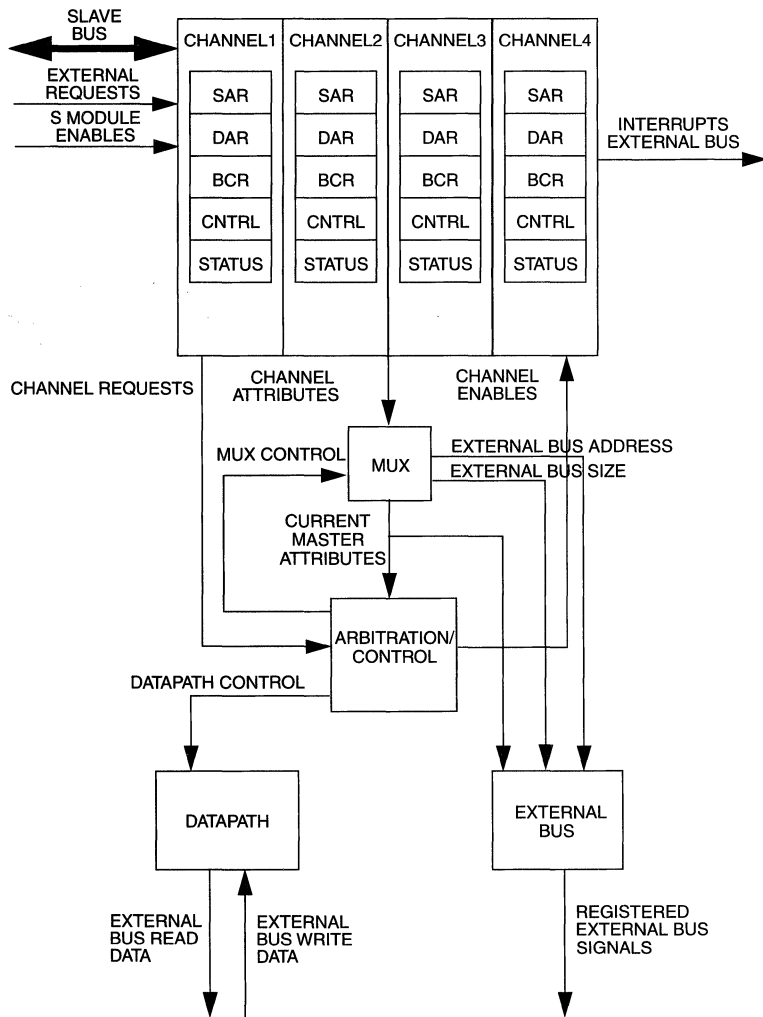


Figure 13-1. DMA Signal Diagram

The DMA contains the following features:

- Four fully independent programmable DMA Controller Module channels/slave bus modules
- Auto-alignment feature for source or destination accesses
- Single and dual address transfers
- Two external request pins provided
- Channel arbitration on transfer boundaries

- Data transfers in 8-, 16-, 32- or 128-bit blocks via a 16-byte buffer
- Supports burst and cycle steal transfers
- Independent transfer widths for source and destination
- Independent source and destination address registers
- Provide two clock data transfers

13.2 DMA SIGNAL DESCRIPTION

This subsection contains a brief description of the DMA module signals that provide handshake control for either a source or destination external device. See Table 13-1 for details.

Table 13-1. DMA Signals.

SIGNAL NAME	DIRECTION	DESCRIPTION
DREQ[1:0]	In	External DMA request

13.2.1 DMA Request ($\overline{\text{DREQ}}[1:0]/\text{PP}[6:5]$)

These multiplexed pins can serve as the DMA request inputs, or as two bits of the parallel port. Programming the Pin Assignment Register (PAR) in the SCM determines the function of each of these three multiplexed pins. You can program these pins on a bit-by-bit basis.

These active-low inputs are asserted by a peripheral device to request an operand transfer between that peripheral and memory.

The $\overline{\text{DREQ}}$ signals are asserted to initiate DMA accesses in the respective channels. The system should force any unused $\overline{\text{DREQ}}$ signals to a logic high state. Although each channel has an individual $\overline{\text{DREQ}}$ pin, in the MCF5307 implementation only the $\overline{\text{DREQ}}$ for channel 0 and 1 go to external pins. The $\overline{\text{DREQ}}$ for channel 2 and channel 3 are connected to the internal slave bus interrupt pins of UART0 and UART1 respectively.

13.3 DMA MODULE OVERVIEW

The DMA controller module transfers data at very high rates, usually much faster than the ColdFire core under software control can handle. The term “DMA” refers to a peripheral device’s capability to access memory in a system in the same manner as a microprocessor does. DMA operations can greatly increase overall system performance.

The DMA module consists of four independent channels. The term DMA is used throughout this section to reference any of the four channels as they are all functionally equivalent. Therefore, it is impossible to implicitly address all four DMA channels at the same time. The MCF5307 on-chip peripherals do not support the single-address transfer mode.

DMA requests can be internally generated by the processor writing to the start bit or externally generated by a device. The processor can program the amount of bus bandwidth allocated for the DMA for each channel. The DMA channels support two transfer modes: continuous mode and cycle steal mode.

The DMA controller supports single- and dual-address transfers. In single-address mode, a channel supports 32 bits of address and 32 bits of data. Single-address transfers can be started by an external device using the request signal. The DMA provides address and control signals during a single-address transfer. The requesting device either sends or receives data to or from the specified address (see Figure 13-2). In dual-address mode, a channel supports 32 bits of address and 32 bits of data. The dual-address transfers can be started by either the internal request mode or by an external device using the request signal. In this mode, two bus transfers occur, one from a source device and the other to a destination device (see Figure 13-3).

Any operation involving the DMA will follow the same basic steps: channel initialization, data transfer, and channel termination. In the channel initialization step, the DMA channel registers are loaded with control information, address pointers, and a byte transfer count. The channel is then started. During the data transfer step, the DMA accepts requests for operand transfers and provides addressing and bus control for the transfers. The channel termination step occurs after operation is complete. The channel indicates the status of the operation in the channel status register.

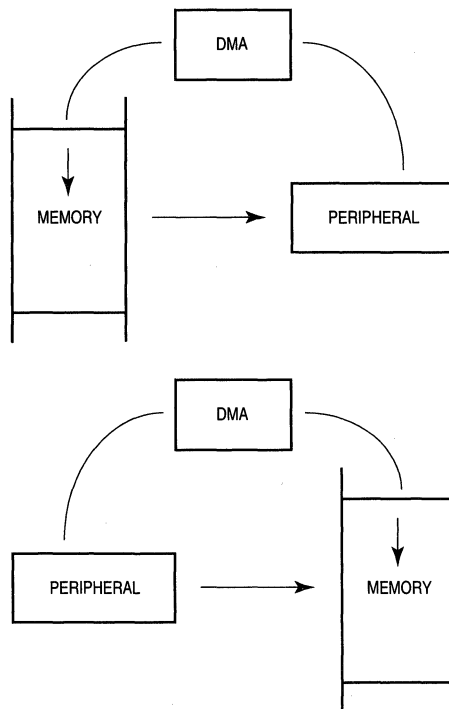


Figure 13-2. Single-Address Transfers

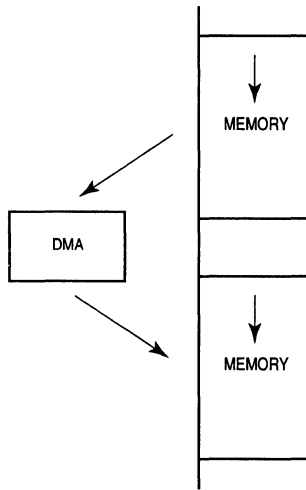


Figure 13-3. Dual-Address Transfer

13.4 DMA CONTROLLER MODULE PROGRAMMING MODEL

The registers of each DMA Controller Module channel are mapped into memory as shown in Figure 13-5. The base address for each channel of the DMA Controller Module is displayed in Figure 13-4.

Base Address Offset	Channel
MBAR + DMA1SAR	Channel 0
MBAR + DMA2SAR	Channel 1
MBAR + DMA3SAR	Channel 2
MBAR + DMA4SAR	Channel 3

Figure 13-4. DMA Controller Module Channel Offsets

The DMA Controller Module register set controls the DMA Controller module. This subsection describes each of the internal registers and the bit assignment for each register. Note that there is no mechanism for the prevention of writes to control registers during DMA transfers.

Base Address Offset	[31:0]	
\$300	Source Address Register 0	
\$304	Destination Address Register 0	
\$308	DMA Control Register 0	Reserved
\$30C	Byte Count Register 0	Reserved
\$310	Status Register 0	Reserved
\$314	Interrupt Vector Register 0	Reserved
\$340	Source Address Register 1	
\$344	Destination Address Register 1	

Figure 13-5. DMA Controller Module Register Model Per Channel

DMA CONTROLLER MODULE

Base Address Offset	[31:0]	
\$348	DMA Control Register 1	Reserved
\$34C	Byte Count Register 1	Reserved
\$350	Status Register 1	Reserved
\$354	Interrupt Vector Register 1	Reserved
\$380	Source Address Register 2	
\$384	Destination Address Register 2	
\$388	DMA Control Register 2	Reserved
\$38C	Byte Count Register 2	Reserved
\$390	Status Register 2	Reserved
\$394	Interrupt Vector Register 2	Reserved
\$3C0	Source Address Register 3	
\$3C4	Destination Address Register 3	
\$3C8	DMA Control Register 3	Reserved
\$3CC	Byte Count Register 3	Reserved
\$3D0	Status Register 3	Reserved
\$3D4	Interrupt Vector Register 3	Reserved

Figure 13-5. DMA Controller Module Register Model Per Channel

13.4.1 Source Address Register (SAR)

The source address register (SAR) is a 32-bit register containing the address from which the DMA Controller Module will request data during a transfer.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SAR31	SAR30	SAR29	SAR28	SAR27	SAR26	SAR25	SAR24	SAR23	SAR22	SAR21	SAR20	SAR19	SAR18	SAR17	SAR16
Reset:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAR15	SAR14	SAR13	SAR12	SAR11	SAR10	SAR9	SAR8	SAR7	SAR6	SAR5	SAR4	SAR3	SAR2	SAR1	SAR0
Reset:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Source Address Register

In Single Address Mode, the SAR provides the address regardless of the direction.

13.4.2 Destination Address Register (DAR)

The destination address register (DAR) is a 32-bit register containing the address to which the DMA Controller Module will send data during a transfer. Note that this register is only used during dual address transfers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DAR31	DAR30	DAR29	DAR28	DAR27	DAR26	DAR25	DAR24	DAR23	DAR22	DAR21	DAR20	DAR19	DAR18	DAR17	DAR16
Reset:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAR15	DAR14	DAR13	DAR12	DAR11	DAR10	DAR9	DAR8	DAR7	DAR6	DAR5	DAR4	DAR3	DAR2	DAR1	DAR0
Reset:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Destination Address Register (DAR)

13.4.3 Byte Count Register (BCR)

The byte count register (BCR) is a 16-bit register containing the number of bytes remaining to be transferred for a given block. The BCR count is the number of bytes remaining to be written.

The BCR decrements on the successful completion of the address phase of either a write transfer in dual address mode or any transfer in single address mode. The amount the BCR decrements is 1, 2, 4, or 16 for byte, word, longword, or line accesses, respectively.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCR15	BCR14	BCR13	BCR12	BCR11	BCR10	BCR9	BCR8	BCR7	BCR6	BCR5	BCR4	BCR3	BCR2	BCR1	BCR0
Reset:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Byte Count Register (BCR)

The DONE bit in the DMA Status Register is set when the entire block transfer is complete.

When a transfer sequence is initiated and the BCR contains a value that is not divisible by 16, 4, or 2 when the DMA is configured for line, longword, or word transfers, respectively, the configuration error bit in the DMA status register (DSR) is set and the transfer is not performed.

13.4.4 DMA Control Register

The DMA control register (DCR) is a 16-bit register that controls the configuration of the DMA Controller Module.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT	EEXT	CS	AA	BWC			SAA	S_RW	SINC	SSIZE		DINC	DSIZE		START
Reset:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INT—Interrupt on completion of transfer.

This field controls whether an interrupt is to be generated at the completion of the transfer or occurrence of an error condition.

- 1 = An internal interrupt signal asserts at the completion of a transfer.
- 0 = No interrupt is generated.

EEXT—Enable External Request.

- 1 = Enables the external request signal to affect transfer initiation. The START bit is always enabled.
- 0 = The external request is ignored.

NOTE

There is no logic precluding collisions with START bit and \overline{DREQ} signal except for EEXT. Use caution when initiating a DMA transfer with the START bit while EEXT=1.

CS—Cycle Steal.

- 1 = Forces a single read/write transfer per request. The request may be internal by setting the START bit, or external by asserting the \overline{DREQ} signal.
- 0 = The DMA performs continuous read/write transfers until the BCR decrements to 0

AA—Auto-Align

This bit and the size bits determine whether the source or destination is auto-aligned. Auto alignment means that the accesses will be optimized based on the address value and the programmed size. For more information see **13.7.2.2 Auto Alignment**.

- 1 = If the SSIZE bits indicate a larger or equivalent transfer size with respect to DSIZE, then the source accesses are auto-aligned. If the DSIZE bits indicate a larger transfer size than SSIZE, then the destination accesses are auto-aligned. Source alignment takes precedence over destination alignment. If auto-alignment is enabled, the appropriate address register increments, regardless of the state of DINC or SINC.
- 0 = No accesses are auto-aligned

BWC—Bandwidth Control

These three bits are decoded to provide for internal bandwidth control. When the byte count has reached the programmed BWC boundary, the request signal to the internal ar-

biter is negated until the completion of the data access to enable the arbiter to allow another master access to the bus. Table 13-2 shows the encodings for these bits. When the bits are cleared, the DMA does not negate its request. The 000 encoding will assert a priority signal when the channel is active, signaling that the transfer has been programmed for a higher priority. The table shows BCR values at which the bus is relinquished. For example, if BWC = 001 and the BCR is set to 516, the bus is relinquished after four bytes are transferred.

Table 13-2. BWC Encoding

BWC	BLOCK SIZE
000	DMA has priority
001	512
010	1024
011	2048
100	4096
101	8192
110	16384
111	32768

SAA—Single Address Access

- 1 = The DMA channel is in single address mode. The DMA provides an address from the SAR and directional control, bit S_RW, to allow two peripherals (one may be memory) to exchange data within a single access. Data will not be stored by the DMA.
- 0 = The DMA channel is in dual address mode

S_RW—Single Address Access Read/Write Value.

This bit specifies the value of the master read signal during single address accesses. This provides directional control to the master bus controller.

- 1 = Forces the master read signal to a logic high state
- 0 = Forces the master read signal to a logic low state

The bit is only valid when the SAA bit is set.

SINC—Source Increment

This bit controls whether the source address increments after each successful transfer.

- 1 = The SAR increments by 1, 2, 4, or 16, depending upon the size of the transfer
- 0 = There is no change to the SAR after a successful transfer

SSIZE—Source Size

This field controls the size of the source bus cycle that the DMA Controller Module is running. See Table 13-3 for the encoding of this field.

Table 13-3. SSIZE Encoding

SSIZE	TRANSFER SIZE
00	Longword
01	Byte
10	Word
11	Line

DINC—Destination Increment

This bit controls whether the destination address increments after each successful transfer.

- 1 = The DAR increments by 1, 2, 4, or 16 depending upon the size of the transfer
- 0 = There is no change to the DAR after a successful transfer

DSIZE—Destination Size

This field controls the size of the destination bus cycle that the DMA Controller Module is running. See Table 13-4 for the encoding of this field.

Table 13-4. DSIZE Encoding

DSIZE	TRANSFER SIZE
00	Longword
01	Byte
10	Word
11	Line

START—Start Transfer

- 1 = Indicates to the DMA to begin the transfer according to the values in the control registers

This bit is self-clearing after one clock and is always read as a logic 0.

13.4.5 DMA Status Register (DSR)

The DMA Status Register (DSR) is an 8-bit register that reports on the status of the DMA Controller Module. On recognition of an event, the DMA Controller Module sets the corresponding bit in the DSR. Only writes to bit 0 of the DSR will have any effect. Setting the DONE bit creates a single-cycle pulse, which will reset the channel thus clearing **all** bits in the register. This must be done at the completion of a transfer, though it may be set during a transfer to abort the transfer.

7	6	5	4	3	2	1	0
-	CE	BES	BED	-	REQ	BSY	DONE
Reset:							
-	0	0	0	-	0	0	0

Bit 7—Reserved

CE—Configuration Error

A configuration error results when either the number of bytes represented by the BCR is not consistent with the requested source or destination transfer size, or the SAR or DAR

contains an address that does not match the requested transfer size for the source or destination, respectively. The bit is cleared during a hardware reset, or by writing a logic one to the DONE bit of the DSR.

- 1 = A configuration error has occurred
- 0 = No configuration error exists

BES—Bus Error on Source

- 1 = The DMA channel has terminated with a bus error either during the read portion of a transfer or during an access in single address mode (SAA = 1)
- 0 = No bus error has occurred

BED—Bus Error on Destination

- 1 = The DMA channel has terminated with a bus error during the write portion of a transfer
- 0 = No bus error has occurred

Bit 3 —Reserved**REQ—Request**

- 1 = The DMA channel has transfers remaining and the channel is not selected
- 0 = There is no request pending or the channel is currently active. The bit is cleared when the channel is selected.

BSY—Busy

- 1 = This bit is set the first time the channel is enabled after a transfer is initiated
- 0 = DMA channel is not active. This bit is cleared to 0 when the DMA has finished the last transaction

DONE—Transaction Done

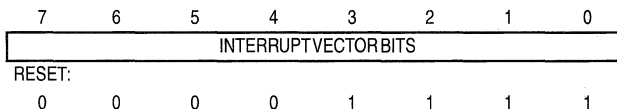
This bit may be read or written and is set when all DMA Controller Module transactions have completed normally, as determined by the transfer count or error conditions. When the BCR reaches zero, DONE is set at the successful conclusion of the final DMA Controller Module transfer

This bit is written with a 1 to reset the DMA control/state bits and can be used in an interrupt handler to clear the DMA interrupt and the DONE and error bits. It can also be used to kill a transfer in progress by resetting state bits. Writing a 0 to this location has no affect.

- 1= DMA transfer is complete
- 0= Writing or reading a 0 at this bit location has no affect.

13.4.6 DMA Interrupt Vector Register

The DMA Interrupt Vector Register (DIVR) is an 8-bit register, which is driven out onto the slave bus in response to an acknowledge cycle.



DMA Interrupt Vector Register

13.5 TRANSFER REQUEST GENERATION

The DMA channel supports two types of request generation methods: internal and external. You can program both types of requests to limit the amount of bus use and can be either cycle-steal mode or continuous mode. The EEXT field in the DCR in the programs the request-generation method used for the channel.

13.5.1 Cycle-Steal Mode

When the CS field in the DCR is set, the DMA is in cycle-steal mode. This means that only one complete transfer from source to destination will take place for each request that the module receives. The request can be either internal or external depending on how the EEXT field is programmed.

13.5.2 Continuous Mode

If the CS field in the DCR is cleared, the DMA is in continuous mode. After a request is asserted, either internal or external, the DMA will continuously transfer data until the BCR is zero or the DONE bit in the DSR is set.

13

The continuous mode can be run at either the maximum rate or a limited rate. The maximum rate of transfer can be achieved if the BWC field in the DCR is programmed to be 000. Then the DMA channel that has started a transfer will continue until the BCR decrements to zero or a 1 is written to the DONE bit in the DSR.

A limited rate can be achieved by programming the BWC field to be anything except 000. The DMA will then perform the specified number of transfers and surrender the bus to allow another device to use the bus. In this mode, the DMA will negate its internal bus request on the last transfer before the boundary programmed in the BWC field. After the transfer is complete, it will then assert its bus request again to regain mastership at the earliest possible time as determined by the internal bus arbiter. The minimum amount of time that the DMA will not have the bus is one bus cycle.

13.6 DATA TRANSFER MODES

Each DMA channel supports single- and dual-address transfers. The single-address transfer mode consists of one DMA bus cycle, which allows either a read or a write cycle to occur. The dual-address transfer mode consists of a source operand read and a destination operand write.

13.6.1 Single-Address Transactions

The DMA Controller Module begins a single address transfer sequence when the SAA bit is set while a DMA request is made. If no error conditions exist, the REQ bit is set. When the channel is enabled, the BSY bit is set and the REQ bit is cleared. The SAR contents are then driven onto the master address bus and the value of the S_{RW} bit is driven onto the master read signal. The BCR decrements on successful address phase accesses until it reaches 0, and the DONE bit is set.

In the event of a termination error, the BES and DONE bit of the DSR are set and no further DMA Controller Module transactions are attempted.

13.6.2 Dual-Address Transactions

The DMA Controller Module begins a dual-address transfer sequence when the SAA bit is cleared while a DMA request is made. If no error condition exists, the REQ bit of the DSR is set.

13.6.2.1 DUAL-ADDRESS READS. The DMA Controller Module will drive the value in the SAR onto the internal address bus. If the SINC bit of the DCR is set, then the SAR increments by the appropriate number of bytes upon a successful read cycle. When the appropriate number of read cycles completes successfully, the DMA initiates the write portion of the transfer.

In the event of a termination error, the BES and DONE bit of the DSR are set, and no further DMA Controller Module transactions are attempted.

13.6.2.2 DUAL-ADDRESS WRITES. The DMA Controller Module will drive the value in the DAR onto the master address bus. If the DINC bit of the DCR is set, the 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. If the BCR equals zero, the DONE bit is set. If the BCR is greater than 0, then another read/write transfer is initiated. If the BCR is a multiple of the programmed BWC, then the DMA request signal is negated until termination of the bus cycle to allow the internal arbiter to switch masters.

In the event of a termination error, the BED and DONE bit of the DSR are set, and no further DMA Controller Module transactions are attempted.

13.7 DMA CONTROLLER MODULE FUNCTIONAL DESCRIPTION

In the following descriptions, “DMA request” implies that the START bit is set or the \overline{DREQ} signal is asserted while the EEXT bit is set. The START bit is cleared when the channel begins an internal access. Before initiating a transfer request, the DMA Controller Module first verifies that the source size and destination size (dual address only), as configured in the DCR, are consistent with the source address and destination address. If a misalignment is detected, no transfer will occur and the CE bit of the DSR will be set.

The CE bit is also set if the BCR contains a value inconsistent with both the destination size (dual address only) and the source size. Depending on the configuration of the DCR, an

interrupt event may be issued when the CE bit is set. Note that if the AA bit is set, error checking is performed only on the appropriate registers.

A “read/write” transfer refers to a dual-address access in which a number of bytes are read from the source address and written to the destination address. The number of bytes transferred is determined by the larger of the sizes specified by the source and destination size encodings.

The source and destination address registers (SAR and DAR) increment at the completion of a successful address phase. The BCR decrements at the completion of a successful address phase write when SAA=0 or any successful address phase when SAA=1. A successful address phase occurs when a valid address request is not held by the arbiter.

13.7.1 Channel Initialization and Startup

Before starting a block transfer operation, the channel registers must be initialized with information describing the channel configuration, request-generation method, and data block. This initialization is accomplished by programming the appropriate information into the channel registers.

13.7.1.1 CHANNEL PRIORITIZATION. The four DMA channels are prioritized in ascending order or as determined by the BWC bits in the DCR. If the BWC bits for a DMA channel are set to 000, then that channel has priority over the channel immediately preceding it. For example, if DMA channel 2 has the BWC bits set to 000, it has priority over DMA channel 1 but not over DMA channel 2. This is assuming that DMA channel 1 has something other than all zeroes in the BWC bits.

Another example would be a case where the BWC bits in DMA 2 and DMA 1 are all zeroes. In this case, DMA 1 would have priority over DMA 0 and DMA 2. The BWC bits being zero in DMA 2 in this case have no effect on prioritization.

13

In the case of simultaneous external requests, the prioritization is either ascending or as determined by each channels BWC bits as described in the previous paragraphs.

13.7.1.2 PROGRAMMING THE DMA CONTROLLER MODULE. The DMA Controller Module is programmed via the slave bus in the ColdFire architecture. The DMA Controller Module monitors every slave bus cycle, and when the module enables assert, the proper action is taken to perform a read or a write, as requested by the slave bus controller. Some general comments on programming the DMA follow:

- No mechanism exists for preventing writes to control registers during DMA accesses
- If the BWC of sequential channels are equivalent, 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 a peripheral device or memory to memory, the source address is the starting address of the data block. This address can be any byte address. In the single-address mode, this register is used regardless of the 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, the DAR is loaded with the address of the peripheral data register. This address can be any byte address. In the single-address mode, this register is not used.

The manner in which the SAR and DAR change after each cycle depends on the values in the DCR SSIZE and DSIZE fields and the SINC and DINC bits, and the starting address in the SAR and DAR. If programmed to increment, the increment value is 1, 2, 4, or 16 for byte, word, longword, or line operands, respectively. If the address register is programmed to remain unchanged (no count), the register is not incremented after the operand transfer.

The BCR must be loaded with the number of byte transfers that are to occur. This register is decremented by 1, 2, 4, or 16 at the end of each transfer. The DSR must be cleared for channel startup.

Once the channel has been initialized, it is started by writing a one to the START bit in the DCR or asserting the $\overline{\text{DREQ}}$ signal, depending on the status of the EEXT bit in the DCR. 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.

If any fields in the DCR are modified while the channel is active, that change is effective immediately. To avoid any problems with changing the setup for the DMA channel, a 1 should be written to the DONE bit in the DSR to stop the DMA channel.

13.7.2 Data Transfers

13.7.2.1 EXTERNAL REQUEST OPERATION. Each channel has the feature of interfacing to an external module to initiate transfers to the module. In the MCF5307 device, the external requests for channel 1 and channel 2 are connected to external pins. The request for channel 3 and channel 4 are connected internally to the slave bus interrupt pins of the UART0 and UART1 modules, respectively. If the EEXT bit is set, when the $\overline{\text{DREQ}}$ signal asserts, the DMA will initiate a transfer provided the channel is idle. If the CS (cycle steal) bit is set, a single read/write transfer will occur on the master bus. If the CS bit is clear, multiple read/write transfers occur on the master bus as programmed. The transfer mode pins can be used to provide an external request acknowledge response. The $\overline{\text{DREQ}}$ signal is not required to be negated until the DONE bit of the DSR asserts. In cycle-steal mode, the maximum length of $\overline{\text{DREQ}}$ assertion to maintain a single transfer depends on configuration. In the worst case of a single-address access, byte accesses, and idle channels, $\overline{\text{DREQ}}$ may be asserted for no more than four rising clock edges (see Figure 13-6).

See Figure 13-7 for timing relationships for a dual-address transfer using cycle-steal mode. The maximum assertion time for $\overline{\text{DREQ}}$ in this configuration is five clocks.

When an access occurs that was initiated by an $\overline{\text{DREQ}}$ signal, the transfer mode signals will indicate an alternate master cycle, while the transfer modifier signals will indicate that the cycle is due to an external request. To create an external DMA acknowledge signal, it may be necessary to decode the address of the peripheral along with a combination of the

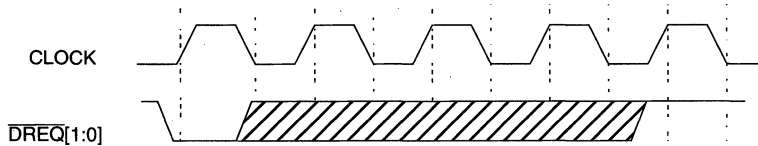


Figure 13-6. External Request Timing - Cycle-Steal Mode, Single-Address Mode

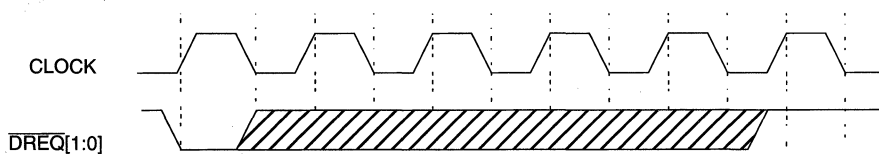


Figure 13-7. External Request Timing - Cycle-Steal Mode, Dual-Address Mode

transfer mode and transfer-type signals. To create an external DMA acknowledge signal for block transfers, you may count the transfers or compare the address to a stored final address to assert the DMA acknowledge to the peripheral.

13.7.2.2 AUTO ALIGNMENT. This feature allows for block transfers to occur at the most optimum size possible based on the address, byte count, and programmed size. To use this feature, AA in the DCR must be set. The source is auto-aligned when the SSIZE bits indicate a larger transfer size compared to 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 that is chosen for alignment increments regardless of the value of the increment bit. Configuration error checking is performed on the registers that are not chosen for alignment.

If the BCR contains a value greater than 16, the address will determine the size of the transfer. Single byte, word or longword transfers will occur until the address is aligned to the programmed size boundary, at which time the programmed size accesses begin. When the BCR is less than 16 at the beginning of a read/write transfer, the number of bytes remaining will dictate the transfer size, longword, word or byte.

For example,

AA = 1, SAR = \$0001, BCR = \$00f0, SSIZE = 00 (longword) and DSIZE = 01 (byte),

Because the SSIZE > DSIZE, the source is auto-aligned. Error checking is performed on the destination registers. The sequence of accesses is as follows:

1. Read byte from \$0001 - write byte, increment SAR
2. Read word from \$0002 - write 2 bytes, increment SAR
3. Read long word from \$0004 - write 4 bytes, increment SAR
4. Repeat longwords until SAR = \$00f0
5. Read byte from \$00f0 - write byte, increment SAR.

If DSIZE is set to another size, then the data writes are optimized to write the largest size allowed based on the address, but not exceeding the configured size.

13.7.2.3 BANDWIDTH CONTROL. This feature provides a mechanism that can force the DMA off the master bus, allowing another master access. This feature can simplify the master bus arbiter design by making arbitration programmable. The decode of the BWC provides 7 levels of block transfer sizes. If the BCR decrements to a value equivalent to the decode of the BWC, the DMA master bus request negates until termination of the bus cycle. The arbiter may then choose to switch the bus to another master, should a request be pending. Note that if AA is set, the BCR may skip over the programmed boundary. In this case, the DMA master bus request will not negate. If the BWC = 0, the request signal will remain asserted until the BCR reaches 0. In addition, an internal signal will assert to indicate that the channel has been programmed to have priority. Note that in this arbitration scheme, the arbiter can always force the DMA to relinquish the bus.

13.7.3 Channel Termination

13.7.3.1 ERROR CONDITIONS. When the DMA Controller Module encounters a read or write cycle that terminates with an error condition, the appropriate bit of the DSR is set, depending on whether the bus cycle was a read (BES) or a write (BED). The DMA transfers are then halted. If the error condition occurred during a write cycle, any data remaining in the internal holding register is lost.

13.7.3.2 INTERRUPTS. If the INT bit of the DCR is set, the DMA will drive the appropriate slave bus interrupt signal. A processor can then read the DSR to determine if the transfer terminated successfully or with an error. The DONE bit of the DSR is then written with a 1 to clear the interrupt, along with clearing the DONE and error bits.

SECTION 14

UART MODULES

The MCF5307 contains two universal asynchronous/synchronous receiver/transmitters (UARTs), which act independently. All references to “UART” refer to one of these instances. Each UART is clocked by the system clock, eliminating the need for an external crystal.

The UART module interfaces directly to the CPU. Shown in Figure 14-1, the UART module, consists of the following major functional areas:

- Serial Communication Channel
- Sixteen Bit Timer for Baud Rate Generation
- Internal Channel Control Logic
- Interrupt Control Logic

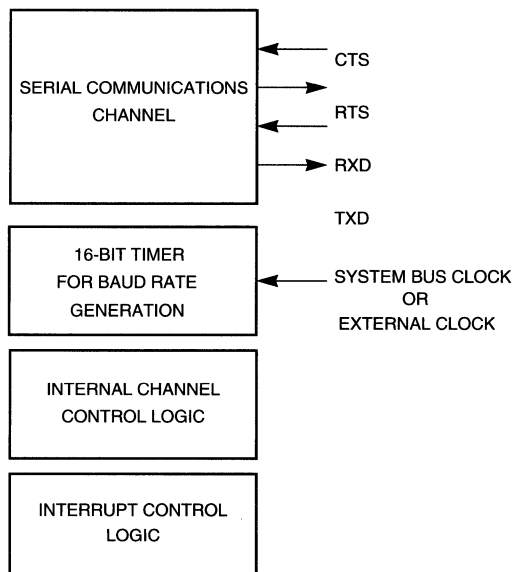


Figure 14-1. Simplified Block Diagram

14.1 SERIAL MODULE OVERVIEW

The MCF5307 contains two independent UART modules. Features of each UART module are as follows:

- Each UART is clocked by the system clock, eliminating the need for external crystal
- Full-Duplex Asynchronous/Synchronous Receiver/Transmitter Channel
- Quadruple-Buffered Receiver
- Double-Buffered Transmitter
- Independently Programmable Baud Rate for Receiver and Transmitter Selectable from:
 - Timer-Generated Baud Rate
- Programmable Data Format:
 - Five to Eight Data Bits Plus Parity
 - Odd, Even, No Parity, or Force Parity
 - One, One and a Half or Two Stop Bits
- Programmable Channel Modes:
 - Normal (Full Duplex)
 - Automatic Echo
 - Local Loopback
 - Remote Loopback
- Automatic Wakeup Mode for Multidrop Applications
- Four Maskable Interrupt Conditions
- 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

14.1.1 Serial Communication Channel

The communication channel provides a full-duplex asynchronous/synchronous receiver and transmitter using an operating frequency derived from the system clock.

The transmitter accepts parallel data from the CPU. It converts the data to a serial bit stream inserting the appropriate start, stop, and optional parity bits. Finally, it outputs a composite serial data stream on the channel transmitter serial data output (TxD). Refer to **Section 14.3.4.1 Transmitter** for additional information.

The receiver accepts serial data on the channel receiver serial data input (RxD), converts it to parallel format, checks for a start bit, stop bit, parity (if any), or break condition, and transfers the assembled character onto the bus during read operations. The receiver may be polled or interrupt driven. Refer to **Section 14.3.4.2 Receiver** for additional information.

14.1.2 Interrupt Control Logic

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 all (up to four) unmasked interrupt status bits in the interrupt status register (UISR).

The interrupt level of the UART module is programmed in the interrupt controller external to the UART module. The UART can be configured to supply the autovector for the interrupt level programmed, or to supply the vector from the UART Interrupt Vector Register (UIVR) when the UART interrupt is acknowledged.

The interrupt level, priority within the level, and auto-vectoring capability can all be programmed in the SIM register ICR12 for UART1 and ICR13 for UART2.

14.1.3 Comparison of UART Module to MC68681

The MCF5307 is code compatible with the MC68681, except that only channel A is implemented, and the system clock is used as a clock source in timer mode. The input and output port lines (IPx and OPx) are not implemented except for CTS and RTS functions.

14.2 UART MODULE SIGNAL DEFINITIONS

The following paragraphs contain a brief description of the UART module signals. Figure 14-2 shows both the external and internal signal groups.

NOTE

The terms *assertion* and *negation* are used throughout this section to avoid confusion when dealing with a mixture of active-low and active-high signals. The term *assert* or *assertion* indicates that a signal is active or true, independent of the level represented by a high or low voltage. The term *negate* or *negation* indicates that a signal is inactive or false.

14.2.1 Transmitter Serial Data Output (TxD)

The transmitter serial data output signal is held high ('mark' condition) when the transmitter is disabled, idle, or operating in the local loopback mode. Data is shifted out on this signal on the falling edge of the clock source, with the least significant bit transmitted first.

14.2.2 Receiver Serial Data Input (RxD)

Data received on the receiver serial data input signal is sampled on the rising edge of the clock source, with the least significant bit received first.

14.2.3 Clear-To-Send ($\overline{\text{CTS}}$)

This active-low input is the clear-to-send input. It can generate an interrupt on change-of-state.

14.2.4 Request-To-Send ($\overline{\text{RTS}}$)

$\overline{\text{RTS}}$ is an active-low output signal that can be programmed to be automatically negated and asserted by either the receiver or transmitter. When connected to the clear-to-send ($\overline{\text{CTS}}$) input of a transmitter, this signal can be used to control serial data flow. It is important to note that $\overline{\text{RTS}}$ of UART2 is muxed with RSTO. This muxing is controlled by the PAR register in the SIM.

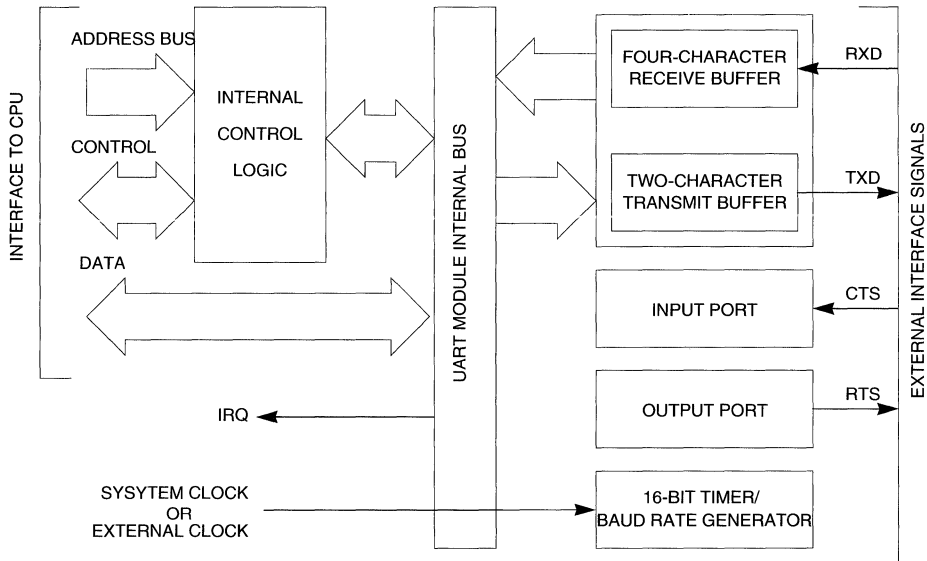


Figure 14-2. External and Internal Interface Signals

14.3 OPERATION

The following paragraphs describe the operation of the baud rate generator, transmitter and receiver, and other functional operating modes of the UART module.

14.3.1 Baud-Rate Generator Logic

The system clock serves as the basic timing reference for the Baud rate Generator Logic. The Baud rate Generator logic consists of a baud-rate generator and a programmable 16-bit timer. The baud-rate generator is used to generate standard baud rates if the system bus clock is 3.6864MHz. Since this is not normally the case, it is recommended to use the 16-bit timer as a programmable divider.

Note

The system bus clock is defined as the clock signal from the PLL (BCLK0).

The processor's main clock is driven into the PLL on the CLKIN pin. The PLL takes this clock and multiplies it by 2 (PSTCLK). Then, for the internal modules, the clock is programmable to be divided by 2, 3, or 4 (BCLK0).

For example, an external processor clock of 45 MHz would run the CPU core at 90 MHz and divide the 90 MHz clock by 2, 3, or 4. This would clock the internal modules (such as the UARTs) at 45, 30, or 22.5 MHz respectively. See **Section 4 Phase Lock Loop (PLL)** for more details.

14.3.2 Baud-Rate Generator/Timer

The 16-bit timer is used as a baud rate generator and provides a synchronous clock mode of operation when used as a divide-by-1 clock and an asynchronous clock mode when used as a divide-by-16 clock. This allows flexible baud rates for the various system clock rates, the divisor value being directly programmable. The UARTs are clocked by the internal system clock or an external clock signal on the TIN pins.

Note

When using an external clock, TIN0 feeds UART1 and TIN1 feeds UART2. The UARTs function independently of each other and can be clocked independently (external clock or system bus clock).

The operation of the 16-bit timer does not affect the operation of the UART. However, if the TIN pin is used as a clocking source for either the Timer or UART, the timer will not be able to use the TIN pin for Timer capture.

14.3.3 Calculating Baud Rates

14.3.3.1 SYSTEM BUS CLOCK . When the system bus clock is chosen as the clocking source for the UARTs, the system bus clock goes through a divide by 32 prescaler. This signal then passes through the 16 bit divider of the concatenated UBG1 and UBG2 prescale registers. Assuming a 45 MHz system bus clock, baud rate calculations would be as follows:

$$\begin{aligned} \text{baud rate} &= (45 \text{ MHz}) / [(32) * (16 \text{ bit divider})] \\ 9600 \text{ appx} &= (45 \text{ MHz}) / [(32) * (146 \text{ decimal})] \\ 146 \text{ decimal} &= \$01A0 \text{ hex therefore UBG1} = \$01 \text{ and UBG2} = \$A0 \end{aligned}$$

See Figure 14-3 for details.

14.3.3.2 EXTERNAL CLOCK. When an external clock pin is chosen as the clocking source for the UARTs, the external clock signal goes through a programmable divide by 16 or divide by 1 prescaler. This signal then passes through the 16 bit divider of the concatenated UBG1 and UBG2 registers.

$$\text{baud rate} = (\text{external clock frequency}) / [(16 \text{ or } 1) * (16 \text{ bit divider})]$$

See Figure 14-3 for details.

14.3.4 Transmitter and Receiver Operating Modes

The functional block diagram of the transmitter and receiver is shown in Figure 14-3. Included therein, are the command and operating registers. The following paragraphs contain descriptions for both of these functions in reference to this diagram. For detailed register information, refer to **Section 14.4 Register Description and Programming**.

14.3.4.1 TRANSMITTER. The transmitter is enabled through its command register (UCR) located within the UART module. The UART module signals the CPU when it is ready to accept a character by setting the transmitter-ready bit (TxRDY) in the UART status register (USR). Functional timing information for the transmitter is shown in Figure 14-4. 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 least significant bit is sent first. Data is shifted from the transmitter output on the falling edge of the clock source.

Following transmission of the stop bits; if a new character is not available in the transmitter holding register, the TxD output remains high ('mark' condition), and the transmitter empty bit (TxEMP) in the USR is set. Transmission resumes and the TxEMP bit is cleared when the CPU loads a new character into the transmitter buffer (UTB). If a disable command is sent to the transmitter, it continues operating until the character in the transmit shift register, if any, is completely sent out. If the transmitter is reset through a software command, operation ceases immediately (refer to **Section 14.4.1.5 Command Register (UCR)**). The transmitter is re-enabled through the UCR to resume operation after a disable or software reset.

If the clear-to-send operation is enabled, the $\overline{\text{CTS}}$ signal 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 transmitted, and TxD remains in the 'mark' state until $\overline{\text{CTS}}$ is asserted again. If the transmitter is forced to send a continuous low condition by issuing a send break command, the state of $\overline{\text{CTS}}$ is ignored by the transmitter.

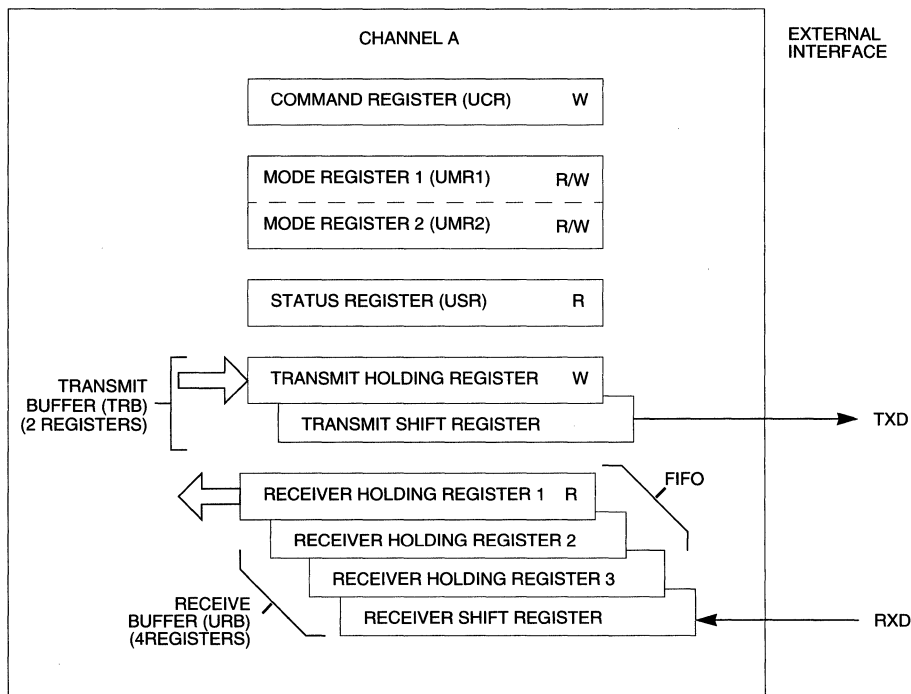
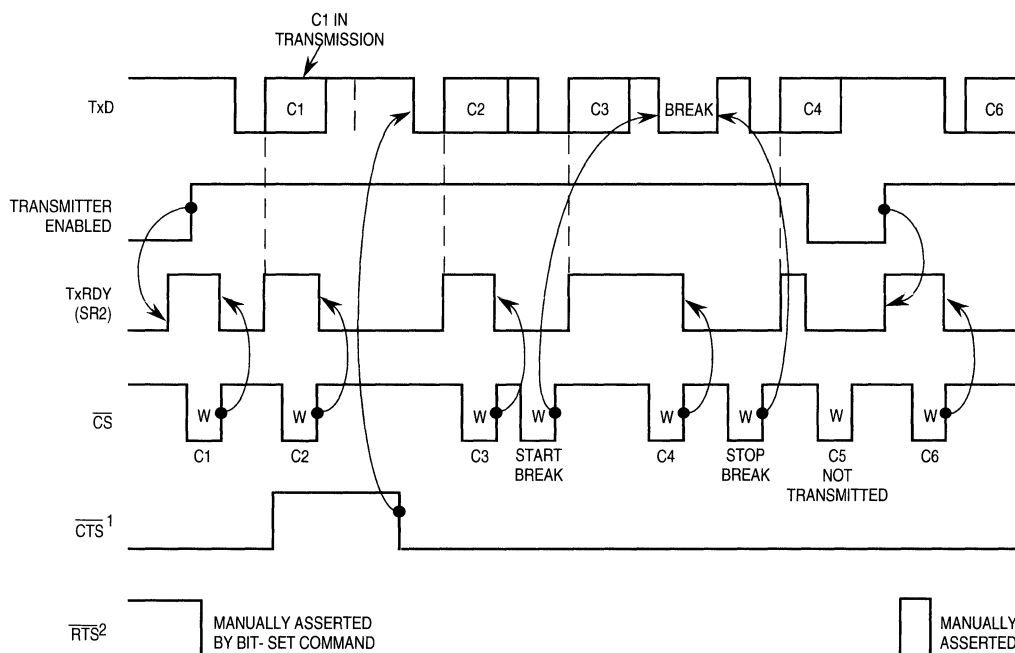


Figure 14-3. Transmitter and Receiver Functional Diagram

14

The transmitter can be programmed to automatically negate the request-to-send ($\overline{\text{RTS}}$) output upon completion of a message transmission. If the transmitter is programmed to operate in this mode, $\overline{\text{RTS}}$ must be manually asserted before a message is transmitted. 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 re-enabled by reasserting $\overline{\text{RTS}}$ before the next message is to be sent.



NOTES:

1. TIMING SHOWN FOR UMR2(4) = 1
2. TIMING SHOWN FOR UMR2(5) = 1
3. C_N = TRANSMIT CHARACTER
4. W = WRITE

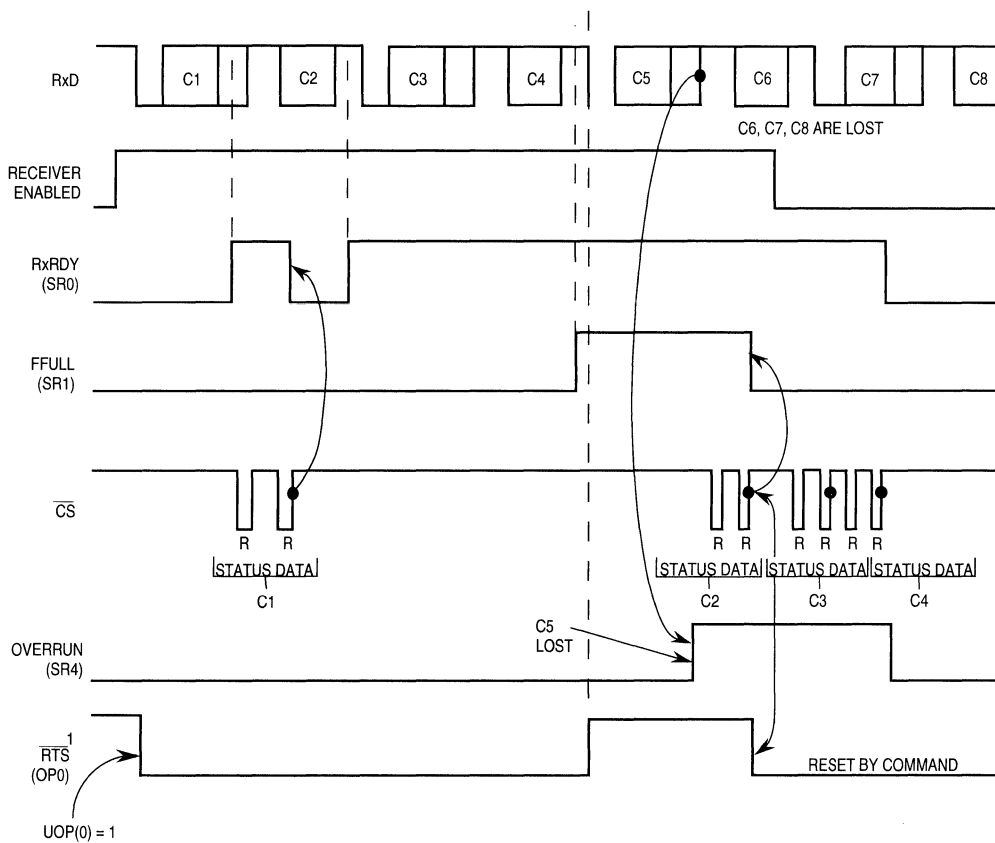
Figure 14-4. Transmitter Timing Diagram

14.3.4.2 RECEIVER. The receiver is enabled through its UCR located within the UART module. The functional timing information for the receiver is shown in Figure 14-5. The receiver looks for a high-to-low (mark-to-space) transition of the start bit on RxD. When a transition is detected, 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 to sample 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 least significant bit is received first. The data is then transferred to a receiver holding register, and the RxRDY bit in the USR is set. If the character length 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 is detected. The parity error (PE), framing error (FE), overrun error (OE), and received break (RB) conditions (if any) set error and break flags in the USR at the received character boundary and are valid only when the RxRDY bit in the USR 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 the RB and RxRDY bits in the USR are set. The RxD signal 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. When the break begins in the middle of a character, the receiver places the damaged character in the receiver first-in-first-out (FIFO) stack and sets the corresponding error conditions and RxRDY bit in the USR. Then, if the break persists until the next character time, the receiver places an all-zero character into the receiver FIFO and sets the corresponding RB and RxRDY bits in the USR.



NOTES:

1. Timing shown for MR1(7) = 1
2. Timing shown for MR1(6) = 0
3. R = Read
4. C_N = Received Character

Figure 14-5. Receiver Timing Diagram

14.3.4.3 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 (refer to Figure 14-3). 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, PE (Parity Error), FE (Framing Error), and RB (Received Break), 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), status is provided in character or block modes.

The RxRDY bit in the USR is set whenever one or more characters are 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 new data can be added at the bottom of the stack by the receiver shift register. 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.

In the character mode, status provided in the USR is given on a character-by-character basis and thus applies only to the character at the top of the FIFO. In the block mode, the status provided in the USR is the logical OR of all characters coming to the top of the FIFO stack since the last reset error command. A continuous logical OR function of the corresponding status bits is produced in the USR as each character reaches the top of the FIFO stack. The block mode is useful in applications where the software overhead of checking each character's error cannot be tolerated. In this mode, entire messages are received, and only one data integrity check is performed at the end of the message. This mode allows a data-reception speed advantage, but does have a disadvantage since each character is not individually checked for error conditions by software. If an error occurs within the message, the error is not recognized until the final check is performed, and no indication exists as to which character in the message is at fault.

In either mode, reading the USR does not affect the FIFO. The FIFO is 'popped' only when the receive buffer is read. The USR should be read prior to reading the receive buffer. If all three of the FIFO's receiver holding registers are full when a new character is received, the new character is held in the receiver shift register until a FIFO position is available. If an additional character is received during this state, the contents of the FIFO are not affected. However, the character previously in the receiver shift register is lost, and the OE (Overflow Error) bit in the USR is set when the receiver detects the start bit of the new overrun character.

To support control flow capability, the receiver can be programmed to automatically negate and assert RTS. When in this mode, RTS is automatically negated by the receiver when a valid start bit is detected and the FIFO stack is full. When a FIFO position becomes available, RTS is asserted by the receiver. Using this mode of operation, overrun errors are prevented by connecting the RTS to the CTS input of the transmitting device.

NOTE

In order to use the RTS or CTS signals, the port B control register must be set up to enable the corresponding I/O pins for these functions. If the FIFO stack contains characters and the receiver is disabled, the characters in the FIFO can still be read by the CPU. If the receiver is reset, the FIFO stack and all receiver status bits, corresponding output ports, and interrupt request are reset. No additional characters are received until the receiver is re-enabled.

14.3.5 Looping Modes

The UART can be configured to operate in various looping modes as shown in Figure 14-6. These modes are useful for local and remote system diagnostic functions. The modes are described in the following paragraphs with further information available in **Section 14.4 Register Description and Programming**.

The UART's transmitter and receiver should both 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.

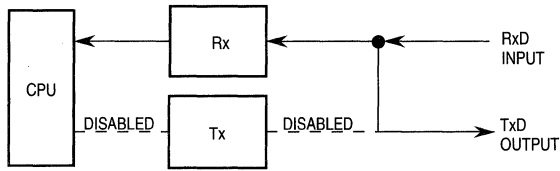
14.3.5.1 AUTOMATIC ECHO MODE. In the automatic echo mode, the UART automatically retransmits the received data on a bit-by-bit basis. The local CPU-to-receiver communication continues normally, but the CPU-to-transmitter link is disabled. While in this mode, received data is clocked on the receiver clock and retransmitted on TxD. The receiver must be enabled, but the transmitter need not be enabled.

Since the transmitter is not active, the TxEMP and TxRDY bits in USR are inactive, and data is transmitted as it is received. Received parity is checked, but not recalculated for transmission. Character framing is also checked, but stop bits are transmitted as received. A received break is echoed as received until the next valid start bit is detected.

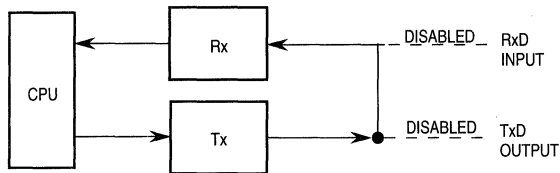
14.3.5.2 LOCAL LOOPBACK MODE. TxD is internally connected to RxD in the local loopback mode. This is useful for testing the operation of a local UART module channel by sending data to the transmitter and checking data assembled by the receiver. In this manner, correct channel operations can be assured. Also, both transmitter and CPU-to-receiver communications continue normally in this mode. While in this mode, the RxD input data is ignored, the TxD is held marking, and the receiver is clocked by the transmitter clock. The transmitter must be enabled, but the receiver need not be enabled.

14.3.5.3 REMOTE LOOPBACK MODE. In this mode, the channel automatically transmits received data on the TxD output on a bit-by-bit basis. The local CPU-to-transmitter link is disabled. This mode is useful in testing receiver and transmitter operation of a remote channel. While in this mode, the receiver clock is used for the transmitter.

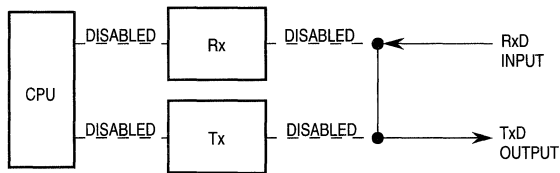
Since the receiver is not active, received data cannot be read by the CPU, and the error status conditions are inactive. Received parity is not checked and is not recalculated for transmission. Stop bits are transmitted as received. A received break is echoed as received until the next valid start bit is detected.



(a) Automatic Echo



(b) Local Loopback



(c) Remote Loopback

Figure 14-6. Looping Modes Functional Diagram

14.3.6 Multidrop Mode

The UART can be programmed to operate in a wakeup mode for multidrop or multiprocessor applications. Functional timing information for the multidrop mode is shown in Figure 14-7. The mode is selected by setting bits 3 and 4 in mode register 1 (UMR1). This mode of operation allows the master station to be connected to several slave stations (maximum of 256). In this mode, the master transmits an address character followed by a block of data characters targeted for one of the slave stations. The slave stations have their channel receivers disabled. However, they continuously monitor the data stream sent out by the master station. When an address character is sent by the master, the slave receiver channel notifies its respective CPU by setting the RxRDY bit in the USR 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 to monitor the data stream for the next address character. Data fields in the data stream are separated by an address character. After a slave receives a block of data, the slave station's CPU disables the receiver and initiates the process again.

A transmitted character 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. The A/D bit identifies the type of character being transmitted to the slave station. The character is interpreted as an address character if the A/D bit is set or as a data character if the A/D bit is cleared. The polarity of the A/D bit is selected by programming bit 2 of UMR1. UMR1 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 a zero (data tag). If the receiver is enabled, all received characters are transferred to the CPU via 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 bit 5). 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.

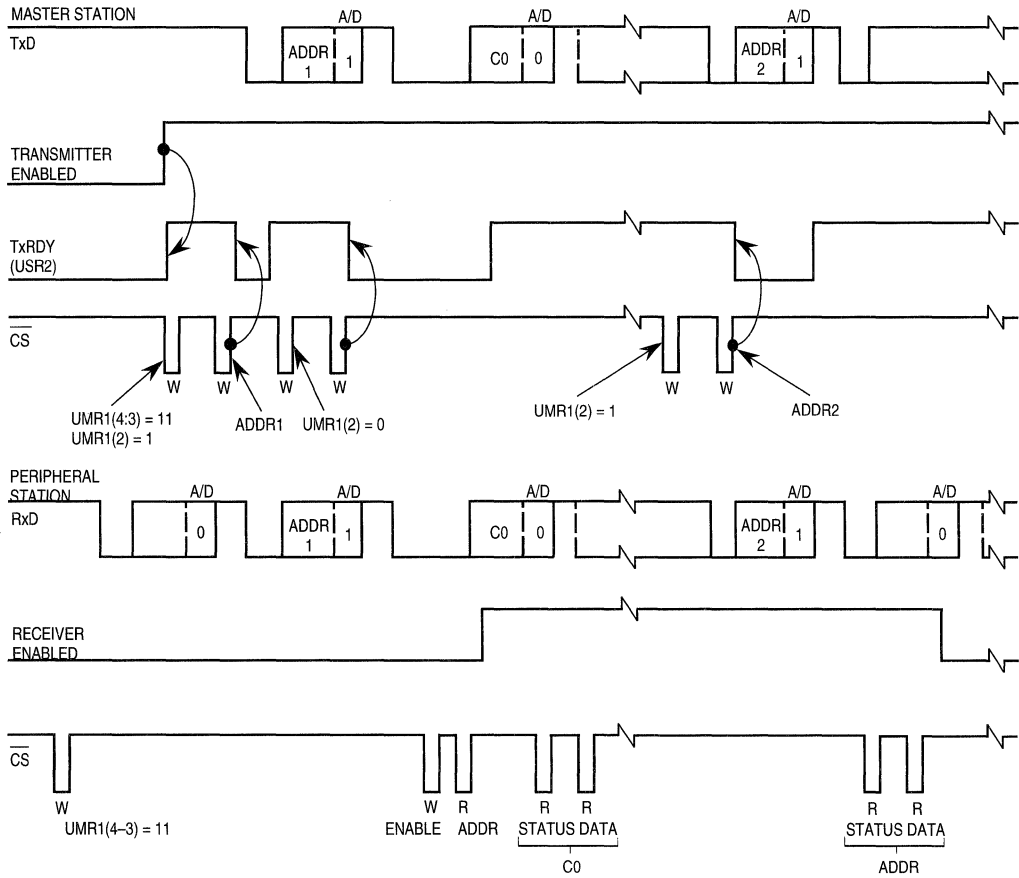


Figure 14-7. Multidrop Mode Timing Diagram

14.3.7 Bus Operation

This section describes the operation of the bus during read, write, and interrupt acknowledge cycles to the UART module. All UART module registers must be accessed as bytes.

14.3.7.1 READ CYCLES. The UART module responds to reads with byte data on D7–D0. Reserved registers return logic zero during reads.

14.3.7.2 WRITE CYCLES. The UART module accepts write data on D7–D0. Write cycles to read-only registers and reserved registers complete in a normal manner without exception processing; however, the data is ignored.

NOTE

The UART module is accessed by the CPU with zero wait states, as the system bus clock is used for the UART module.

14.3.7.3 INTERRUPT ACKNOWLEDGE CYCLES. The UART module is capable of arbitrating for interrupt servicing and supplying the interrupt vector when it has successfully won arbitration. The vector number must be provided if interrupt servicing is necessary; thus, the interrupt vector register (UIVR) must be initialized. If the UIVR is not initialized, a spurious interrupt exception is taken if interrupts are generated. This works in conjunction with the interrupt controller, which allows a programmable IPL for the interrupt.

14.4 REGISTER DESCRIPTION AND PROGRAMMING

This section contains a detailed description of each register and its specific function as well as flowcharts of basic UART module programming.

14.4.1 Register Description

The operation of the UART module is controlled by writing control bytes into the appropriate registers. A list of UART module registers and their associated addresses is shown in Table 14-1.

NOTE

All UART module registers are only accessible as bytes. The contents of the mode registers (UMR1 and UMR2), clock-select register (UCSR), and the auxiliary control register (UACR) bit 7 should only be changed after the receiver/transmitter is issued a software RESET command—i.e., channel operation must be disabled. Care should also be taken if the register contents are changed during receiver/transmitter operations, as undesirable results may be produced.

In the registers discussed in the following pages, the numbers above the register description represent the bit position in the register. The register description contains the mnemonic for the bit. The values shown below the register description are the values of those register bits

after a hardware reset. A value of U indicates that the bit value is unaffected by reset. The read/write status is shown in the last line.

Table 14-1. UART Module Programming Model

UART1	UART2	REGISTER READ (R/W = 1)	REGISTER WRITE (R/W = 0)
MBAR+\$1C0	MBAR+\$200	MODE REGISTER (UMR1, UMR2)	MODE REGISTER (UMR1, UMR2)
MBAR+\$1C4	MBAR+\$204	STATUS REGISTER (USR)	CLOCK-SELECT REGISTER (UCSR)
MBAR+\$1C8	MBAR+\$208	DO NOT ACCESS ¹	COMMAND REGISTER (UCR)
MBAR+\$1CC	MBAR+\$20C	RECEIVER BUFFER (URB)	TRANSMITTER BUFFER (UTB)
MBAR+\$1D0	MBAR+\$210	INPUT PORT CHANGE REGISTER (UIPCR)	AUXILIARY CONTROL REGISTER (UACR)
MBAR+\$1D4	MBAR+\$214	INTERRUPT STATUS REGISTER (UISR)	INTERRUPT MASK REGISTER (UIMR)
MBAR+\$1D8	MBAR+\$218	BAUD RATE GENERATOR PRESCALE MSB (UBG1)	
MBAR+\$1DC	MBAR+\$21C	BAUD RATE GENERATOR PRESCALE LSB (UBG2)	
		DO NOT ACCESS ¹	
MBAR+\$1F0	MBAR+\$230	INTERRUPT VECTOR REGISTER (UIVR)	INTERRUPT VECTOR REGISTER (UIVR)
MBAR+\$1F4	MBAR+\$234	INPUT PORT REGISTER (UIP)	DO NOT ACCESS ¹
MBAR+\$1F8	MBAR+\$238	DO NOT ACCESS ¹	OUTPUT PORT BIT SET CMD (UOP1) ²
MBAR+\$1FC	MBAR+\$23C	DO NOT ACCESS ¹	OUTPUT PORT BIT RESET CMD (UOP0) ²

NOTES

1. This address is used for factory testing and should not be read. Reading this location results in undesired effects and possible incorrect transmission or reception of characters. Register contents may also be changed.
2. Address-triggered commands.

14.4.1.1 MODE REGISTER 1 (UMR1). UMR1 controls some of the UART module configuration. This register can be read or written at any time. It is accessed when the mode register pointer points to UMR1. The pointer is set to UMR1 by RESET or by a set pointer command using the MISC[2:1] bits in the UCR control register. The pointer will point to the UMR1 register after reading or writing UMR1, the pointer points to UMR2.

14

UMR1 MBAR + \$1C0, \$200

7	6	5	4	3	2	1	0
RxRTS	RxIRQ	ERR	PM1	PM0	PT	B/C1	B/C0

RESE
T:

0 0 0 0 0 0 0 0

Read/Write

Supervisor or User

(RxRTS)—Receiver Request-to-Send

- 1 = Upon receipt of a valid start bit, $\overline{\text{RTS}}$ is negated if the UART's FIFO is full. $\overline{\text{RTS}}$ is reasserted when the FIFO has an empty position available.
- 0 = The receiver has no effect on $\overline{\text{RTS}}$.

This feature can be used for flow control to prevent overrun in the receiver by using the $\overline{\text{RTS}}$ output to control the $\overline{\text{CTS}}$ input of the transmitting device. If both the receiver and transmitter are programmed for $\overline{\text{RTS}}$ control, $\overline{\text{RTS}}$ control is disabled for both since this configuration is incorrect. See **Section 14.4.1.2 Mode Register 2 (UMR2)** for information on programming the transmitter $\overline{\text{RTS}}$ control.

(RxIRQ) — Receiver Interrupt Select

- 1 = FFULL is the source that generates IRQ.
- 0 = RxRDY is the source that generates IRQ.

(ERR) — Error Mode

This bit controls the meaning of the three FIFO status bits (RB, FE, and PE) in the USR.

- 1 = Block mode—The values in the channel USR are the accumulation (i.e., the logical OR) of the status for all characters coming to the top of the FIFO since the last reset error status command for the channel was issued. Refer to **Section 14.4.1.5 Command Register (UCR)** for more information on UART module commands.
- 0 = Character mode—The values in the channel USR reflect the status of the character at the top of the FIFO.

NOTE

ERR = 0 must be used to get the correct A/D flag information when in multidrop mode.

(PM1–PM0) — Parity Mode

These bits encode the type of parity used for the channel (see Table 14-2). The parity bit is added to the transmitted character, and the receiver performs a parity check on incoming data. These bits can alternatively select multidrop mode for the channel.

(PT) — Parity Type

This bit selects the parity type if parity is programmed by the parity mode bits, and if multidrop mode is selected, it configures the transmitter for data character transmission or ad-

dress character transmission. Table 14-2 lists the parity mode and type or the multidrop mode for each combination of the parity mode and the parity type bits.

Table 14-2. PMx and PT Control Bits

PM1	PM0	PARITY MODE	PT	PARITY TYPE
0	0	With Parity	0	Even Parity
0	0	With Parity	1	Odd Parity
0	1	Force Parity	0	Low Parity
0	1	Force Parity	1	High Parity
1	0	No Parity	X	No Parity
1	1	Multidrop Mode	0	Data Character
1	1	Multidrop Mode	1	Address Character

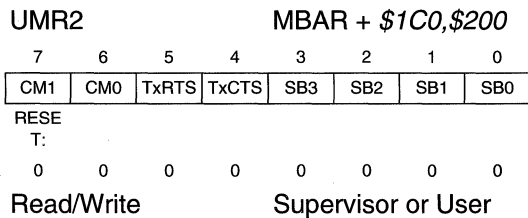
(B/C1–B/C0) — Bits per Character

These bits select the number of data bits per character to be transmitted. The character length listed in Table 14-3 does not include start, parity, or stop bits.

Table 14-3. B/Cx Control Bits

B/C1	B/C0	BITS/CHARACTER
0	0	Five Bits
0	1	Six Bits
1	0	Seven Bits
1	1	Eight Bits

14.4.1.2 MODE REGISTER 2 (UMR2). UMR2 controls some of the UART module configuration. It is accessed when the mode register pointer points to UMR2, which occurs after any access to UMR1. Accesses to UMR2 do not change the pointer.



(CM1–CM0) — Channel Mode

These bits select a channel mode as listed in Table 14-4. See **Section 14.3.5 Looping Modes** for more information on the individual modes.

Table 14-4. CMx Control Bits

CM1	CM0	MODE
0	0	Normal
0	1	Automatic Echo
1	0	Local Loopback
1	1	Remote Loopback

(TxRTS) — Transmitter Ready-to-Send

This bit controls the negation of the $\overline{\text{RTS}}$ signal.

- 1 = In applications where the transmitter is disabled after transmission is complete, setting this bit causes the particular OP bit to be cleared automatically one bit time after the characters, if any, in the channel transmit shift register and the transmitter holding register are completely transmitted, including the programmed number of stop bits. This feature is used to automatically terminate transmission of a message. If both the receiver and the transmitter in the same channel are programmed for $\overline{\text{RTS}}$ control, $\overline{\text{RTS}}$ control is disabled for both since this is an incorrect configuration.
- 0 = The transmitter has no effect on $\overline{\text{RTS}}$.

(TxCTS) — Transmitter Clear-to-Send

- 1 = Enables clear-to-send operation. The transmitter checks the state of the $\overline{\text{CTS}}$ input each time it is ready to send a character. If $\overline{\text{CTS}}$ is asserted, the character is transmitted. If $\overline{\text{CTS}}$ is negated, the channel TxD remains in the high state, and the transmission is delayed until $\overline{\text{CTS}}$ is asserted. Changes in $\overline{\text{CTS}}$ while a character is being transmitted do not affect transmission of that character. If both TxCTS and TxRTS are enabled, TxCTS controls the operation of the transmitter.
- 0 = The $\overline{\text{CTS}}$ has no effect on the transmitter.

(SB3–SB0) — Stop-Bit Length Control

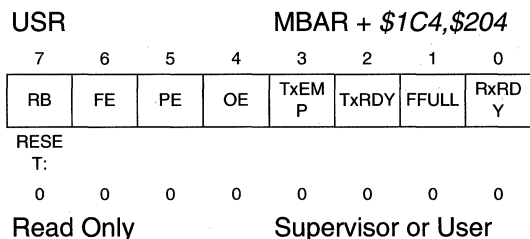
These bits select the length of the stop bit appended to the transmitted character as listed in Table 14-5. Stop-bit lengths of nine-sixteenth to two bits, in increments of one-sixteenth bit, are programmable for character lengths of six, seven, and eight bits. For a character length of five bits, one and one-sixteenth to two bits are programmable in increments of one-sixteenth bit. In all cases, the receiver only checks for a high condition at the center of the first stop-bit position—i.e., one bit time after the last data bit or after the parity bit, if parity is enabled.

If an external 1× clock is used for the transmitter, UMR2 bit 3 = 0 selects one stop bit, and UMR2 bit 3 = 1 selects two stop bits for transmission.

Table 14-5. SBx Control Bits

SB3	SB2	SB1	SB0	LENGTH 6-8 BITS	LENGTH 5 BITS
0	0	0	0	0.563	1.063
0	0	0	1	0.625	1.125
0	0	1	0	0.688	1.188
0	0	1	1	0.750	1.250
0	1	0	0	0.813	1.313
0	1	0	1	0.875	1.375
0	1	1	0	0.938	1.438
0	1	1	1	1.000	1.500
1	0	0	0	1.563	1.563
1	0	0	1	1.625	1.625
1	0	1	0	1.688	1.688
1	0	1	1	1.750	1.750
1	1	0	0	1.813	1.813
1	1	0	1	1.875	1.875
1	1	1	0	1.938	1.938
1	1	1	1	2.000	2.000

14.4.1.3 STATUS REGISTER (USR). The USR indicates the status of the characters in the FIFO and the status of the transmitter and receiver.



(RB) — Received Break

1 = An all-zero character of the programmed length has been received without a stop bit. The RB bit is only valid when the RxRDY bit is set. 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 internal or external 1× clock or 16 successive edges of the external 16× clock.

The received break circuit detects breaks that originate in the middle of a received character. However, if a break begins in the middle of a character, it must persist until the end of the next detected character time.

0 = No break has been received.

(FE) — Framing Error

- 1 = A stop bit was not detected when the corresponding data character in the FIFO was received. The stop-bit check is made in the middle of the first stop-bit position. The bit is valid only when the RxRDY bit is set.
- 0 = No framing error has occurred.

(PE) — Parity Error

- 1 = When the with parity or force parity mode is programmed in the UMR1, the corresponding character in the FIFO was received with incorrect parity. When the multidrop mode is programmed, this bit stores the received A/D bit. This bit is valid only when the RxRDY bit is set.
- 0 = No parity error has occurred.

(OE) — Overrun Error

- 1 = One or more characters in the received data stream have been lost. This bit 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. This bit is cleared by the reset error status command in the UCR.
- 0 = No overrun has occurred.

(TxEMP) — Transmitter Empty

- 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.
- 0 = The transmitter buffer is not empty. Either a character is currently being shifted out, or the transmitter is disabled. The transmitter is enabled/disabled by programming the TCx bits in the UCR.

(TxRDY) — Transmitter Ready

- 1 = The transmitter holding register is empty and ready to be loaded with a character. This bit is set when the character is transferred to the transmitter shift register. This bit is also set when the transmitter is first enabled. Characters loaded into the transmitter holding register while the transmitter is disabled are not transmitted.
- 0 = The transmitter holding register was loaded by the CPU, or the transmitter is disabled.

(FFULL) — FIFO Full

- 1 = A character has been received and is waiting in the receiver buffer FIFO.
- 0 = The FIFO is not full, but may contain up to two unread characters.

(RxDY) — Receiver Ready

- 1 = One or more characters has been received and is waiting in the receiver buffer FIFO.
- 0 = The CPU has read the receiver buffer, and no characters remain in the FIFO after this read.

14.4.1.4 CLOCK-SELECT REGISTER (UCSR). The UCSR selects the system bus clock or a divide by 16 or 1 external clock. The upper 4 bits set the receiver and the lower 4 bits set the transmitter clock source. To use the system bus clock for both the transmitter and receiver, the UCSR should be programmed with \$DD. The transmitter and receiver can be programmed with different clock sources.

UCSR				MBAR + \$1C4,\$204			
7	6	5	4	3	2	1	0
RCS3	RCS2	RCS1	RCS0	TCS3	TCS2	TCS1	TCS0
RESE				T:			
0	0	0	0	0	0	0	0
Write Only				Supervisor or User			

(RCS3–RCS0) — Receiver Clock Select

These bits select the clock source for the receiver channel. Table 14-6 details the register bits necessary for each mode.

Table 14-6. RCS[3:0] Control Bits

RCS3	RCS2	RCS1	RCS0	CLOCK
1	1	0	1	TIMER
1	1	1	0	x16 ext. clk.
1	1	1	1	x1 ext. clk.

(TCS3–TCS0) — Transmitter Clock Select

These bits select the clock source for the transmitter channel. Table 14-7 details the register bits necessary for each mode

Table 14-7. TCS[3:0] Control Bits

TCS3	TCS2	TCS1	TCS0	CLOCK
1	1	0	1	TIMER
1	1	1	0	x16 ext. clk.
1	1	1	1	x1 ext. clk.

14.4.1.5 COMMAND REGISTER (UCR). The UCR is used to supply commands to the UART. Multiple commands can be specified in a single write to the UCR if the commands are not conflicting – e.g., reset transmitter and enable transmitter commands cannot be specified in a single command.

UCR	MBAR + \$1C8,\$208						
7	6	5	4	3	2	1	0
—	MISC2	MISC1	MISC0	TC1	TC0	RC1	RC0
RESE							
T:							
0	0	0	0	0	0	0	0
Write Only				Supervisor or User			

(MISC2–MISC0) — Miscellaneous Commands

These bits select a single command as listed in Table 14-8.

Table 14-8. MISC[2:0] Control Bits

MISC2	MISC1	MISC0	COMMAND
0	0	0	No Command
0	0	1	Reset Mode Register Pointer
0	1	0	Reset Receiver
0	1	1	Reset Transmitter
1	0	0	Reset Error Status
1	0	1	Reset Break-Change Interrupt
1	1	0	Start Break
1	1	1	Stop Break

Reset Mode Register Pointer

The reset mode register pointer command causes the mode register pointer to point to UMR1.

Reset Receiver

With this command, the receiver is immediately disabled, the FFULL and RxRDY bits in the USR are cleared, and the receiver FIFO pointer is reinitialized. All other registers are unaltered. This command should be used in lieu of the receiver disable command whenever the receiver configuration is changed because it places the receiver in a known state.

Reset Transmitter

The reset transmitter command resets the transmitter by, immediately disabling the transmitter, and clearing the TxEMP and TxRDY bits in the USR. All other registers are unaltered. This command should be used in lieu of the transmitter disable command whenever

DUART

the transmitter configuration is changed because it places the transmitter in a known state.

Reset Error Status

The reset error status command clears the RB, FE, PE, and OE bits (in the USR). This command is also used in the block mode to clear all error bits after a data block is received.

Reset Break

Change Interrupt—The reset break-change interrupt command clears the delta break (DBx) bits in the UISR.

Start Break

The start break command forces TxD low. If the transmitter is empty, the start of the break conditions can be delayed up to one bit time. If the transmitter is active, the break begins when transmission of the character is complete. If a character is in the transmitter shift register, the start of the break is delayed until the character is transmitted. If the transmitter holding register has a character, that character is transmitted after the break. The transmitter must be enabled for this command to be accepted. The state of the $\overline{\text{CTS}}$ input is ignored for this command.

Stop Break

The stop break command causes TxD to go high (mark) within two bit times. Characters stored in the transmitter buffer, if any, are transmitted.

(TC1–TC0) — Transmitter Commands

These bits select a single command as listed in Table 14-9.

Table 14-9. TC[1:0] Control Bits

TC1	TC0	COMMAND
0	0	No Action Taken
0	1	Enable Transmitter
1	0	Disable Transmitter
1	1	Do Not Use

No Action Taken

The no action taken command causes the transmitter to stay in its current mode. If the transmitter is enabled, it remains enabled; if disabled, it remains disabled.

Transmitter Enable

The transmitter enable command enables operation of the channel's transmitter. The TxEMP and TxRDY bits in the USR are also set. If the transmitter is already enabled, this command has no effect.

Transmitter Disable

The transmitter disable command terminates transmitter operation and clears the TxEMP and TxRDY bits in the USR. However, if a character is being transmitted when the transmitter is disabled, the transmission of the character is completed before the transmitter becomes inactive. If the transmitter is already disabled, this command has no effect.

Do Not Use

Do not use this bit combination because the result is indeterminate.

(RC1–RC0) — Receiver Commands

These bits select a single command as listed in Table 14-10.

Table 14-10. RCx Control Bits

RC1	RC0	COMMAND
0	0	No Action Taken
0	1	Enable Receiver
1	0	Disable Receiver
1	1	Do Not Use

No Action Taken

The no action taken command causes the receiver to stay in its current mode. If the receiver is enabled, it remains enabled; if disabled, it remains disabled.

Receiver Enable

If the UART module is not in multidrop mode, The receiver enable command enables operation of the channel's receiver. This command also forces the receiver into the search-for-start-bit state. If the receiver is already enabled, this command has no effect.

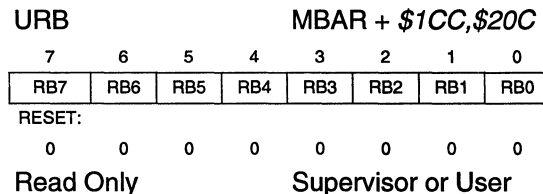
Receiver Disable

The receiver disable command disables the receiver immediately. Any character being received is lost. The command has no effect on the receiver status bits or any other control register. If the UART module is programmed to operate in the local loopback mode or multidrop mode, the receiver operates even though this command is selected. If the receiver is already disabled, this command has no effect.

Do Not Use

Do not use this bit combination because the result is indeterminate.

14.4.1.6 RECEIVER BUFFER (URB). The receiver buffer contains three receiver holding registers and a serial shift register. The RxD pin is connected to the serial shift register. The holding registers act as a FIFO. The CPU reads from the top of the stack while the receiver shifts and updates from the bottom of the stack when the shift register has been filled (see Figure 14-3).



(RB7–RB0) — These bits contain the character in the receiver buffer.

14.4.1.7 TRANSMITTER BUFFER (UTB). The transmitter buffer consists of two registers, the transmitter holding register and the transmitter shift register (see Figure 14-2). The holding register accepts characters from the bus master if the TxRDY bit in the channel's USR is set. A write to the transmitter buffer clears the TxRDY bit, inhibiting any more characters until the shift register is ready to accept more data. When the shift register is empty, it checks to see if the holding register has a valid character to be sent (TxRDY bit cleared). If there is a valid character, the shift register loads the character and reasserts the TxRDY bit in the USR. Writes to the transmitter buffer when the channel's USR TxRDY bit is clear and when the transmitter is disabled have no effect on the transmitter buffer.

UTB				MBAR + \$1CC,\$20C			
7	6	5	4	3	2	1	0
TB7	TB6	TB5	TB4	TB3	TB2	TB1	TB0
RESET:							
0	0	0	0	0	0	0	0
Write Only				Supervisor or User			

(TB7–TB0) — These bits contain the character in the transmitter buffer.

14.4.1.8 INPUT PORT CHANGE REGISTER (UIPCR). The UIPCR shows the current state and the change-of-state for the $\overline{\text{CTS}}$ pin.

UIPCR				MBAR + \$1D0,\$210			
7	6	5	4	3	2	1	0
0	0	0	COS	1	1	1	$\overline{\text{CTS}}$
RESET:							
0	0	0	0	0	1	1	$\overline{\text{CTS}}$
Read Only				Supervisor or User			

Bits 7, 6, 5, 3, 2, 1 — Reserved by Motorola.

(COS) — Change-of-State

- 1 = A change-of-state (high-to-low or low-to-high transition), lasting longer than 25–50 μs has occurred at the corresponding IPx input. When these bits are set, the UACR can be programmed to generate an interrupt to the CPU.
- 0 = No change-of-state has occurred since the last time the CPU read the UIPCR. A read of the UIPCR also clears the UISR COS bit.

($\overline{\text{CTS}}$) — Current State

Starting two serial clock periods after reset, the $\overline{\text{CTS}}$ bit reflects the state of the $\overline{\text{CTS}}$ pin. If the $\overline{\text{CTS}}$ pin is detected as asserted at that time, the COS bit is set, which initiates an interrupt if the IEC bit of the UACR register is enabled.

- 1 = The current state of the $\overline{\text{CTS}}$ input is logic one.
- 0 = The current state of the $\overline{\text{CTS}}$ input is logic zero.

14.4.1.9 AUXILIARY CONTROL REGISTER (UACR). The UACR selects which baud rate is used and controls the handshake of the transmitter/receiver.

UACR				MBAR + \$1D0,\$210			
7	6	5	4	3	2	1	0
BRG	CTMS 2	CTMS 1	CTMS 0	—	—	—	IEC
RESET:							
0	0	0	0	0	0	0	0
Write Only				Supervisor or User			

(BRG) — Baud Rate Generator Set Select

1 = Set 2 of the available baud rates is selected.

0 = Set 1 of the available baud rates is selected. Refer to **Section 14.4.1.4 Clock-select Register (UCSR)** for more information on the baud rates.

(CTMS2–CTMS0) — Timer Mode and Source Select

Table 14-11 shows the timer mode and source select bit fields.

Table 14-11. Timer Mode and Source Select Bits

CTMS2	CTMS1	CTMS0	MODE COMMAND	CLOCK SOURCE SELECT COMMAND
1	1	0	Timer	System clock

NOTE: Other values are invalid and should not be used.

(IEC) — Input Enable Control

1 = UISR bit 7 is set and an interrupt is generated when the COS bit in the UIPCR is set by an external transition on the \overline{CTS} input (if bit 7 of the interrupt mask register (UIMR) is set to enable interrupts).

0 = Setting the corresponding bit in the UIPCR has no effect on UISR bit 7.

14.4.1.10 INTERRUPT STATUS REGISTER (UISR). The UISR provides status for all potential interrupt sources. The contents of this register are masked by the UIMR. If a flag in the UISR is set and the corresponding bit in UIMR is also set, the internal interrupt output is asserted. If the corresponding bit in the UIMR is cleared, the state of the bit in the UISR has no effect on the output.

NOTE

The UIMR does not mask reading of the UISR. True status is provided regardless of the contents of UIMR. The contents of UISR are cleared when the UART module is reset.

UISR				MBAR + \$1D4,\$214			
7	6	5	4	3	2	1	0
COS	—	—	—	—	DB	RxRDY	TxRDY
RESET:							
0	0	0	0	0	0	0	0
Read Only				Supervisor or User			

(COS) — Change-of-State

- 1 = A change-of-state has occurred at the $\overline{\text{CTS}}$ input and has been selected to cause an interrupt by programming bit 0 of the UACR.
- 0 = COS bit in the UIPCR is not selected.

(DB) — Delta Break

- 1 = The receiver has detected the beginning or end of a received break.
- 0 = No new break-change condition to report. Refer to **Section 14.4.1.5 Command Register (UCR)** for more information on the reset break-change interrupt command.

(RxRDY) — Receiver Ready or FIFO Full

The function of this bit is programmed by UMR1 bit 6. It is a duplicate of either the FFULL or RxRDY bit of USR.

(TxRDY) — Transmitter Ready

This bit is the duplication of the TxRDY bit in USR.

- 1 = The transmitter holding register is empty and ready to be loaded with a character.
- 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 transmitted.

14.4.1.11 INTERRUPT MASK REGISTER (UIMR). The UIMR selects the corresponding bits in the UISR that cause an interrupt. If one of the bits in the UISR is set and the corresponding bit in the UIMR is also set, the internal interrupt output is asserted. If the corre-

UIMR				MBAR + \$1D4,\$214			
7	6	5	4	3	2	1	0
COS	—	—	—	—	DB	FFULL	TxRDY
RESET:							
0	0	0	0	0	0	0	0
Write Only				Supervisor or User			

(COS) — Change-of-State

1 = Enable interrupt

0 = Disable interrupt

(DB) — Delta Break

1 = Enable interrupt

0 = Disable interrupt

(FFULL) — FIFO Full

1 = Enable interrupt

0 = Disable interrupt

(TxRDY) — Transmitter Ready

1 = Enable interrupt

0 = Disable interrupt

14.4.1.12 TIMER UPPER PRELOAD REGISTER (UBG1). This register holds the eight most significant bits of the preload value. This value divides the system bus clock to the UART in order to provide a given baud rate.

14.4.1.13 TIMER UPPER PRELOAD REGISTER (UBG2). This register holds the eight least significant bits of the preload value. This value divides the system bus clock to the UART in order to provide a given baud rate

NOTE

The minimum value that can be loaded on the concatenation of UBG1 with UBG2 is \$0002. Both (UBG1) and (UBG2) are write only and cannot be read by the CPU.

14.4.1.14 INTERRUPT VECTOR REGISTER (UIVR). The UIVR contains the 8-bit vector number of the internal interrupt.

UIVR				MBAR + \$1F0,\$230			
7	6	5	4	3	2	1	0
IVR7	IVR6	IVR5	IVR4	IVR3	IVR2	IVR1	IVR0
RESET:							
0	0	0	0	1	1	1	1
Read/Write				Supervisor or User			

(IVR7–IVR0) — Interrupt Vector Bits

This 8-bit number indicates the offset from the base of the vector table where the address of the exception handler for the specified interrupt is located. The UIVR is reset to \$0F, which indicates an uninitialized interrupt condition.

14.4.1.15 INPUT PORT REGISTER (UIP). The UIP register shows the current state of the $\overline{\text{CTS}}$ input.

UIP							MBAR + \$1F4,\$234
7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	$\overline{\text{CTS}}$
RESET:							
1	1	1	1	1	1	1	1
Read Only				Supervisor or User			

$\overline{\text{CTS}}$ — Current State

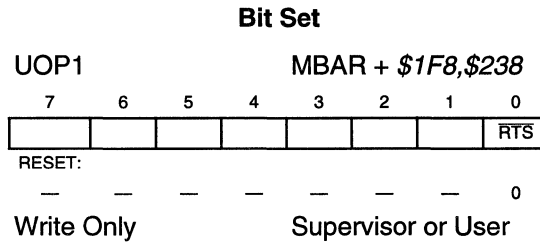
- 1 = The current state of the $\overline{\text{CTS}}$ input is logic one.
- 0 = The current state of the $\overline{\text{CTS}}$ input is logic zero.

The information contained in this bit is latched and reflects the state of the input pin at the time that the UIP is read.

NOTE

This bit has the same function and value as the UIPCR bit 0.

14.4.1.16 OUTPUT PORT DATA REGISTERS (UOP1, UOP0). The $\overline{\text{RTS}}$ output is set by performing a bit set command (writing to UOP1) and is cleared by performing a bit reset command (writing to UOP0).

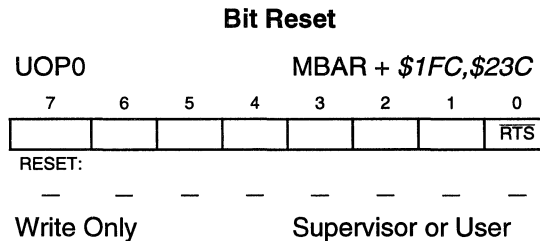


$\overline{\text{RTS}}$ — Output Port Parallel Output

- 1 = A write cycle to the OP bit set command address sets all OP bits corresponding to one bits on the data bus.
- 0 = These bits are not affected by writing a zero to this address.

NOTE

The output port bits are inverted at the pins, so the $\overline{\text{RTS}}$ set bit provides an asserted $\overline{\text{RTS}}$ pin.



$\overline{\text{RTS}}$ — Output Port Parallel Output

- 1 = A write cycle to the OP bit reset command address clears all OP bits corresponding to one bits on the data bus.
- 0 = These bits are not affected by writing a zero to this address.

14.4.2 Programming

The basic interface software flowchart required for operation of the UART module is shown in Figure 14-8. The routines are divided into three categories:

- UART Module Initialization
- I/O Driver
- Interrupt Handling

14.4.2.1 UART MODULE INITIALIZATION. The UART module initialization routines consist of SINIT and CHCHK. SINIT is called at system initialization time to check UART operation. Before SINIT is called, the calling routine allocates two words on the system stack. Upon return to the calling routine, SINIT passes information on the system stack to reflect the status of the UART. If SINIT finds no errors, the receiver and transmitter are enabled. The CHCHK routine performs the actual checks as called from the SINIT routine. When called, SINIT places the UART in the local loopback mode and checks for the following errors:

- Transmitter Never Ready
- Receiver Never Ready
- Parity Error
- Incorrect Character Received

14.4.2.2 I/O DRIVER EXAMPLE. The I/O driver routines consist of INCH and OUTCH. INCH is the terminal input character routine and gets a character from the receiver. OUTCH is used to send a character to the transmitter.

14.4.2.3 INTERRUPT HANDLING. The interrupt handling routine consists of SIRQ, 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 UART MODULE INITIALIZATION SEQUENCE

The following steps are required to properly initialize the UART module.

NOTE

The UART module registers can be accessed by word or byte operations, but only the data byte D7–D0 is valid.

Command Register (UCR)

- Reset the receiver and transmitter.
- Reset the mode pointer ($MISC[2:0] = "001"$)

Interrupt Vector Register (UIVR)

- Program the vector number for a UART Module interrupt.

Interrupt Mask Register (UIMR)

- Enable the preferred interrupt sources.

Auxiliary Control Register (UACR)

- Select baud rate set (BRG bit).
- Initialize the Input Enable Control (IEC bit).
- Select timer mode and clock source if necessary.

Clock Select Register (UCSR)

- Select the receiver and transmitter clock. Use timer as source if required.

Mode Register 1 (UMR1)

- If preferred, program operation of receiver ready-to-send (RxRTS Bit).
- Select receiver-ready or FIFO-Full notification (R/F 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).

Mode Register 2 (UMR2)

- Select the mode of operation (CMx bits).
- If preferred, program operation of transmitter ready-to-send (TxRTS Bit).
- If preferred, program operation of clear-to-send (TxCTS Bit).
- Select stop-bit length (SBx Bits).

Command Register (UCR)

- Enable the receiver and transmitter.

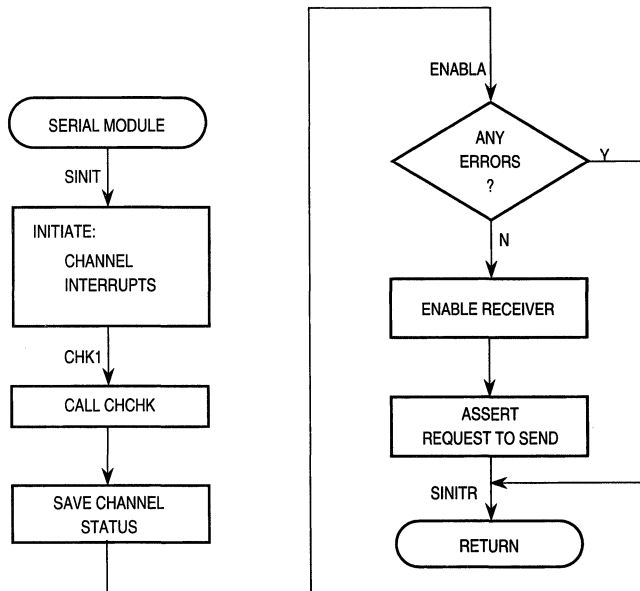


Figure 14-8. UART Mode Programming Flowchart (Sheet 1 of 5)

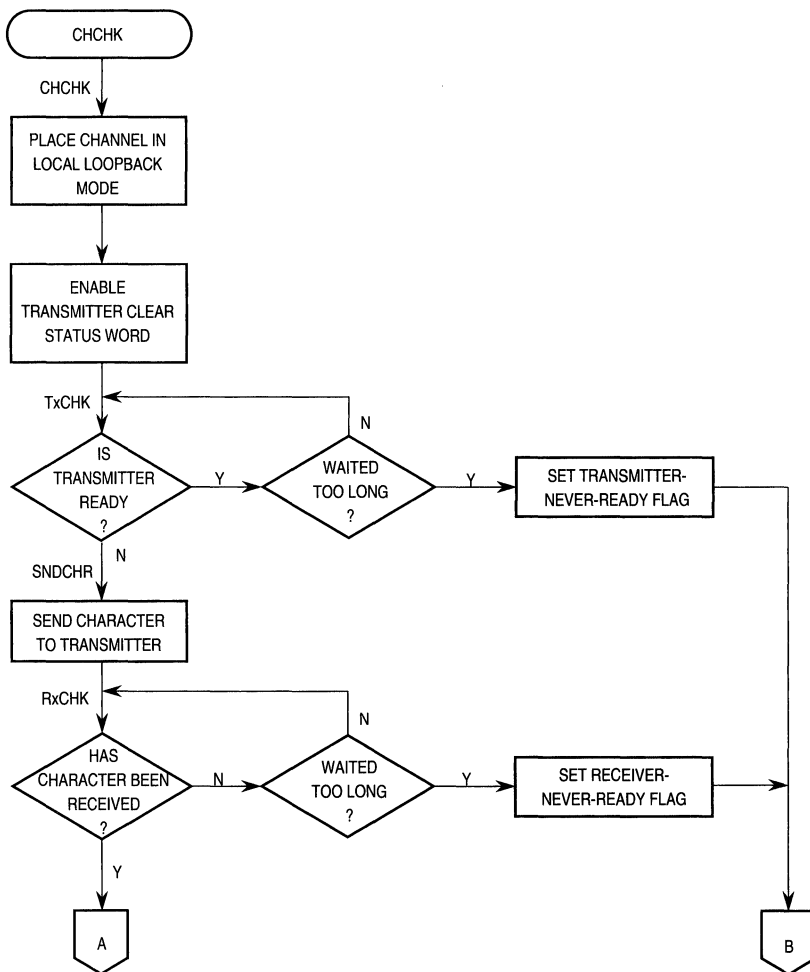


Figure 14-8. UART Mode Programming Flowchart (Sheet 2 of 5)

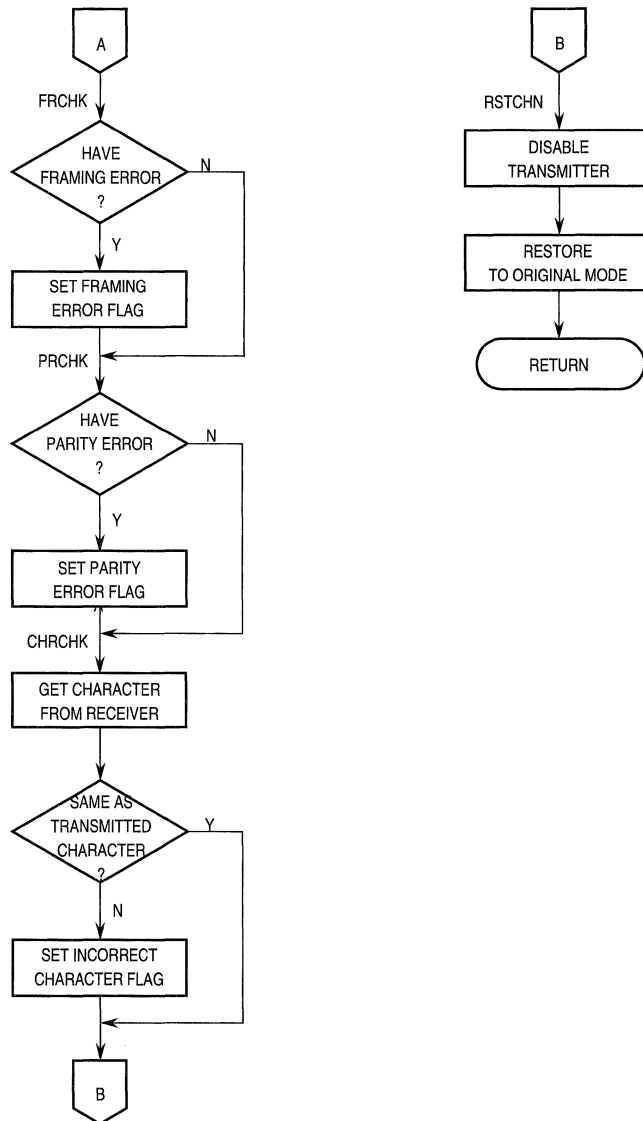


Figure 14-8. UART Mode Programming Flowchart (Sheet 3 of 5)

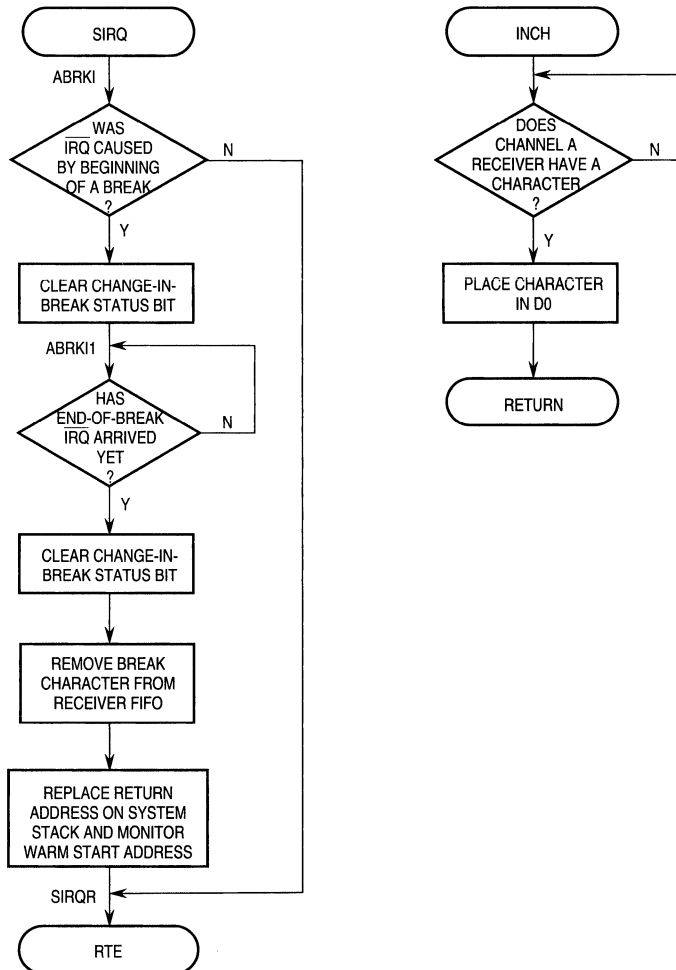


Figure 14-8. UART Mode Programming Flowchart (Sheet 4 of 5)

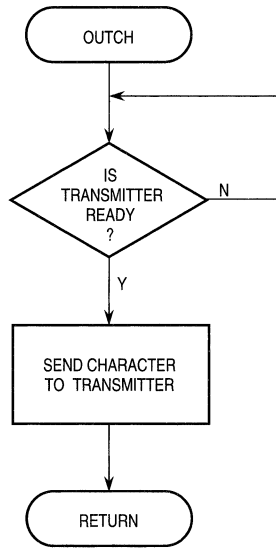


Figure 14-8. UART Mode Programming Flowchart (Sheet 5 of 5)

SECTION 15

M-BUS MODULE

15.1 OVERVIEW

Motorola bus (M-Bus) is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange between devices. It is compatible with the widely used I²C bus standard¹. This two-wire bus minimizes the interconnection between devices.

This bus is suitable for applications requiring occasional communications over a short distance between many devices. The flexible M-Bus allows additional devices to be connected to the bus for expansion and system development.

The interface operates up to 100 kbps with maximum bus loading and timing.

The M-Bus system is a true multimaster bus including collision detection and arbitration that prevents data corruption if two or more masters attempt to control the bus simultaneously. This feature allows for complex applications with multiprocessor control. It can also be used for rapid testing and alignment of end products via external connections to an assembly line computer.

15.2 INTERFACE FEATURES

The M-Bus module has the following key features:

- Compatibility with I²C Bus standard
- Multimaster 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

¹. I²C-Bus is a proprietary Philips interface bus.

Figure 15-1 shows the complete M-Bus module.

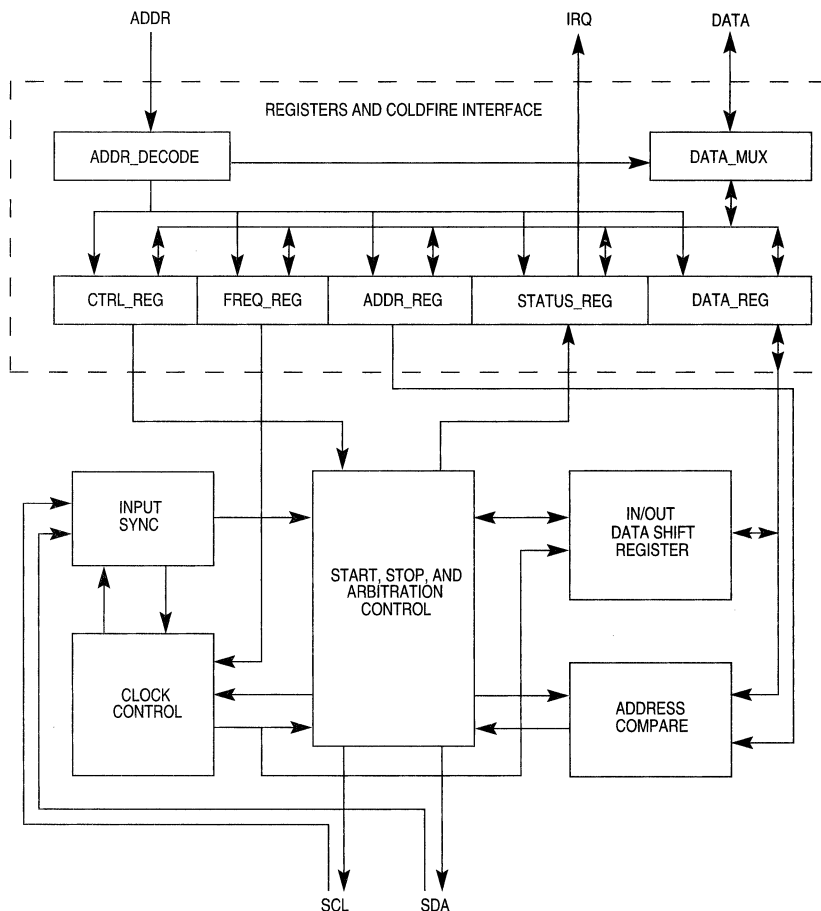


Figure 15-1. M-Bus Module Block Diagram

15.3 M-BUS SYSTEM CONFIGURATION

The M-Bus system uses a serial data line (SDA) and a serial clock line (SCL) for data transfer. All devices connected to these two signals must have open drain or open collector outputs. The logic AND function is exercised on both lines with external pullup resistors.

The default state of the M-Bus is as a slave receiver out of reset. Thus, when not programmed to be a master or responding to a slave transmit address, the M-Bus should always return to the default state of slave receiver.

NOTE

This M-Bus module is designed to be compatible with the I²C bus protocol from Philips. For further information on M-Bus system configuration, protocol, and restrictions please refer to the Philips I²C Standard

15.4 M-BUS PROTOCOL

Normally, a standard communication is composed of four parts: (1) START signal, (2) slave address transmission, (3) data transfer, and (4) STOP signal. They are described briefly in the following sections and illustrated in Figure 15-2.

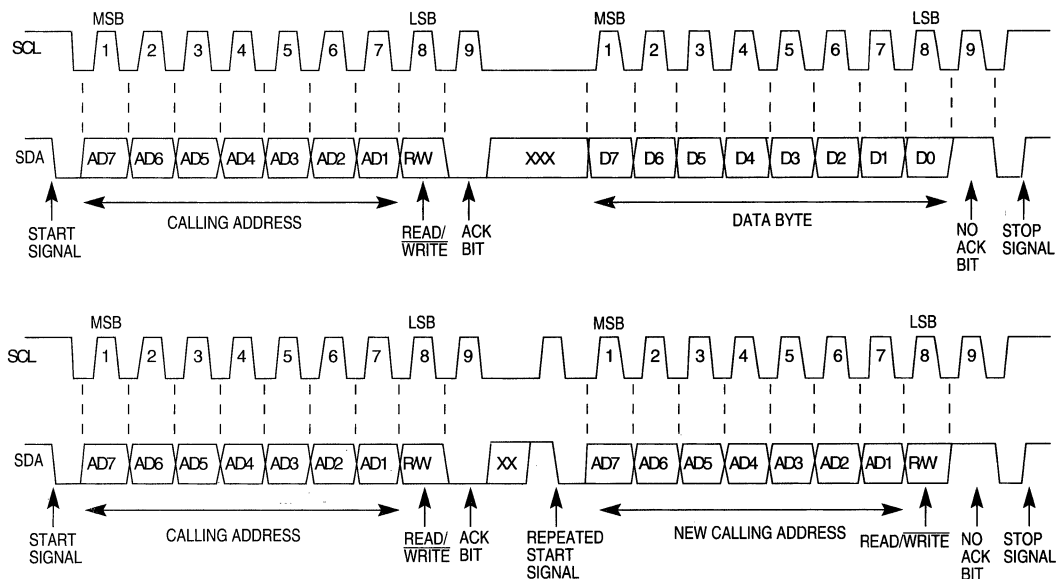


Figure 15-2. M-Bus Standard Communication Protocol

15.4.1 1START Signal

When the bus is free, i.e., no master device is engaging the bus (both SCL and SDA lines are at logic high), a master can initiate communication by sending a START signal. As shown in Figure 15-2, a START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer can contain several bytes of data) and awakens all slaves.

15.4.2 Slave Address Transmission

The first byte of data transferred by the master immediately after the START signal is the slave address. This is a seven-bit calling address followed by a R/W bit. The R/W bit tells

the slave data transfer direction. No two slaves in the system can have the same address. In addition, if the M-Bus is master, it must not transmit an address that is equal to its slave address. The M-Bus cannot be master and slave at the same time.

Only the slave with an address that matches the one transmitted by the master will respond by returning an acknowledge bit by pulling the SDA low at the 9th clock (see Figure 15-2).

15.4.3 Data Transfer

Once successful slave addressing is achieved, the data transfer can proceed on a byte-by-byte basis in the direction specified by the R/W bit sent by the calling master.

Each data byte is 8 bits long. Data can be changed only while SCL is low and must be held stable while SCL is high, as shown in Figure 15-2. There is one clock pulse on SCL for each data bit with the MSB being transferred first. Each byte of data must be followed by an acknowledge bit, which is signalled from the receiving device by pulling the SDA low at the ninth clock. One complete data byte transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop signal to abort the data transfer or a start signal (repeated start) to commence a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it can send "more data" to the slave. The slave releases the SDA line for the master to generate a STOP or START signal.

15.4.4 Repeated START Signal

As shown in Figure 15-2, a repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication. The master uses this method to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

15.4.5 STOP Signal

The master can terminate the communication by generating a STOP signal to free the bus. However, the master can generate a START signal followed by a calling command without generating a STOP signal first. This is called repeated START. A STOP signal is defined as a low-to-high transition of SDA while SCL is at logical 1 (see Figure 15-2). 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.

15.4.6 Arbitration Procedure

M-Bus is a true multimaster bus that allows more than one master to be connected on it. If two or more masters try to simultaneously control the bus, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the devices. A data arbitration procedure determines the relative priority of the contending masters. A bus master loses arbitration if it transmits logic 1 while another master transmits logic 0. The losing masters

immediately switch over to slave-receive mode and stop driving SDA output. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, the M-Bus or I²C device sets a status bit to indicate loss of arbitration.

15.4.7 Clock Synchronization

Because wire-AND logic is performed on SCL line, a high-to-low transition on SCL line affects all the devices connected on the bus. The devices start counting their low period when the master drives the SCL line low. Once a device clock has gone low, it holds the SCL line low until the clock high state is reached. However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see Figure 15-3). When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. There is then no difference between the device clocks and the state of the SCL line and all the devices start counting their high periods. The first device to complete its high period pulls the SCL line low again.

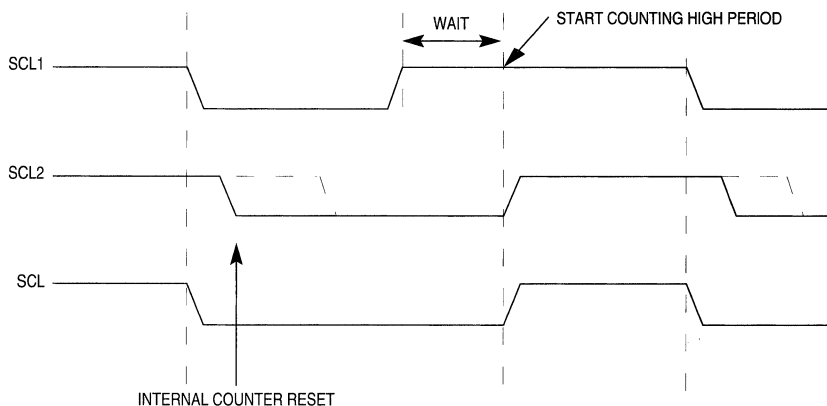


Figure 15-3. Synchronized Clock SCL

15.4.8 Handshaking

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

15.4.9 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 greater than the master SCL low period, the resulting SCL bus signal low period is stretched.

15.5 PROGRAMMING MODEL

Five registers are used in the M-Bus interface and the internal configuration of these registers is discussed in the following paragraphs. The programmer's model of the M-Bus interface is shown below in Table 15-1.

Table 15-1. M-Bus Interface Programmer's Model

ADDRESS	M-BUS MODULE REGISTERS
MBAR+\$280	M-Bus Address Register (MADR)
MBAR+\$284	M-Bus Frequency Divider Register (MFDR)
MBAR+\$288	M-Bus Control Register (MBCR)
MBAR+\$28C	M-Bus Status Register (MBSR)
MBAR+\$290	M-Bus Data I/O Register (MBDR)

15.5.1 M-Bus Address Register (MADR)

This register contains the address the M-Bus will respond to when addressed as a slave; note that it is not the address sent on the bus during the address transfer.

M-Bus Address Register (MADR)				Address MBAR+\$280			
7	6	5	4	3	2	1	0
ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	-
RESET	0	0	0	0	0	0	0
	Read/Write			Supervisor or User Mode			

ADR7–ADR1 — Slave Address

Bit 1 to bit 7 contain the specific slave address to be used by the M-Bus module.

NOTE

The default mode of M-Bus is slave mode for an address match on the bus.

15.5.2 M-Bus Frequency Divider Register (MFDR)

M-Bus Frequency Divider Register (MFDR)				Address MBAR+\$284			
7	6	5	4	3	2	1	0
-	-	MBC5	MBC4	MBC3	MBC2	MBC1	MBC0
RESET	0	0	0	0	0	0	0
	Read/Write			Supervisor or User Mode			

MBC5–MBC0 — M-Bus Clock Rate 5–0

This field is used to prescale the clock for bit rate selection. Due to the potential slow rise and fall times of the SCL and SDA signals, the bus signals are sampled at the prescaler frequency. The serial bit clock frequency is equal to the CPU clock divided by the divider

shown in Table 15-2². Note that the MFDR frequency value can be changed at any point in a program.

Table 15-2. MBUS Prescalar Values

MBC5-0 (HEX)	DIVIDER (DEC)	MBC5-0 (HEX)	DIVIDER (DEC)
00	28	20	20
01	30	21	22
02	34	22	24
03	40	23	26
04	44	24	28
05	48	25	32
06	56	26	36
07	68	27	40
08	80	28	48
09	88	29	56
0A	104	2A	64
0B	128	2B	72
0C	144	2C	80
0D	160	2D	96
0E	192	2E	112
0F	240	2F	128
10	288	30	160
11	320	31	192
12	384	32	224
13	480	33	256
14	576	34	320
15	640	35	384
16	768	36	448
17	960	37	512
18	1152	38	640
19	1280	39	768
1A	1536	3A	896
1B	1920	3B	1024
1C	2304	3C	1280
1D	2560	3D	1536
1E	3072	3E	1792
1F	3840	3F	2048

² In previous implementations of the M-Bus (e.g., MC68307), the MBC[5] bit was not implemented. Clearing this bit in software maintains complete compatibility with such products.

15.5.3 M-Bus Control Register (MBCR)

M-Bus Control Register (MBCR)					Address MBAR+\$288			
	7	6	5	4	3	2	1	0
	MEN	MIEN	MSTA	MTX	TXAK	RSTA	-	
RESET	0	0	0	0	0	0	0	0
	Read/Write				Supervisor or User Mode			

MEN — M-Bus Enable

This bit controls the software reset of the entire M-Bus module.

- 1 = The M-Bus module is enabled. This bit must be set before any other MBCR bits have any effect.
- 0 = The module is disabled, but registers can still be accessed.

If the M-Bus module is enabled in the middle of a byte transfer, the interface behaves as follows: the slave mode ignores the current transfer on the bus and starts operating whenever a subsequent start condition is detected. Master mode will not be aware that the bus is busy; therefore, if a start cycle is initiated, the current bus cycle can become corrupt. This would ultimately result in either the current bus master or the M-Bus module losing arbitration, after which bus operation would return to normal.

MIEN — M-Bus Interrupt Enable

- 1 = Interrupts from the M-Bus module are enabled. An M-Bus interrupt occurs provided the MIF bit in the status register is also set.
- 0 = Interrupts from the M-Bus module are disabled. This does not clear any currently pending interrupt condition.

MSTA — Master/Slave Mode Select Bit

At reset, this bit is cleared. When this bit is changed from 0 to 1, a START signal is generated on the bus, and the master mode is selected. When this bit is changed from 1 to 0, a STOP signal is generated and the operation mode changes from master to slave.

MSTA is cleared without generating a STOP signal when the master loses arbitration.

- 1 = Master Mode
- 0 = Slave Mode

MTX — Transmit/Receive mode select bit

This bit selects the direction of master and slave transfers. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. In master mode, this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high.

- 1 = Transmit
- 0 = Receive

TXAK — Transmit Acknowledge Enable

This bit specifies the value driven onto SDA during acknowledge cycles for both master and slave receivers. **Note that writing this bit only applies when the M-Bus is a receiver, not a transmitter.**

1 = No acknowledge signal response is sent (i.e., acknowledge bit = 1)

0 = An acknowledge signal will be sent out to the bus at the 9th clock bit after receiving one byte data

RSTA — Repeat Start

Writing a 1 to this bit will generate a repeated START condition on the bus, provided it is the current bus master. This bit will always be read as a low. Attempting a repeated start at the wrong time, if the bus is owned by another master, will result in loss of arbitration.

1 = Generate repeat start cycle

15.5.4 M-Bus Status Register (MBSR)

This status register is read-only with the exception of bit 1 (MIF) and bit 4 (MAL), which can be cleared by software. All bits are cleared on reset except bit 7 (MCF) and bit 0 (RXAK), which are set (=1) at reset.

M-Bus Status Register (MBSR)					Address MBAR+\$28C			
	7	6	5	4	3	2	1	0
	MCF	MAAS	MBB	MAL	-	SRW	MIF	RXAK
RESET	1	0	0	0	0	0	0	1
	Read/Write				Supervisor or User Mode			

MCF — Data Transferring Bit

While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer.

1 = Transfer complete

0 = Transfer in progress

MAAS — Addressed as a Slave Bit

When its own specific address (M-Bus Address Register) is matched with the calling address, this bit is set. The CPU is interrupted provided the MIEN is set. Next, the CPU must check the SRW bit and set its TX/RX mode accordingly.

Writing to the M-Bus Control Register clears this bit.

1 = Addressed as a slave

0 = Not addressed

MBB — Bus Busy Bit

This bit indicates the status of the bus. When a START signal is detected, the MBB is set. If a STOP signal is detected, it is cleared.

- 1 = Bus is busy
- 0 = Bus is idle

MAL — Arbitration Lost

Hardware sets the arbitration lost bit (MAL) when the arbitration procedure is lost. Arbitration is lost in the following circumstances:

1. SDA sampled as low when the master drives a high during an address or data-transmit cycle.
2. SDA sampled as a low when the master drives a high during the acknowledge bit of a data-receive cycle.
3. A start cycle is attempted when the bus is busy.
4. A repeated start cycle is requested in slave mode.
5. A stop condition is detected when the master did not request it.

This bit must be cleared by software by writing a low to it.

SRW — Slave Read/Write

When MAAS is set, this bit indicates the value of the R/W command bit of the calling address sent from the master. This bit is valid only when 1) a complete transfer has occurred and no other transfers have been initiated and 2) M-Bus is a slave and has an address match. Checking this bit, the CPU can select slave transmit/receive mode according to the command of the master.

- 1 = Slave transmit, master reading from slave
- 0 = Slave receive, master writing to slave

MIF — M-Bus Interrupt

The MIF bit is set when an interrupt is pending, which will cause a processor interrupt request (provided MIEN is set). MIF is set when one of the following events occurs:

1. Complete one byte transfer (set at the falling edge of the 9th clock)
2. Receive a calling address that matches its own specific address in slave-receive mode
3. Arbitration lost

This bit must be cleared by software by writing a low to it in the interrupt routine.

RXAK — Received Acknowledge

The value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion

of 8 bits data transmission on the bus. If RXAK is high, it means no acknowledge signal has been detected at the 9th clock.

- 1 = No acknowledge received
- 0 = Acknowledge received

15.5.5 M-Bus Data I/O Register (MBDR)

M-Bus Data I/O Register (MBDR)				Address MBAR+\$290				
	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
RESET	0	0	0	0	0	0	0	0
	Read/Write				Supervisor or User Mode			

When an address and R/W bit is written to the MBDR and the M-Bus is the master, a transmission will start. When data is written to the MBDR, a data transfer is initiated. The most significant bit is sent first in both cases. In the master-receive mode, reading the MBDR register allows the read to occur but also initiates next byte data receiving. In slave mode, the same function is available after it is addressed.

15.6 M-BUS PROGRAMMING EXAMPLES

15.6.1 Initialization Sequence

Reset will put the M-Bus Control Register to its default status. Before the interface can transfer serial data, you must perform an initialization procedure as follows:

1. Update the Frequency Divider Register (MFDR) and select the required division ratio to obtain SCL frequency from the system bus clock.
2. Update the M-Bus Address Register (MADR) to define its slave address.
3. Set the MEN bit of the M-Bus Control Register (MBCR) to enable the M-Bus interface system.
4. Modify the bits of the M-Bus Control Register (MBCR) to select master/slave mode, transmit/receive mode, and interrupt-enable or not.

15.6.2 Generation of START

After completion of the initialization procedure, you can transmit serial data by selecting the “master transmitter” mode. If the device is connected to a multimaster bus system, you must test the state of the M-Bus Busy Bit (MBB) to check whether the serial bus is free.

If the bus is free (MBB=0), the start condition 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 is set to indicate the direction of transfer required.

The bus free time (i.e., the time between a STOP condition and the following START condition) is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period, you may have to wait until the

M-Bus is busy after writing the calling address to the MBDR before proceeding with the following instructions.

An example of a program that generates the START signal and transmits the first byte of data (slave address) is shown below:

```
CHFLAGMOVE.BMSR,-(A7);   Check the MBB bit of the BTST.
B#5, (A7)+
BNE.SCHFLAG;             Status Register. If it is set, wait until it is clear
TXSTARTMOVE.BMBCR,-(A7); Set transmit mode
BSET.B#4, (A7)
MOVE.B(A7)+, MBCR
MOVE.BMBCR, -(A7);       Set master mode
BSET.B#5, (A7);          Generate START condition
MOVE.B(A7)+, MBCR;
MOVE.BCALLING,-(A7);     Transmit the calling address, D0=R/W
MOVE.B(A7)+, MBDR;
MBFREEMOVE.BMSR,-(A7);   Check the MBB bit of the Status Register. If it is clear,
                           wait until it is set.

BTST.B#5, (A7)+;
BEQ.SMBFREE;
```

15.6.3 Post-Transfer Software Response

Transmission or reception of a byte will set the data transferring bit (MCF) to 1, which indicates one byte communication is finished. The M-Bus interrupt bit (MIF) is also set. An interrupt will be generated if the interrupt function is enabled during initialization by setting the MIEN bit. Software must clear the MIF bit in the interrupt routine first. The MCF bit will be cleared by reading from the M-Bus Data I/O Register (MDR) in receive mode or writing to MDR in transmit mode.

Software can service the M-Bus I/O in the main program by monitoring the MIF bit if the interrupt function is disabled. Polling should monitor the MIF bit rather than the MCF bit because that operation is different when arbitration is lost.

When an interrupt occurs at the end of the address cycle, the master will always be in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/\overline{W} bit in MBDR, then the MTX bit should be toggled.

During slave-mode address cycles (MAAS=1), the SRW bit in the status register is read to determine the direction of the subsequent transfer and the MTX bit is programmed accordingly. For slave-mode data cycles (MAAS=0), the SRW bit is not valid. The MTX bit in the control register should be read to determine the direction of the current transfer.

The following is an example of a software response by a "master transmitter" in the interrupt routine (see Figure 15-4).

```

ISRLEA.LMBSR, -(A7);      Load effective address
BCLR.B#1, (A7)+;          Clear the MIF flag
MOVE.BMBCR, -(A7);        Push the address on stack,
BTST.B#5, (A7)+;          check the MSTA flag
BEQ.SSLAVE;                Branch if slave mode
MOVE.BMBCR, -(A7);        Push the address on stack,
BTST.B#4, (A7)+;          check the mode flag
BEQ.SRECEIVE;             Branch if in receive mode
MOVE.BMBSR, -(A7);        Push the address on stack,
BTST.B#0, (A7)+;          check ACK from receiver
BNE.B END;                 If no ACK, end of transmission
TRANSMITMOVE.BDATABUF, -(A7); Stack data byte
MOVE.B(A7)+, MBDR;         Transmit next byte of data

```

15.6.4 Generation of STOP

A data transfer ends with a STOP signal generated by the “master” device. A master transmitter can generate a STOP signal after all the data has been transmitted. The following code is an example showing how a master transmitter generates a stop condition.

```

MASTXMOVE.BMBSR, -(A7); If no ACK, branch to end
BTST.B#0, (A7)+
BNE.B END
MOVE.BTXCNT,D0;           Get value from the transmitting counter
BEQ.SEND;                 If no more data, branch to end
MOVE.BDATABUF, -(A7);    Transmit next byte of data
MOVE.B(A7)+,MBDR
MOVE.BTXCNT,D0;           Decrease the TXCNT
SUBQ.L#1,D0
MOVE.BD0, TXCNT
BRA.SEMASTX;              Exit
ENDLEA.LMBCR, -(A7);     Generate a STOP condition
BCLR.B#5, (A7)+
EMASTXRTE;                Return from interrupt

```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data, which can be done by setting the transmit acknowledge bit (TXAK) before reading the 2nd to last byte of data. Before reading the last byte of data, a STOP signal must first be generated. The following code is an example showing how a master receiver generates a STOP signal.

```

MASRMOVE.BRXCNT,D0;      Decrease RXCNT
SUBQ.L#1,D0
MOVE.BD0,RXCNT
BEQ.SENMASR;             Last byte to be read
MOVE.BRXCNT,D1;          Check second-to-last byte to be read
EXTB>LD1
SUBI.L#1,D1;
BNE.SNXMAR;              Not last one or second last
LAMARBSET.B#3,MBCR;      Disable ACK
BRANXMAR                  Transmitting
ENMASRBCLR.B#5,MBCR;     Last one, generate 'STOP'signal
NXMARMOVE.BMBDR,RXBUF;   Read data and store RTE

```


15.6.5 Generation of Repeated START

At the end of data transfer, if the master still wants to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example follows.

```
RESTARTMOVE.BMBCR,-(A7); Another START (RESTART)
BSET.B#2,(A7)
MOVE.B(A7)+,MBCR
MOVE.BCALLING,-(A7); Transmit the calling address, D0=R/W-
MOVE.BCALLING,-(A7);
MOVE.B(A7)+,MBDR
```

15.6.6 Slave Mode

In the slave interrupt service routine, the module addressed as slave bit (MAAS) should be tested to check if a calling of its own address has just been received. If MAAS is set, software should set the transmit/receive mode select bit (MTX bit of MBCR) according to the R/\bar{W} command bit (SRW). Writing to the MBCR clears the MAAS automatically. The only time MAAS 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 MAAS cleared. A data transfer can now be initiated by writing information to MBDR, for slave transmits, or read from MBDR, in slave-receive mode. A dummy read of the MBDR in slave/receive mode will release SCL, allowing the master to transmit data.

In the slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means an "end-of-data" signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A read from MBDR then releases the SCL line so that the master can generate a STOP signal.

15.6.7 Arbitration Lost

If several masters try to simultaneously engage the bus, only one master wins and the others lose arbitration. The devices that lost arbitration are immediately switched to slave receive mode by the hardware. Their data output to the SDA line is stopped, but SCL is still generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with MAL=1 and MSTA=0. If one master tries to transmit or do a START while the bus is being engaged by another master, the hardware will: (1) inhibit the transmission, (2) switch the MSTA bit from 1 to 0 without generating STOP condition, (3) generate an interrupt to CPU and, (4) set the MAL to indicate the failed attempt to engage the bus. When considering these cases, the slave service routine should test the MAL first and the software should clear the MAL bit if it is set.

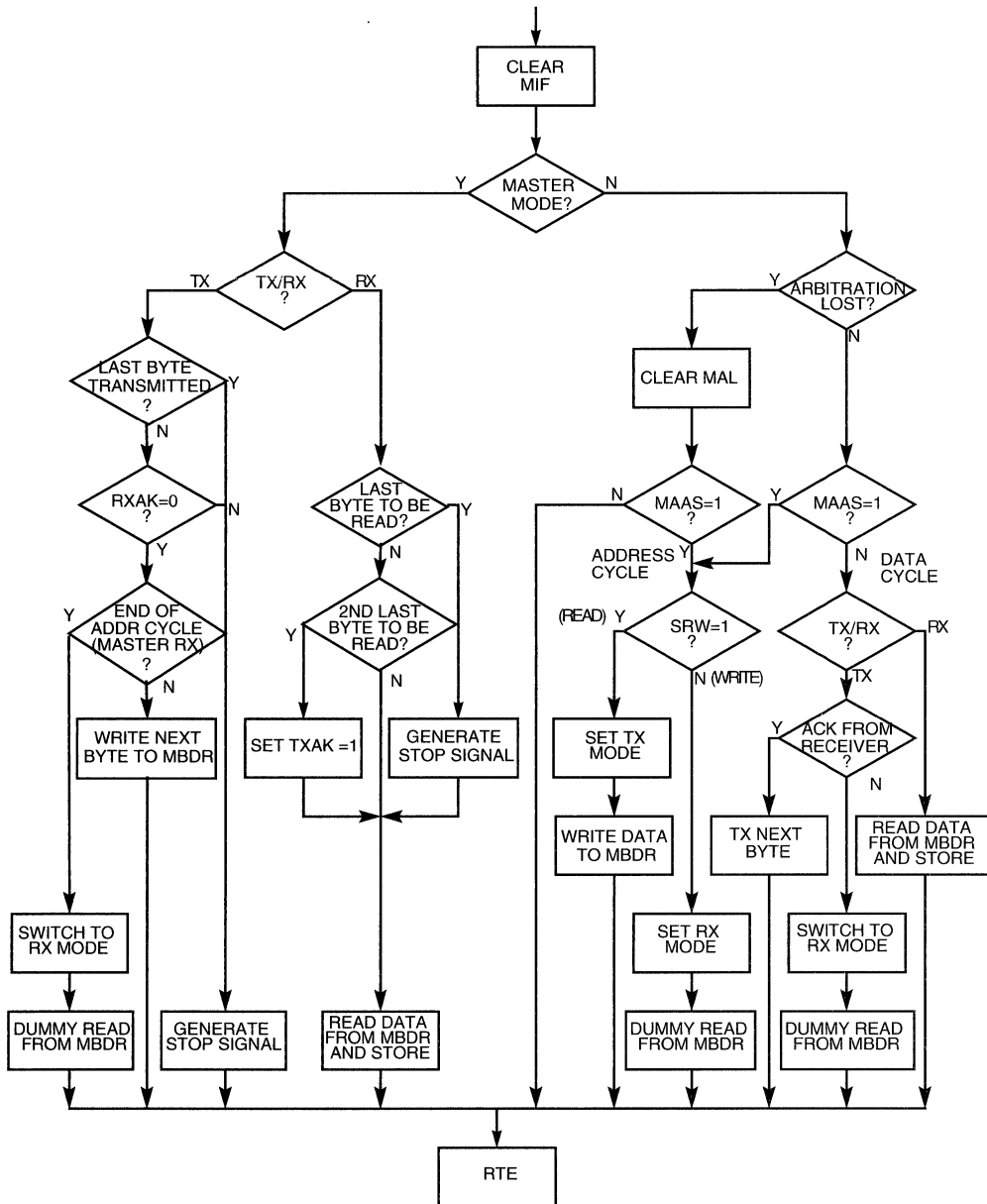


Figure 15-4. Flow-Chart of Typical M-Bus Interrupt Routine

SECTION 16 DEBUG SUPPORT

This section details the hardware debug support functions within the ColdFire family of processors. The Version 3 ColdFire implements an enhanced debug architecture compared to the original specification. The original design plus these enhancements is known as Revision B (or Rev. B), while the initial definition is Revision A (or Rev. A). The enhanced functionality is clearly identified in this section. The Rev. B enhancements are backward compatible with the original ColdFire debug definition.

The general topic of debug support is divided into three separate areas:

- Real-Time Trace Support
- Background Debug Mode (BDM)
- Real-Time Debug Support

Each of the three areas is addressed in detail in the following subsections.

The logic required to support these three areas is contained in a Debug Module, which is shown in the system block diagram in Figure 16-1.

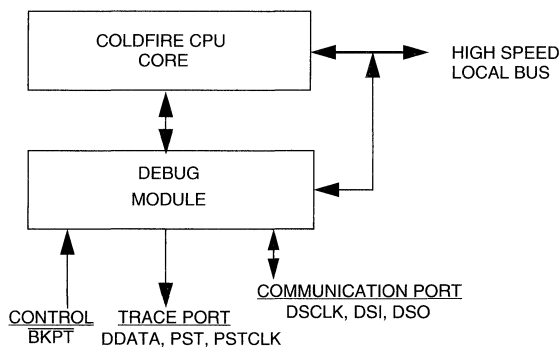


Figure 16-1. Processor/Debug Module Interface

16.1 SIGNAL DESCRIPTION

This section describes the ColdFire 53xx signals associated with the Debug Module. All ColdFire debug signals are unidirectional and related to the rising-edge of the processor core's clock signal.

16.1.1 Breakpoint ($\overline{\text{BKPT}}$)

16.1.1.1 REV A FUNCTIONALITY. This active-low, input signal is used to request a manual breakpoint. Its assertion causes the processor to enter a halted state after the completion of the current instruction. The halt status is reflected on the processor status (PST) pins as the value \$F.

16.1.1.2 REV. B ENHANCEMENT. In addition to the baseline functionality, if the BKD bit of the Configuration/Status Register is set ($\text{CSR}[18]$), the assertion of the $\overline{\text{BKPT}}$ signal generates a debug interrupt exception in the processor.

16.1.2 Debug Data ($\text{DDATA}[3:0]$)

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.

16.1.3 Development Serial Clock (DSCLK)

This input signal is synchronized internally and provides the clock for the serial communication port to the Debug Module. The maximum frequency is 1/5 the speed of the processor's clock (CLK). At the synchronized rising edge of DSCLK, the data input on DSI is sampled, and the DSO output changes state.

16.1.4 Development Serial Input (DSI)

The input signal is synchronized internally and provides the data input for the serial communication port to the Debug Module.

16.1.5 Development Serial Output (DSO)

This signal provides serial output communication for the Debug Module responses.

16.1.6 Processor Status ($\text{PST}[3:0]$)

These output signals report the processor status. Table 16-1 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.

16.1.7 Processor Status Clock (PSTCLK)

Since the debug trace port signals transition each processor cycle and are not related to the external bus frequency, an additional signal is output from the ColdFire microprocessor. The PSTCLK signal is a delayed version of the processor's high-speed clock and its rising-edge

is used by the development system to sample the values on the PST and DDATA output buses. The PSTCLK signal is intended for use in the standard 26-pin debug connector.

If the real-time trace functionality is not being used, the PCD bit of the CSR may be set (CSR[17] = 1) to force the PSTCLK, PST and DDATA outputs to be quiescent and not toggle at the processor's clock speed.

Table 16-1. Processor Status Encoding

PST[3:0]		DEFINITION
(HEX)	(BINARY)	
\$0	0000	Continue execution
\$1	0001	Begin execution of an instruction
\$2	0010	Reserved
\$3	0011	Entry into user-mode
\$4	0100	Begin execution of PULSE and WDDATA instructions
\$5	0101	Begin execution of taken branch or Sync_PC ¹
\$6	0110	Reserved
\$7	0111	Begin execution of RTE instruction
\$8	1000	Begin 1-byte transfer on DDATA
\$9	1001	Begin 2-byte transfer on DDATA
\$A	1010	Begin 3-byte transfer on DDATA
\$B	1011	Begin 4-byte transfer on DDATA
\$C	1100	Exception processing†
\$D	1101	Emulator-mode entry exception processing†
\$E	1110	Processor is stopped, waiting for interrupt†
\$F	1111	Processor is halted †

NOTE: †These encodings are asserted for multiple cycles.

1 Rev. B enhancement.

16.2 REAL-TIME TRACE SUPPORT

In the area of debug functions, one fundamental requirement is support for real-time trace functionality, i.e., definition of the dynamic execution path. The ColdFire 53xx 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 nibbles (4 bits): one nibble allows the processor to transmit information concerning the execution status of the core (processor status, PST), while the other nibble allows operand data to be displayed (debug data, DDATA). The processor status (PST) timing is synchronous with the processor clock (CLK) and may not be related to the current bus transfer. Table 16-1 shows the encoding of these signals.

The processor status (PST) outputs can be used with an external image of the program to completely track the dynamic execution path of the machine when used with external development systems. The tracking of this dynamic path is complicated by any change-of-flow operation. This is especially evident when the branch target address is calculated based on the contents of a program-visible register (variant addressing.) For this reason, the debug data (DDATA) outputs can be configured to display the target address of these types of change-of-flow instructions. Because the DDATA bus is only 4 bits wide, the address is displayed a nibble at a time across multiple clock cycles.

The Debug Module includes two 32-bit storage elements for capturing the internal ColdFire 53xx bus information. These two elements effectively form a FIFO buffer connecting the processor's high-speed local bus to the external development system through the DDATA signals. The FIFO buffer captures branch target addresses along with certain operand data values for eventual display on the DDATA output port on nibble at a time starting with the least-significant bit. The execution speed of the ColdFire processor is affected only when both storage elements contain valid data waiting to be dumped onto the DDATA port. In this case, the processor core is stalled until one FIFO entry is available. In all other cases, data output on the DDATA port does not impact execution speed.

16.2.1 Processor Status Signal Encoding

The processor status (PST) signals are encoded to reflect the state of the Operand Execution Pipeline, and are generally not related to the current external bus transfer.

16.2.1.1 CONTINUE EXECUTION (PST = \$0). Many instructions complete in a single processor cycle. If an instruction requires more clock cycles, the subsequent clock cycles are indicated by driving the PST outputs with this encoding.

16.2.1.2 BEGIN EXECUTION OF AN INSTRUCTION (PST = \$1). 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.

16.2.1.3 ENTRY INTO USER MODE (PST = \$3). This encoding indicates the ColdFire processor has entered user mode. This encoding is signaled after the instruction which caused the user mode entry has executed (signaled with the appropriate encoding.)

16.2.1.4 BEGIN EXECUTION OF PULSE OR WDDATA INSTRUCTIONS (PST = \$4). The ColdFire 53xx instruction set architecture includes a PULSE opcode. This opcode generates a unique PST encoding, \$4, when executed. This instruction can define logic analyzer triggers for debug and/or performance analysis. Additionally, a WDDATA instruction is supported that allows the processor core to write any operand (byte, word, longword) directly to the DDATA port, independent of any Debug Module configuration. This opcode also generates the special PST encoding (\$4) when executed, followed by the appropriate marker and then the data transfer on the DDATA outputs. The length of the data transfer is dependent on the operand size of the WDDATA instruction.

16

16.2.1.5 BEGIN EXECUTION OF TAKEN BRANCH (PST = \$5). This encoding is generated whenever a taken branch is executed. For certain opcodes, the branch target address may be optionally displayed on DDATA depending on the control parameters contained in the configuration/status register (CSR). The number of bytes of the address to be displayed is also controlled in the CSR and indicated by the PST marker value immediately preceding the DDATA outputs.

The bytes are always displayed in a least-significant to most-significant order. The processor captures only those target addresses associated with taken branches using a variant addressing mode, i.e., all JMP and JSR instructions using address register indirect or indexed addressing modes, all RTE and RTS instructions as well as 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 these types of change-of-flow operations, the ColdFire 53xx processor uses the debug pins to output a sequence of information on successive processor clock cycles

1. Identify a taken branch has been executed using the PST pins (\$5).
2. Using the PST pins, optionally signal the target address is to be displayed on the DDATA pins. The encoding (\$9, \$A, \$B) identifies the number of bytes that are displayed.
3. The new target address is optionally available on subsequent cycles using the nibble-wide DDATA port. The number of bytes of the target address displayed on this port is a configurable parameter (2, 3, or 4 bytes).

Another example of a variant branch instruction would be a JMP (A0) instruction. If the CSR was programmed to display the lower two bytes of an address, the outputs of the PST and DDATA signals when this instruction executed are shown in Figure 16-2.

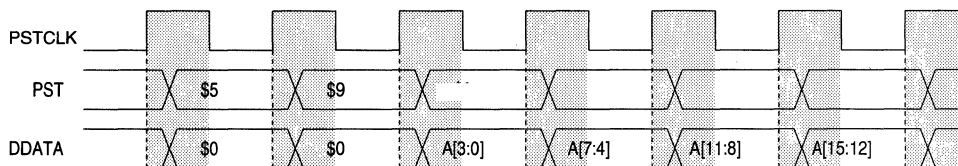


Figure 16-2. Example PST/DDATA Diagram

PST is driven with a \$5 indicating a taken branch. In the second cycle, PST is driven with a marker value of \$9 indicating a two-byte address that is displayed four bits at a time on the DDATA signals over the next four clock cycles. The remaining four clock cycles display the lower two-bytes of the address (A0), least significant nibble to most significant nibble. The output of the PST signals after the JMP instruction completes is dependent on the target instruction. The PST can continue with the next instruction before the address has completely displayed on the DDATA because of the DDATA FIFO. If the FIFO is full and the next instruction needs to display captured values on DDATA, the pipeline stalls (PST = \$0) until space is available in the FIFO.

16.2.1.6 BEGIN EXECUTION OF RTE INSTRUCTION (PST = \$7). The unique encoding is generated whenever the return-from-exception (RTE) instruction is executed.

16.2.1.7 BEGIN DATA TRANSFER (PST = \$8 - \$B). These encodings serve as markers to indicate the number of bytes to be displayed on the DDATA port on subsequent clock cycles. This encoding is driven onto the PST port one processor cycle before the actual data is displayed on DDATA. When PST outputs a \$8/\$9/\$A/\$B marker value, the DDATA port outputs 1/2/3/4 bytes of captured data respectively on consecutive processor cycles.

16.2.1.8 EXCEPTION PROCESSING (PST = \$C). This encoding is displayed during normal exception processing. Exceptions which enter emulation mode (debug interrupt, or optionally trace) generate a different encoding. Because this encoding defines a multicycle mode, the PST outputs are driven with this value until exception processing is completed.

16.2.1.9 EMULATOR MODE EXCEPTION PROCESSING (PST = \$D). This encoding is displayed during emulation mode (debug interrupt, or optionally trace). Because this encoding defines a multicycle mode, the PST outputs are driven with this value until exception processing is completed.

16.2.1.10 PROCESSOR STOPPED (PST = \$E). This encoding is generated as a result of the STOP instruction. The ColdFire processor remains in the stopped state until an interrupt occurs. Because this encoding defines a multicycle mode, the PST outputs are driven with this value until the stopped mode is exited.

16.2.1.11 PROCESSOR HALTED (PST = \$F). This encoding is generated when the ColdFire processor is halted (see **Section 16.3.1 CPU Halt.**) Because this encoding defines a multicycle mode, the PST outputs are driven with this value until the processor is restarted, or reset.

16.3 BACKGROUND-DEBUG MODE (BDM)

The ColdFire Family supports a modified version of the background debug mode (BDM) functionality found on Motorola's CPU32 family of parts. BDM implements a low-level system debugger in the microprocessor hardware. Communication with the development system is handled via a dedicated, high-speed serial command interface.

Unless noted otherwise, the BDM functionality provided by ColdFire is a proper subset of the CPU32 functionality. The main differences include the following:

- ColdFire 53xx implements the BDM controller in a dedicated hardware module. Although some BDM operations do require the CPU to be halted (e.g. CPU register accesses), other BDM commands such as memory accesses can be executed while the processor is running.
- On CPU32 parts, the DSO signal can inform hardware that a serial transfer can start. ColdFire clocking schemes restrict the use of this bit. Because DSO changes only when DSCLK is high, DSO cannot be used to indicate the start of a serial transfer. The development system count the number of clocks in any given transfer.
- The read/write system register commands, RSREG and WSREG, have been replaced by read/write control register commands, RCREG and WCREG. These commands use the register coding scheme from the MOVEC instruction.
- The read/write Debug Module register commands, RDMREG and WDMREG, have been added to support Debug Module register accesses.
- CALL and RST commands are not supported and generates an illegal command response.
- Illegal command responses can be returned using the FILL and DUMP commands, if not immediately preceded by certain, specific BDM commands.

- For any command performing a byte-sized memory read operation, the upper 8 bits of the response data are undefined. The referenced data is returned in the lower 8 bits of the response.
- The Debug Module forces alignment for memory-referencing operations: long accesses are forced to a 0-modulo-4 address; word accesses are forced to a 0-modulo-2 address. An address error response is never returned.

16.3.1 CPU Halt

Although many BDM operations can occur in parallel with CPU operation, unrestricted BDM operation requires the CPU to be halted. A number of sources can cause the CPU to halt, including the following as shown in order of priority:

1. The occurrence of the catastrophic fault-on-fault condition automatically halts the processor.
2. The occurrence of a hardware breakpoint can be configured to generate a pending halt condition in a manner similar to the assertion of the `BKPT` signal. In all cases, the assertion of this type of halt is first made pending in the processor. Next, the processor samples for pending halt and interrupt conditions once per instruction. Once the pending condition is asserted, the processor halts execution at the next sample point. See **Section 16.4.1 Theory of Operation** for more detail.
3. The execution of the `HALT` instruction, also known as `BGND` on the 683xx devices, immediately suspends execution. By default this is a supervisor instruction and attempted execution while in user mode generates a privilege violation exception. A User Halt Enable (UHE) control bit is provided in the Configuration/Status Register (`CSR`) to allow execution of `HALT` in user mode. The processor may be restarted after the execution of the `HALT` instruction by serial shifting a “GO” command into the debug module. Execution continues at the instruction following the `HALT` opcode.
4. The assertion of the `BKPT` input pin is treated as a pseudo-interrupt, i.e., the halt condition is made pending 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.

There are two special cases involving the assertion of the `BKPT` pin to be considered.

After the system reset signal is negated, the processor waits for 16 clock cycles before beginning reset exception processing. If the `BKPT` input pin is asserted within the first eight cycles after `RSTI` is negated, the processor enters the halt state, signaling that halt status, (`$F`), on the `PST` outputs. While in this state, all resources accessible via the Debug Module can be referenced. This is the only opportunity to force the ColdFire processor into emulation mode via the `EMU` bit in the configuration/status register (`CSR`). Once the system initialization is complete, the processor response to a BDM `GO` command is dependent on the set of BDM commands performed while breakpointed. Specifically, if the processor's `PC` register was loaded, then the `GO` command simply causes the processor to exit the halted state and pass control to the instruction address contained in the `PC`. Note in this case, the normal reset exception processing is bypassed. Conversely, if the `PC` register was not

loaded, then the GO command causes the processor to exit the halted state and continue with reset exception processing.

ColdFire also handles a special case of the assertion of BKPT 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. Once halted, all BDM commands may be exercised. When the processor is restarted, it continues with the execution of the next sequential instruction, i.e., the instruction following the STOP opcode.

The halt source is indicated in CSR[27:24]. For simultaneous halt conditions, the highest priority source is indicated.

16.3.2 BDM Serial Interface

Once the CPU is halted and the halt status reflected on the PST outputs, the development system may send unrestricted commands to the Debug Module. The Debug Module implements a synchronous protocol using a three-pin interface: development serial clock (DSCLK), development serial input (DSI), and development serial output (DSO). The development system serves as the serial communication channel master and is responsible for generation of the clock (DSCLK). The operating range of the serial channel is DC to 1/5 of the processor frequency. The channel uses a full duplex mode, where data is transmitted 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 seen in Figure 16-3, all state transitions are enabled on a rising edge of the processor clock when DSCLK is high, i.e., DSI is sampled and DSI is driven. The DSCLK signal must also be sampled low (on a positive edge of CLK) between each bit exchange. The MSB is transferred first.

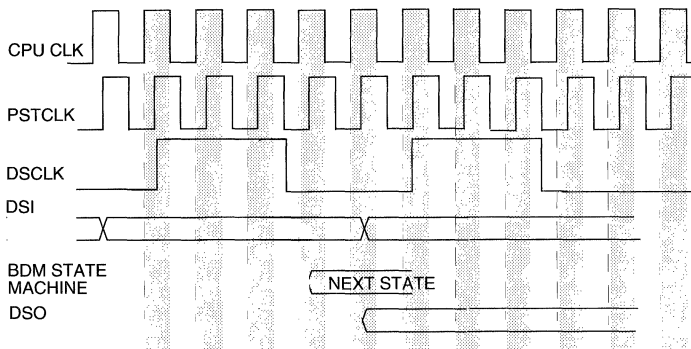


Figure 16-3. BDM Serial Transfer

Both DSCLK and DSI are synchronized inputs. The DSCLK signal essentially acts as a pseudo “clock enable” and is sampled on the rising edge of CLK as well as the DSI. The DSO output is delayed from the DSCLK-enabled CLK rising edge. All events in the Debug Module’s serial state machine are based on the rising edge of the microprocessor clock).

16.3.2.1 RECEIVE PACKET FORMAT. The basic receive packet of information is 17 bits long, 16 data bits plus a status bit, as shown below in Figure 16-4.

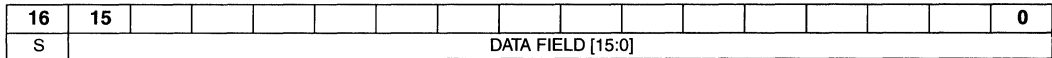


Figure 16-4. Receive BDM Packet

Status[16]

The status bit indicates the status of CPU-generated messages as listed in Table 16-2.

Table 16-2. CPU-Generated Message Encoding

S BIT	DATA	MESSAGE TYPE
0	xxxx	Valid Data Transfer
0	\$FFFF	Status Ok
1	\$0000	Not Ready with Response; Come Again
1	\$0001	Error - Terminated Bus Cycle; Data Invalid
1	\$FFFF	Illegal Command

Data Field[15:0]

The data field contains the message data to be communicated from the Debug Module to the development system. The response message is always a single word, with the data field encoded as shown in Table 16-2.

16.3.2.2 TRANSMIT PACKET FORMAT. The basic transmit packet of information is 17 bits long, 16 data bits plus a control bit, as shown below in Figure 16-5.

16	15	0
C	DATA FIELD [15:0]	

Figure 16-5. Transmit BDM Packet

Control[16]

The control bit is not used but is reserved by Motorola for future use. Command and data transfers initiated by the development system should clear bit 16.

Data Field[15:0]

The data field contains the message data to be communicated from the development system to the Debug Module.

16.3.3 BDM Command Set

ColdFire supports a subset of BDM instructions from the MC683xx parts, as well as extensions to provide access to new hardware features. The BDM commands must not be issued whenever the ColdFire processor is accessing the Debug Module registers using the WDEBUG instruction, or the resulting behaviour is undefined.

16.3.3.1 BDM COMMAND SET SUMMARY. The BDM command set is summarized in Table 16-3. Subsequent paragraphs contain detailed descriptions of each command.

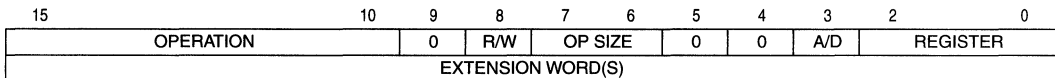
Table 16-3. BDM Command Summary

COMMAND	MNEMONIC	DESCRIPTION	CPU IMPACT ¹	PAGE
READ A/D REGISTER	RAREG/RDREG	Read the selected address or data register and return the results via the serial interface.	HALTED	16-14
WRITE A/D REGISTER	WAREG/WDREG	The data operand is written to the specified address or data register.	HALTED	16-15
READ MEMORY LOCATION	READ	Read the data at the memory location specified by the longword address.	STEAL	16-16
WRITE MEMORY LOCATION	WRITE	Write the operand data to the memory location specified by the longword address.	STEAL	16-17
DUMP MEMORY BLOCK	DUMP	Used in conjunction with the READ command 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. Subsequent operands are retrieved with the DUMP command.	STEAL	16-19
FILL MEMORY BLOCK	FILL	Used in conjunction with the WRITE command 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. Subsequent operands are written with the FILL command.	STEAL	16-21
RESUME EXECUTION	GO	The pipeline is flushed and refilled before resuming instruction execution at the current PC.	HALTED	16-23
NO OPERATION	NOP	NOP performs no operation and may be used as a null command.	PARALLEL	16-23
OUTPUTS THE CURRENT PC	SYNC_PC	Captures the current PC and displays it on the PST/DDATA output pins.	PARALLEL	16-24
READ CONTROL REGISTER	RCREG	Read the system control register.	HALTED	16-24
WRITE CONTROL REGISTER	WCREG	Write the operand data to the system control register.	HALTED	16-26
READ Debug Module REGISTER	RDMREG	Read the Debug Module register.	PARALLEL	16-26
WRITE DEBUG MODULE REGISTER	WDMREG	Write the operand data to the Debug Module register.	PARALLEL	16-27

NOTE: 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 which can be interleaved with CPU accesses
- Parallel - Command is executed in parallel with CPU activity
- Refer to command summaries for detailed operation descriptions.

16.3.3.2 COLD FIRE BDM COMMANDS. All ColdFire Family BDM commands include a 16-bit operation word followed by an optional set of one or more extension words.



BDM Command Format

Operation Field

The operation field specifies the command.

R/W Field

The R/W field specifies the direction of operand transfer. When the bit is set, the transfer is from the CPU to the development system. When the bit is cleared, data is written to the CPU or to memory from the development system.

Operand Size

For sized operations, this field specifies the operand data size. All addresses are expressed as 32-bit absolute values. The size field is encoded as listed in Table 16-4.

Table 16-4. BDM Size Field Encoding

ENCODING	OPERAND SIZE	BIT VALUES
00	Byte	8 bits
01	Word	16 bits
10	Longword	32 bits
11	Reserved	

Address / Data (A/D) Field

The A/D field is used in commands that operate on address and data registers in the processor. It determines whether the register field specifies a data or address register. A one indicates an address register; zero, a data register.

Register Field

In commands that operate on processor registers, this field specifies which register is selected. The field value contains the register number.

Extension Word(s) (as required):

Certain commands require extension words for addresses and/or immediate data. Addresses require two extension words because only absolute long addressing is permitted. Immediate data can be either one or two words in length—byte and word data each require a single extension word; longword data requires two words. Both 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.”

16.3.3.3 COMMAND SEQUENCE DIAGRAM. A command sequence diagram (see Figure 16-6) illustrates the serial bus traffic for each command. Each bubble in the diagram represents a single 17-bit transfer across the bus. The top half in each bubble corresponds to the data transmitted by the development system to the Debug Module; the bottom half corresponds to the data returned by the Debug Module in response to the previous development system commands. Command and result transactions are overlapped to minimize latency.

The cycle in which the command is issued contains the development system command mnemonic (in this example, “read memory location”). During the same cycle, the Debug Module responds with either the low-order results of the previous command or a command complete status (if no results were required).

During the second cycle, the development system supplies the high-order 16 bits of the memory address. The Debug Module returns a “not ready” response unless the received command was decoded as unimplemented, in which case the response data is the illegal command encoding. If an illegal command response occurs, the development system should retransmit the command.

NOTE

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.

In the third cycle, the development system supplies the low-order 16 bits of a memory address. The Debug Module always returns the “not ready” response in this cycle. At the completion of the third cycle, the Debug Module initiates a memory read operation. Any serial transfers that begin while the memory access is in progress return the “not ready” response.

Results are returned in the two serial transfer cycles following the completion of the memory access. The data transmitted to the Debug Module during the final transfer is the opcode for the following command. If a memory or register access is terminated with a bus error, the error status (S=1, DATA=\$0001) is returned in place of the result data.

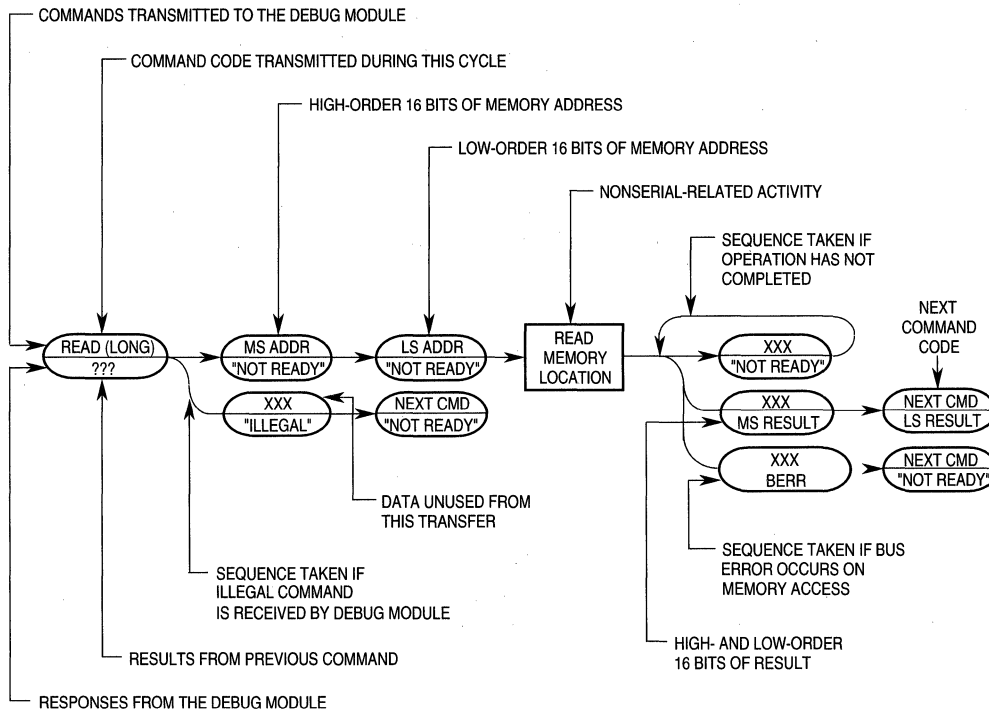


Figure 16-6. Command Sequence Diagram

16.3.3.4 COMMAND SET DESCRIPTIONS. The BDM command set is summarized in Table 16-3. Subsequent paragraphs contain detailed descriptions of each command.

Note

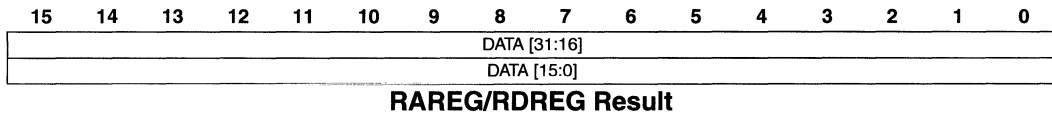
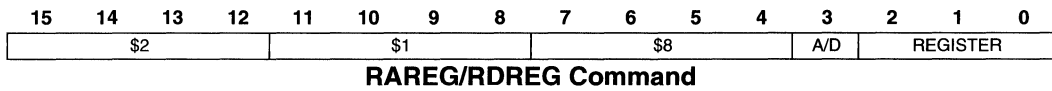
The BDM status bit (S) is zero for normally-completed commands, while illegal commands, “not ready” responses and bus-errored transfers return a logic one in the S bit. Refer to **Section 16.3.2 BDM Serial Interface** for information on the serial packet receive packet format

16

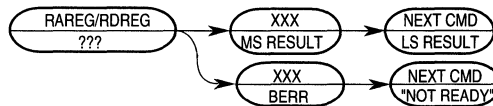
Unassigned command opcodes are reserved by Motorola for future expansion. All unused command formats within any revision level performs a NOP and return the ILLEGAL command response.

16.3.3.4.1 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.

Formats:



Command Sequence:

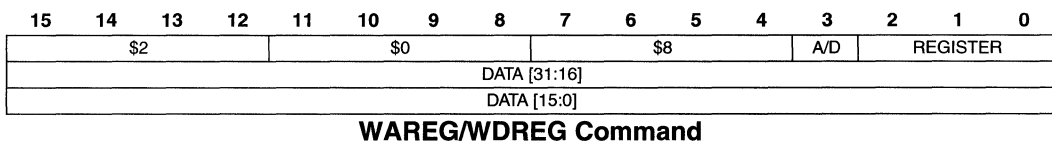


Operand Data:
None

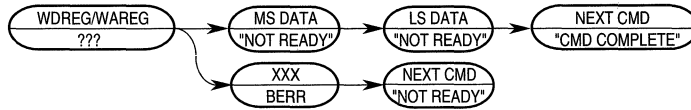
Result Data:
The contents of the selected register are returned as a longword value. The data is returned most significant word first.

16.3.3.4.2 Write A/D Register (WAREG/WDREG). The operand longword data is written to the specified address or data register. All 32 register bits are altered by the write. A bus error response is returned if the CPU core is not halted.

Command Formats:



Command Sequence:



Operand Data:

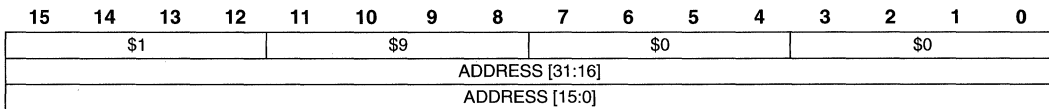
Longword data is written into the specified address or data register. The data is supplied most significant word first.

Result Data:

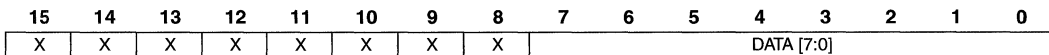
Command complete status is indicated by returning the data \$FFFF (with the status bit cleared) when the register write is complete.

16.3.3.4.3 Read Memory Location (READ). Read the operand data from the memory location specified by the longword address. The address space is defined by the contents of the low-order 5 bits {TT, TM} of the BDM Address Attribute Register (BAAR). The hardware forces the low-order bits of the address to zeros for word and longword accesses to ensure that operands are always accessed on natural boundaries: words on 0-modulo-2 addresses, longwords on 0-modulo-4 addresses.

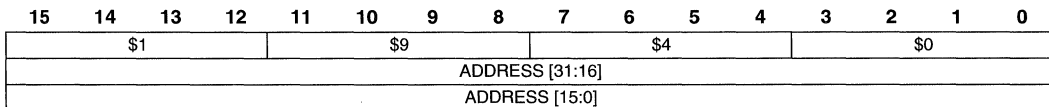
Formats:



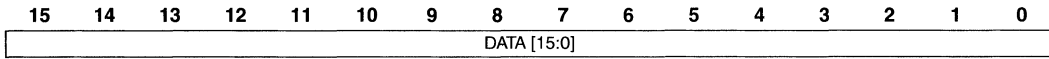
Byte READ Command



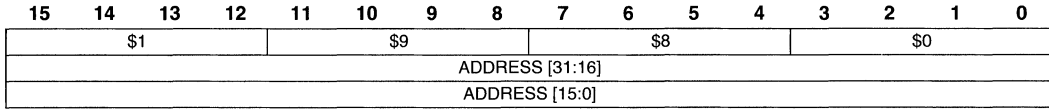
Byte READ Result



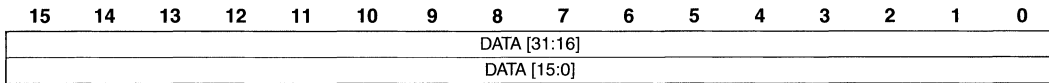
Word READ Command



Word READ Result

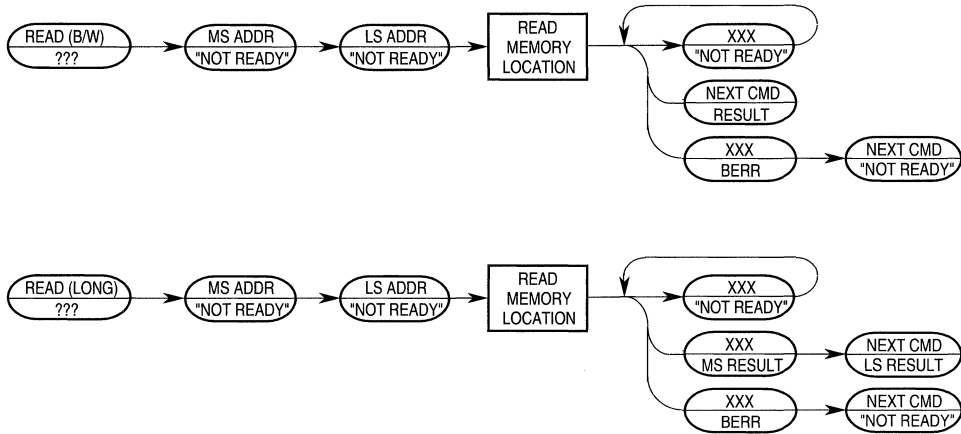


Long READ Command



Long READ Result

Command Sequence:



Operand Data:

The single operand is the longword address of the requested memory location.

Result Data:

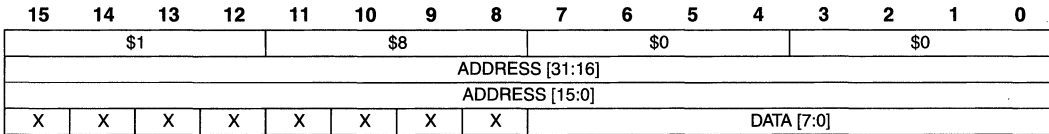
The requested data is returned as either a word or longword. Byte data is returned in the least significant byte of a word result, with the upper byte undefined. Word results return 16 bits of significant data; longword results return 32 bits. A value of \$0001 (with the status bit set) is returned if a bus error occurs.

16.3.3.4.4 Write Memory Location (WRITE). Write the operand data to the memory location specified by the longword address. The address space is defined by the contents

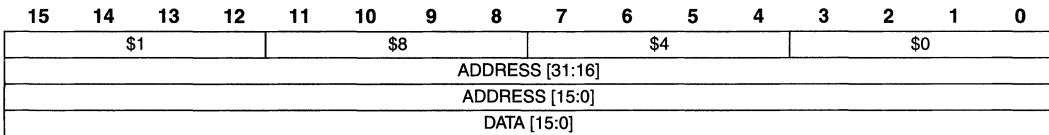
DEBUG SUPPORT

of the low-order 5 bits {TT, TM} of the BDM Address Attribute Register (BAAR). The hardware forces the low-order bits of the address to zeros for word and longword accesses to ensure that operands are always accessed on natural boundaries: words on 0-modulo-2 addresses, longwords on 0-modulo-4 addresses.

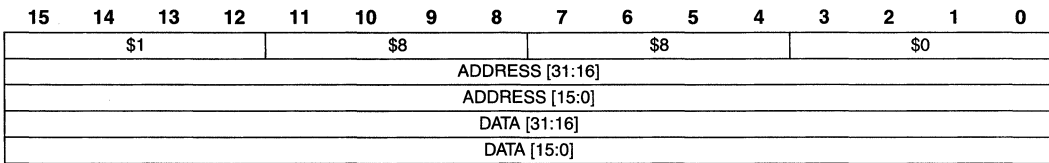
Formats:



Byte WRITE Command

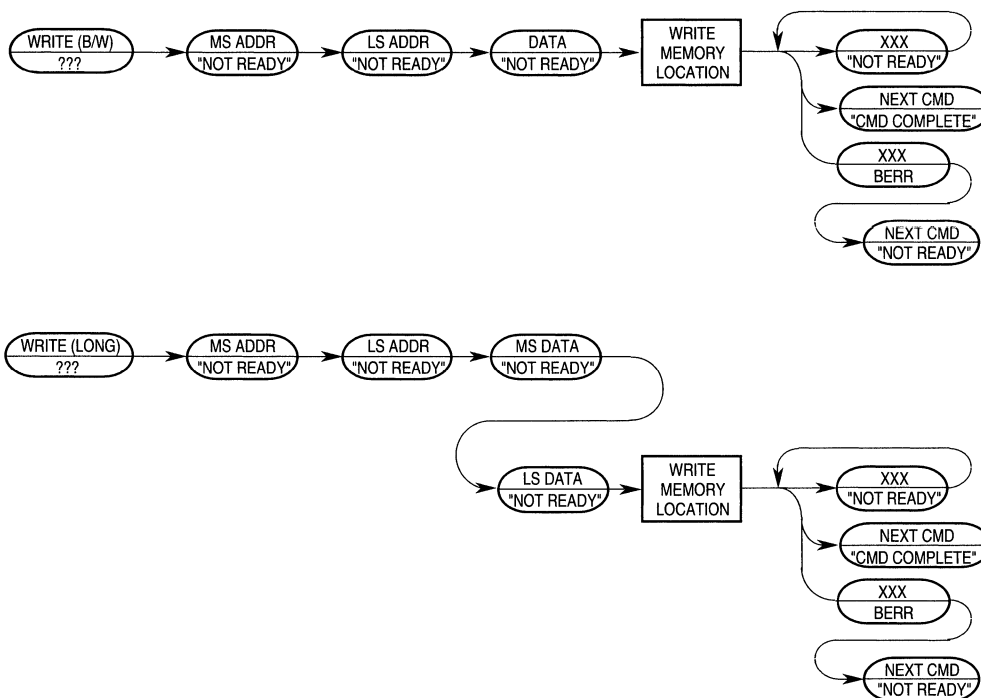


Word WRITE Command



Long WRITE Command

Command Sequence:



Operand Data:

Two operands are required for this instruction. The first operand is a longword absolute address that specifies a location to which the operand data is to be written. The second operand is the data. Byte data is transmitted as a 16-bit word, justified in the least significant byte; 16- and 32-bit operands are transmitted as 16 and 32 bits, respectively.

Result Data:

Command complete status is indicated by returning the data \$FFFF (with the status bit cleared) when the register write is complete. A value of \$0001 (with the status bit set) is returned if a bus error occurs.

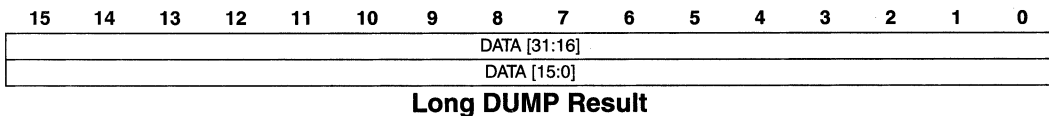
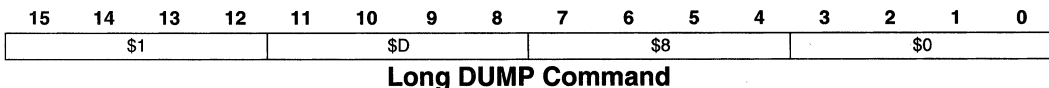
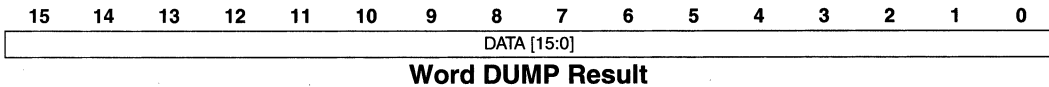
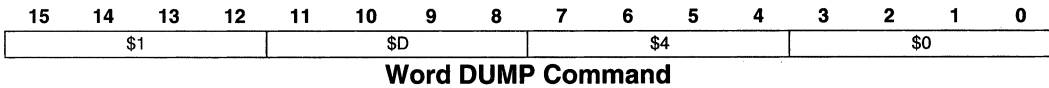
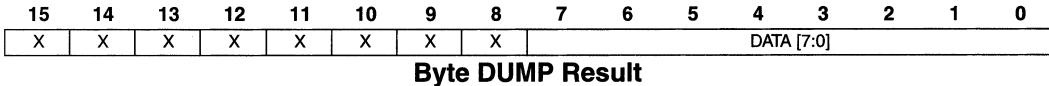
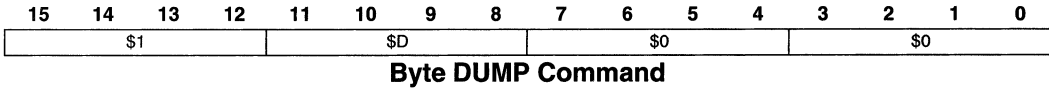
16.3.3.4.5 Dump Memory Block (DUMP). DUMP is used in conjunction with the READ command to access large blocks of memory. An initial READ is executed to set up the starting address of the block and to retrieve the first result. The DUMP command retrieves subsequent operands. The initial address is incremented 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

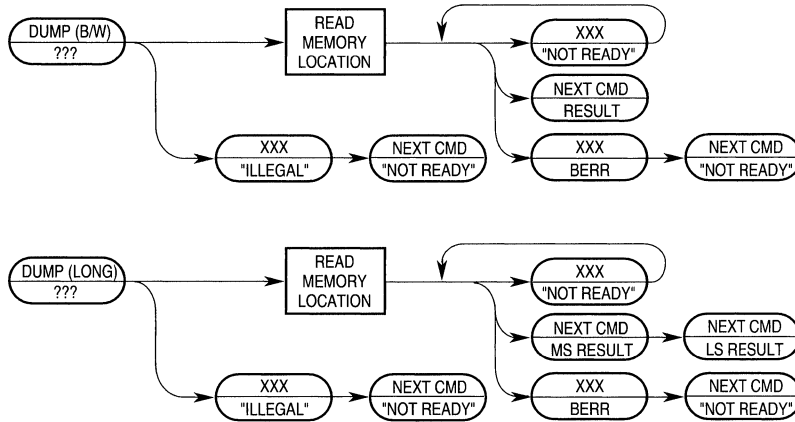
The DUMP command does not check for a valid address — DUMP is a valid command only when preceded by another DUMP, NOP or by a READ command. Otherwise, an illegal command response is returned. The NOP command can be used for intercommand padding without corrupting the address pointer.

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

Command Formats:



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 \$0001 (with the status bit set) is returned if a bus error occurs.

16.3.3.4.6 Fill Memory Block (FILL). FILL is used in conjunction 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.

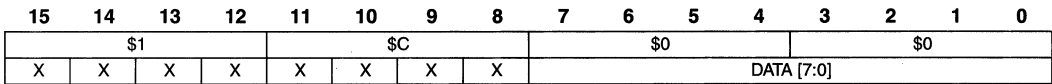
NOTE

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

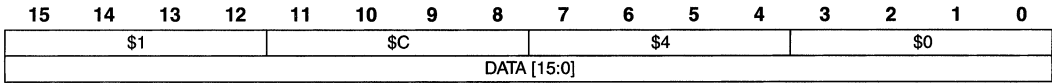
The size field is examined each time a FILL command is processed, allowing the operand size to be altered dynamically.

DEBUG SUPPORT

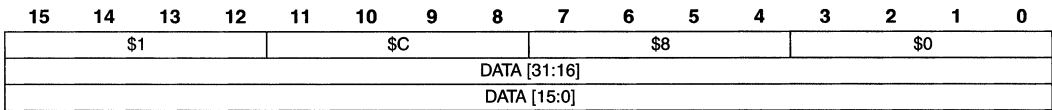
Formats:



Byte FILL Command

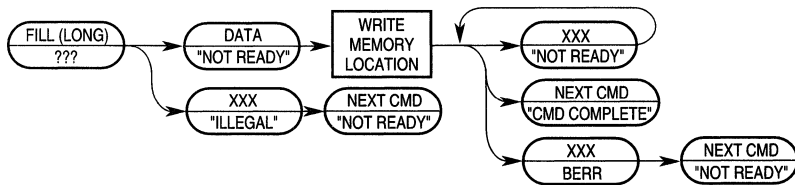
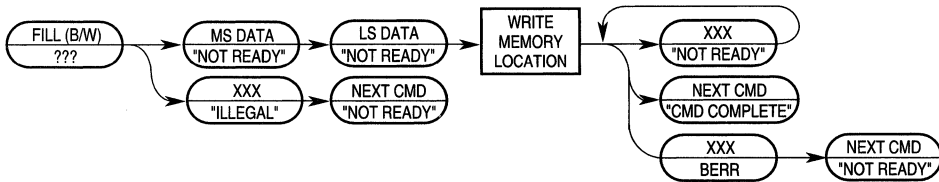


Word FILL Command



Long FILL Command

Command Sequence:



Operand Data:

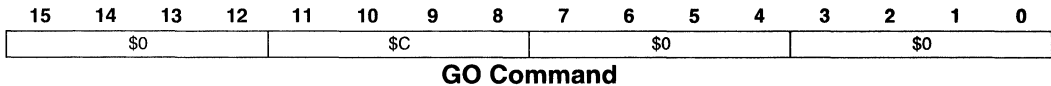
A single operand is data to be written to the memory location. Byte data is transmitted as a 16-bit word, justified in the least significant byte; 16- and 32-bit operands are transmitted as 16 and 32 bits, respectively.

Result Data:

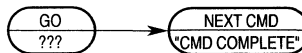
Command complete status is indicated by returning the data \$FFFF (with the status bit cleared) when the register write is complete. A value of \$0001 (with the status bit set) is returned if a bus error occurs.

16.3.3.4.7 Resume Execution (GO). The pipeline is flushed and refilled before resuming normal instruction execution. Prefetching begins at the current PC and current privilege level. If any register (e.g., the PC or SR) was altered by a BDM command while halted, the updated value is used as the prefetching resumes.

Formats:



Command Sequence:



Operand Data:

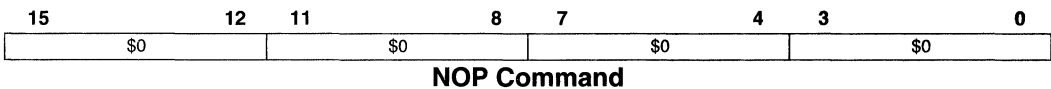
None

Result Data:

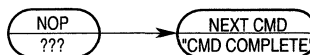
The “command complete” response (\$0FFFF) is returned during the next shift operation.

16.3.3.4.8 No Operation (NOP). NOP performs no operation and may be used as a null command where required.

Formats:



Command Sequence:



Operand Data:

None

Result Data:

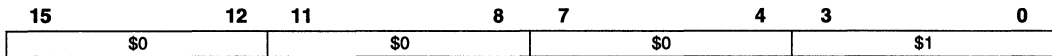
The “command complete” response, \$FFFF (with the status bit cleared), is returned during the next shift operation.

16.3.3.4.9 Synchronize PC to the PST/DDATA Lines(SYNC_PC). Capture the current PC and display 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 the BTB bits of the CSR (CSR [9:8]). The specific sequence of PST and DDATA values is defined below :

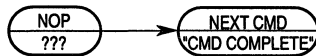
Debug signals a SYNC_PC command is pending. CPU completes the current instruction. CPU forces an instruction fetch to the next PC, generates a PST = \$5 value indicating a “taken branch” and signals DDATA. DDATA captures the instruction address corresponding to the PC. DDATA generates a PST marker (\$9 - \$B) as defined by CSR. BTB and displays the captured PC address.

This command can be used to dynamically access the PC for performance monitoring. The execution of this command is considerably less obtrusive to the real-time operation of an application than a “halt-CPU/read-PC/resume” command sequence.

Format:



Command Sequence:



Operand Data:

None

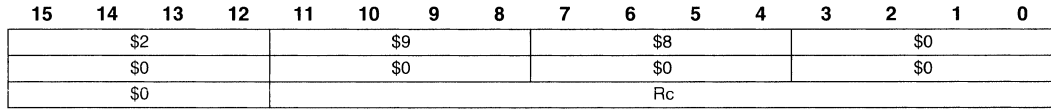
Result Data:

The "command complete" response, \$FFFF (with the status bit cleared), is returned during the next shift operation.

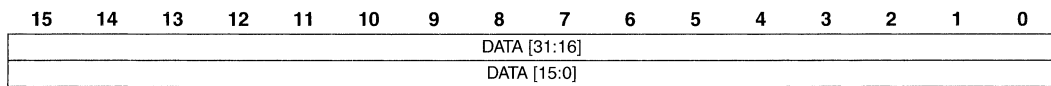
16.3.3.4.10 Read Control Register (RCREG). Read the selected control register and return the 32-bit result. Accesses to the processor/memory control registers are always 32

bits in size, regardless of the implemented register width. The second and third words of the command effectively form a 32-bit address used by the Debug Module 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.

Formats:



RCREG Command



RCREG Result

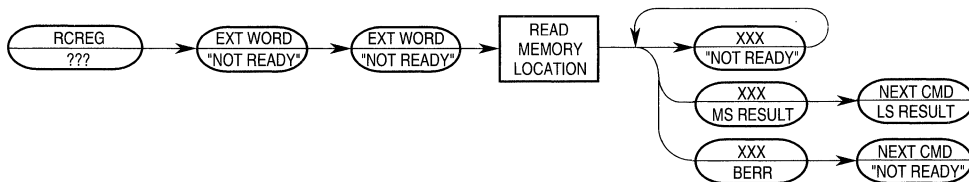
Rc encoding:

Table 16-5. Control Register Map

Rc	REGISTER DEFINITION
\$002	Cache Control Register (CACR)
\$004	Access Control Register 0 (ACR0)
\$005	Access Control Register 1 (ACR1)
\$801	Vector BASE Register (VBR)
\$804	MAC Status Register (MACSR)†
\$805	MAC Mask Register (MASK)†
\$806	MAC Accumulator (ACC)†
\$80E	Status Register (SR)
\$80F	Program Register (PC)
\$C00	ROM Base Address Register (ROMBAR)
\$C04	RAM Base Address Register (RAMBAR)

NOTE: †Available if the optional MAC unit is present.

Command Sequence:



Operand Data:

The single operand is the 32-bit Rc control register select field.

Result Data:

The contents of the selected control register are returned as a longword value. The data is returned most significant word first. For those control registers with widths less than 32 bits, only the implemented portion of the register is guaranteed to be correct. The remaining bits of the longword are undefined.

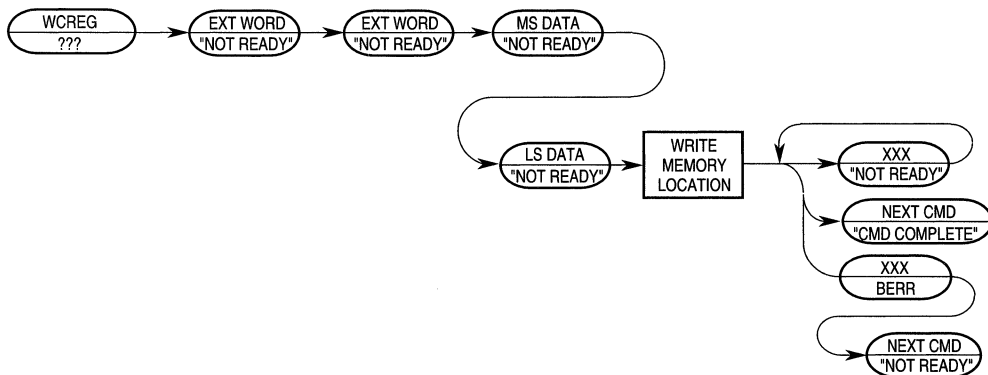
16.3.3.4.11 Write Control Register (WCREG). The operand (longword) data is written to the specified control register. The write alters all 32 register bits.

Formats:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
\$2				\$8				\$8				\$0							
\$0				\$0				\$0				\$0							
\$0				Rc															
DATA [31:16]																			
DATA [15:0]																			

WCREG Command

Command Sequence:



Operand Data:

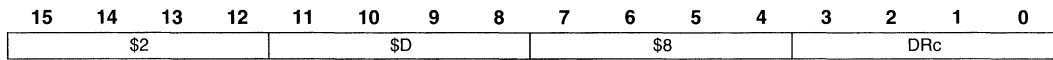
Two operands are required for this instruction. The first long operand selects the register to which the operand data is to be written. The second operand is the data.

Result Data:

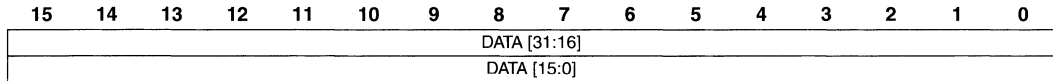
Successful write operations return a \$FFFF. Bus errors on the write cycle are indicated by the assertion of bit 16 in the status message and by a data pattern of \$0001.

16.3.3.4.12 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 the CSR (DRc = \$0).

Command Formats:



RDMREG BDM Command



RDMREG BDM Result

DRc encoding:

Table 16-6. Definition of DRc Encoding - Read

DRc[3:0]	DEBUG REGISTER DEFINITION	MNEMONIC	INITIAL STATE
\$0	Configuration/Status	CSR	\$0
\$1-\$F	Reserved	-	-

Command Sequence:



Operand Data:

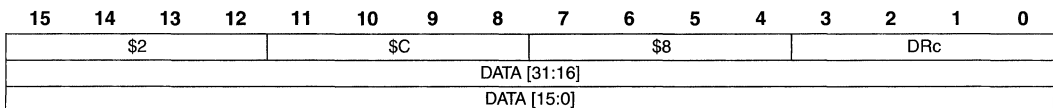
None

Result Data:

The contents of the selected debug register are returned as a longword value. The data is returned most significant word first.

16.3.3.4.13 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. The `DSCLK` signal must be inactive while debug module register writes from the CPU accesses are performed using the `WDEBUG` instruction.

Command Format:



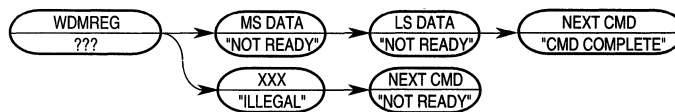
WDMREG BDM Command

DRc encoding:

Table 16-7. Definition of DRc Encoding - Write

DRc[3:0]	DEBUG REGISTER DEFINITION	MNEMONIC	INITIAL STATE
\$0	Configuration/Status	CSR	\$0
\$1-\$4	Reserved	-	-
\$5	BDM Address Attribute	BAAR	\$5
\$6	Bus Attributes and Mask	AATR	\$5
\$7	Trigger Definition	TDR	\$0
\$8	PC Breakpoint	PBR	-
\$9	PC Breakpoint Mask	PBMR	-
\$A-\$B	Reserved	-	-
\$C	Operand Address High Breakpoint	ABHR	-
\$D	Operand Address Low Breakpoint	ABLR	-
\$E	Data Breakpoint	DBR	-
\$F	Data Breakpoint Mask	DBMR	-

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 (\$0FFFF) is returned when register write is complete.

16.3.3.4.14 Unassigned Opcodes. Unassigned command opcodes are reserved by Motorola. All unused command formats within any revision level performs a NOP and return the ILLEGAL command response.

16.4 REAL-TIME DEBUG SUPPORT

The ColdFire Family provides support for the debug of real-time applications. For these types of embedded systems, the processor cannot be halted during debug, but must continue to operate. 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 a number of hardware resources to support various hardware breakpoint functions. Specifically, three types of breakpoints are supported: PC with mask, operand address range, and data with mask. These three basic breakpoints can be configured into one- or two-level triggers with the exact trigger response also

programmable. The Debug Module programming model is accessible from either the external development system using the serial interface or from the processor's supervisor programming model using the WDEBUG instruction.

16.4.1 Theory of Operation

The breakpoint hardware can be configured to respond to triggers in several ways. The desired response is programmed into the Trigger Definition Register. In all situations where a breakpoint triggers, an indication is provided on the DDATA output port, when not displaying captured operands or branch addresses, as shown in Table 16-8.

Table 16-8. DDATA[3:0], CSR[31:28] Breakpoint Response

DDATA[3:0], CSR[31:28]	BREAKPOINT STATUS
\$000x	No Breakpoints Enabled
\$001x	Waiting for Level 1 Breakpoint
\$010x	Level 1 Breakpoint Triggered
\$101x	Waiting for Level 2 Breakpoint
\$110x	Level 2 Breakpoint Triggered
All other encodings are reserved for future use.	

The breakpoint status is also posted in the CSR.

The BDM instructions load and configure the desired breakpoints using the appropriate registers. As the system operates, a breakpoint trigger generates a response as defined in the TDR. If the system can tolerate the processor being halted, a BDM-entry can be used. With the TRC bits of the TDR equal to \$1, the breakpoint trigger causes the core to halt as reflected in the PST = \$F status. **For PC breakpoints, the halt occurs before the targeted instruction is executed. For address and data breakpoints, the processor may have executed several additional instructions. As a result, trigger reporting is considered imprecise.**

If the processor core cannot be halted, the special debug interrupt can be used. With this configuration, TRC bits of the TDR equal to \$2, the breakpoint trigger is converted into a debug interrupt to the processor. This interrupt 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 immediately (before the execution of the targeted instruction). This is possible because the PC breakpoint comparison is enabled at the same time the interrupt sampling occurs. For the address and data breakpoints, the reporting is considered imprecise because several additional instructions may be executed after the triggering address or data is seen.

Once the debug interrupt is recognized, the processor aborts execution and initiates exception processing. At the initiation of the exception processing, the core enters emulator mode. After the standard 8-byte exception stack is created, the processor fetches a unique exception vector, 12, from the vector table (Refer to the *ColdFire Family Programmer's Reference Manual Rev 1.0* MCF5200PRM/AD).

Execution continues at the instruction address contained in this exception vector. All interrupts are ignored while in emulator mode. You can program the debug-interrupt handler to perform the necessary context saves using the supervisor instruction set. As an example, this handler may save the state of all the program-visible registers as well as the current context into a reserved memory area.

Once the required operations are completed, the return-from-exception (RTE) instruction is executed and the processor exits emulator mode. Once the debug interrupt handler has completed its execution, the external development system can then access the reserved memory locations using the BDM commands to read memory.

In the Rev. A implementation, if a hardware breakpoint (e.g., a PC trigger) is left unmodified by the debug interrupt service routine, another debug interrupt is generated after the RTE instruction completes execution. In the Rev. B design, the hardware has been modified to inhibit the generation of another debug interrupt during the first instruction after the RTE exits emulator mode. This behaviour is consistent with the existing logic involving trace mode, where the execution of the first instruction occurs before another trace exception is generated. This Rev. B enhancement disables all hardware breakpoints until the first instruction after the RTE has completed execution, regardless of the programmed trigger response.

16.4.1.1 EMULATOR MODE. Emulator mode is used to facilitate non-intrusive emulator functionality. This mode can be entered in three different ways:

- The EMU bit in the CSR may be programmed to force the ColdFire processor to begin execution in emulator mode. This bit is only examined when RSTI is negated and the processor begins reset exception processing. It may be set while the processor is halted before the reset exception processing begins. Refer to **Section 16.3.1 CPU Halt**.
- A debug interrupt always enters emulation mode when the debug interrupt exception processing begins.
- The TCR bit in the CSR may be programmed to force the processor into emulation mode when trace exception processing begins.

During emulation mode, the ColdFire processor exhibits the following properties:

- All interrupts are ignored, including level seven.
- If the MAP bit of the CSR is set, all memory accesses are forced into a specially mapped address space signalled by TT = \$2, TM = \$5 or \$6. This includes the stack frame writes and the vector fetch for the exception which forced entry into this mode.
- If the MAP bit in the CSR is set, all caching of memory accesses is disabled. Additionally, the SRAM module is disabled while in this mode.

The return-from-exception (RTE) instruction exits emulation mode. The processor status output port provides a unique encoding for emulator mode entry (\$D) and exit (\$7).

16.4.1.2 DEBUG MODULE HARDWARE.

16.4.1.2.1 Reuse of Debug Module Hardware (Rev. A). The Debug Module implementation provides a common hardware structure for both BDM and breakpoint functionality. Several structures are used for both BDM and breakpoint purposes. Table 16-9 identifies the shared hardware structures.

Table 16-9. Shared BDM/Breakpoint Hardware

REGISTER	BDM FUNCTION	BREAKPOINT FUNCTION
AATR	Bus Attributes for All Memory Commands	Attributes for Address Breakpoint
ABHR	Address for All Memory Commands	Address for Address Breakpoint
DBR	Data for All BDM Write Commands	Data for Data Breakpoint

The shared use of these hardware structures means the loading of the register to perform any specified function is destructive to the shared function. For example, if an operand address breakpoint is loaded into the Debug Module, a BDM command to access memory overwrites the breakpoint. If a data breakpoint is configured, a BDM write command overwrites the breakpoint contents.

16.4.1.2.2 The New Debug Module Hardware (Rev. B). The new Debug Module implementation has added hardware registers so that there are no restrictions concerning the interaction between BDM commands and the use of the hardware breakpoint logic. In some cases, the additional hardware is not program-visible, while in other cases, there have been extensions to the Debug Module programming model. As example, consider the following two registers:

The hardware register containing the BDM memory address is not a program-visible resource. Rather, it is a hardware register loaded automatically during the execution of a BDM commands. In the Rev B design, the execution of a BDM command does not affect the hardware breakpoint logic unless those registers are specifically accessed.

The other register added to the Debug Module programming model is the BDM Address Attribute Register (BAAR). It is mapped to an DRc[3:0] address of \$5. This 8-bit register is equivalent in the format of the low-order byte of the AATR register (Refer to section 15.4.2.7). This register specifies the memory space attributes associated with all BDM memory-referencing commands.

16.4.2 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. All of these registers are treated as 32-bit quantities, regardless of the actual number of bits in the implementation. The registers, known as the debug control registers, are accessed through the BDM port using two new BDM commands: WDMREG and RDMREG. These commands contain a 4-bit field, DRc, which specifies the particular register being accessed.

These registers are also accessible from the processor’s supervisor programming model through the execution of the WDEBUG instruction. Thus, the breakpoint hardware within the Debug Module may be accessed by the external development system using the serial interface, or by the operating system running on the processor core. It is the responsibility of the software to guarantee that all accesses to these resources are serialized and logically consistent. The hardware provides a locking mechanism in the CSR to allow the external development system to disable any attempted writes by the processor to the breakpoint registers (setting IPW = 1). The BDM commands must not be issued if the ColdFire processor is accessing the Debug Module registers using the WDEBUG instruction.

Figure 16-7 illustrates the Debug Module programming model.

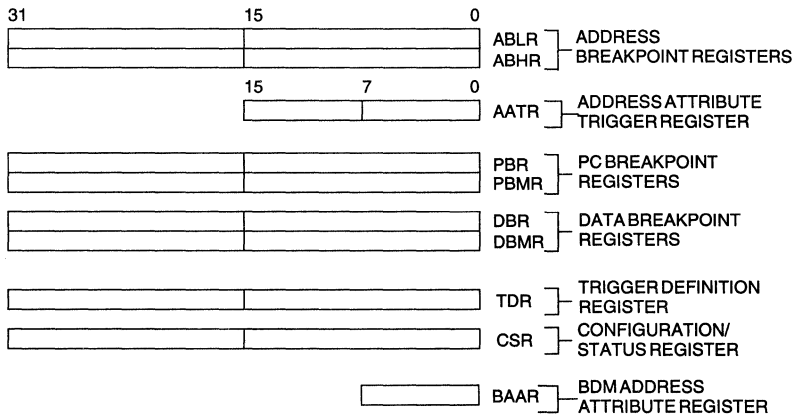


Figure 16-7. Debug Programming Model

16.4.2.1 ADDRESS BREAKPOINT REGISTERS (ABLR, ABHR). The address breakpoint registers define a region in the operand address space of the processor that can be used as part of the trigger. The full 32-bits of the ABLR and ABHR values are compared with the address for all transfers on the processor’s high-speed local bus. The trigger definition register (TDR) determines if the trigger is the inclusive range bound by ABLR and ABHR, all addresses outside this range, or the address in ABLR only. The ABHR is accessible in supervisor mode as debug control register \$C using the WDEBUG instruction and via the BDM port using the RDMREG and WDMREG commands. The ABLR is accessible in supervisor mode as debug control register \$D using the WDEBUG instruction and via the BDM port using the WDMREG commands. The ABHR is overwritten by the BDM hardware when accessing memory as described in **Section 16.4.1.2 Debug Module Hardware**.

BITS	31	0
FIELD	ADDRESS	
RESET	-	
R/W	W	

Address Breakpoint Low Register (ABLR)

Field Definition:

ADDRESS[31:0]–Low Address

This field contains the 32-bit address which marks the lower bound of the address breakpoint range. Additionally, if a breakpoint on a specific address is required, the value is programmed into the ABLR.

BITS	31	0
FIELD	ADDRESS	
RESET	-	
R/W	W	

Address Breakpoint High Register (ABHR)

Field Definition:

ADDRESS[31:0]–High Address

This field contains the 32-bit address which marks the upper bound of the address breakpoint range.

16.4.2.2 ADDRESS ATTRIBUTE TRIGGER REGISTER (AATR). The AATR defines the address attributes and a mask to be matched in the trigger. The AATR value is compared with the address attribute signals from the processor’s local high-speed bus, as defined by the setting of the TDR. The AATR is accessible in supervisor mode as debug control register \$6 using the WDEBUG instruction and via the BDM port using the WDMREG command. The lower five bits of the AATR are also used for BDM command definition to define the address space for memory references as described in **Section 16.4.1.2 Debug Module Hardware**.

BITS	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	0	0		0		0		0	0	0		101				
R/W	W	W		W		W		W	W	W		W				

Address Attribute Trigger Register (AATR)

Field Definitions:

RM[15]—Read/Write Mask

This field corresponds to the R-field. Setting this bit causes R to be ignored in address comparisons.

SZM[14:13]—Size Mask

This field corresponds to the SZ field. Setting a bit in this field causes the corresponding bit in SZ to be ignored in address comparisons.

TTM[12:11]—Transfer Type Mask

This field corresponds to the TT field. Setting a bit in this field causes the corresponding bit in TT to be ignored in address comparisons.

TMM[10:8]—Transfer Modifier Mask

This field corresponds to the TM field. Setting a bit in this field causes the corresponding bit in TM to be ignored in address comparisons.

R[7]—Read/Write

This field is compared with the R/W signal of the processor's local bus.

SZ[6:5]—Size

This field is compared to the size signals of the processor's local bus. These signals indicate the data size for the bus transfer.

- 00 = Longword
- 01 = Byte
- 10 = Word
- 11 = Reserved

TT[4:3]—Transfer Type

This field is compared with the transfer type signals of the processor's local bus. These signals indicate the transfer type for the bus transfer. These signals are always encoded as if the ColdFire 53xx is in the ColdFire IACK mode.

- 00 = Normal Processor Access
- 01 = Reserved
- 10 = Emulator Mode Access
- 11 = Acknowledge/CPU Space Access

These bits also define the TT encoding for BDM memory commands. In this case, the 01 encoding generates an alternate master access (For backward compatibility).

TM[2:0]—Transfer Modifier

This field is compared with the transfer modifier signals of the processor's local bus. These signals provide supplemental information for each transfer type. These signals are always

encoded as if the processor is operating in the ColdFire IACK mode. The encoding for normal processor transfers (TT = 0) is:

- 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

The encoding for emulator mode transfers (TT = 10) is:

- 0xx = Reserved
- 100 = Reserved
- 101 = Emulator Mode Data Access
- 110 = Emulator Mode Code Access
- 111 = Reserved

The encoding for acknowledge/CPU space transfers (TT = 11) is:

- 000 = CPU Space Access
- 001 = Interrupt Acknowledge Level 1
- 010 = Interrupt Acknowledge Level 2
- 011 = Interrupt Acknowledge Level 3
- 100 = Interrupt Acknowledge Level 4
- 101 = Interrupt Acknowledge Level 5
- 110 = Interrupt Acknowledge Level 6
- 111 = Interrupt Acknowledge Level 7

These bits also define the TM encoding for BDM memory commands (For backward compatibility).

16.4.2.3 PROGRAM COUNTER BREAKPOINT REGISTER (PBR, PBMR). The PC breakpoint registers define a region in the code address space of the processor that can be used as part of the trigger. The PBR value is masked by the PBMR value, allowing only those bits in PBR that have a corresponding zero in PBMR to be compared with the processor's program counter register, as defined in the TDR. The PBR is accessible in supervisor mode as debug control register \$8 using the WDEBUG instruction and via the BDM port using the RDMREG and WDMREG commands. The PBMR is accessible in supervisor mode as debug control register \$9 using the WDEBUG instruction and via the BDM port using the WDMREG command.

BITS	31													0
FIELD	ADDRESS													
RESET	-													
R/W	W													

Program Counter Breakpoint Register (PBR)

Field Definition:

ADDRESS[31:0]—PC Breakpoint Address

This field contains the 32-bit address to be compared with the PC as a breakpoint trigger.

BITS	31													0
FIELD	MASK													
RESET	-													
R/W	W													

Program Counter Breakpoint Mask Register (PBMR)

Field Definition:

MASK[31:0]—PC Breakpoint Mask

This field contains the 32-bit mask for the PC breakpoint trigger. A zero in a bit position causes the corresponding bit in the PBR to be compared to the appropriate bit of the PC. A one causes that bit to be ignored.

16.4.2.4 DATA BREAKPOINT REGISTER (DBR, DBMR). The data breakpoint registers define a specific data pattern that can be used as part of the trigger into debug mode. The DBR value is masked by the DBMR value, allowing only those bits in DBR that have a corresponding zero in DBMR to be compared with the data value from the processor’s local bus, as defined in the TDR. The DBR is accessible in supervisor mode as debug control register \$E using the WDEBUG instruction and via the BDM port using the RDMREG and WDMREG commands. The DBMR is accessible in supervisor mode as debug control register \$F using the WDEBUG instruction and via the BDM port using the WDMREG command. The DBR is overwritten by the BDM hardware when accessing memory as described in **Section 16.4.1.2 Debug Module Hardware**.

16

BITS	31													0
FIELD	ADDRESS													
RESET														
R/W	W													

Data Breakpoint Register (DBR)

Field Definition:

DATA[31:0]—Data Breakpoint Value

This field contains the 32-bit value to be compared with the data value from the processor's local bus as a breakpoint trigger.

BITS	31	0
FIELD	MASK	
RESET	-	
R/W	W	

Data Breakpoint Mask Register (DBMR)

Field Definition:

MASK[31:0]—Data Breakpoint Mask

This field contains the 32-bit mask for the data breakpoint trigger. A zero in a bit position causes the corresponding bit in the DBR to be compared to the appropriate bit of the internal data bus. A one causes that bit to be ignored.

The data breakpoint register supports both aligned and misaligned references. The relationship between the processor address, the access size, and the corresponding location within the 32-bit data bus is shown in Table 16-10.

Table 16-10. Access Size and Operand Data Location

ADDRESS[1:0]	ACCESS SIZE	OPERAND LOCATION
00	Byte	Data[31:24]
01	Byte	Data[23:16]
10	Byte	Data[15:8]
11	Byte	Data[7:0]
0x	Word	Data[31:16]
1x	Word	Data[15:0]
xx	Long	Data[31:0]

16.4.2.5 TRIGGER DEFINITION REGISTER (TDR). The TDR configures the operation of the hardware breakpoint logic within the Debug Module and controls the actions taken under the defined conditions. The breakpoint logic may be configured as a one- or two-level trigger, where bits [31:16] of the TDR define the 2nd level trigger and bits [15:0] define the first level trigger. The TDR is accessible in supervisor mode as debug control register \$7 using the WDEBUG instruction and via the BDM port using the WDMREG command.

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	TRC		EBL	EDLW	EDWL	EDWU	EDLL	EDLM	EDUM	EDUU	DI	EAI	EAR	EAL	EPC	PCI
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

BITS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	LXT		EBL	EDLW	EDWL	EDWU	EDLL	EDLM	EDUM	EDUU	DI	EAI	EAR	EAL	EPC	PCI
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Trigger Definition Register (TDR)

Field Definitions:

TRC—Trigger Response Control

The trigger response control determines how the processor is to respond to a completed trigger condition. The trigger response is always displayed on the DDATA pins.

- 00 = display on DDATA only
- 01 = processor halt
- 10 = debug interrupt
- 11 = reserved

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)

EBL—Enable Breakpoint Level

If set, this bit serves as the global enable for the breakpoint trigger. If cleared, all breakpoints are disabled.

EDLW—Enable Data Breakpoint for the Data Longword

If set, this bit enables the data breakpoint based on the entire processor's local data bus. The assertion of any of the ED bits enables the data breakpoint. If all bits are cleared, the data breakpoint is disabled.

EDWL—Enable Data Breakpoint for the Lower Data Word

If set, this bit enables the data breakpoint based on the low-order word of the processor's local data bus.

EDWU—Enable Data Breakpoint for the Upper Data Word

If set, this bit enables the data breakpoint trigger based on the high-order word of the processor's local data bus.

EDLL—Enable Data Breakpoint for the Lower Lower Data Byte

If set, this bit enables the data breakpoint trigger based on the low-order byte of the low-order word of the processor's local data bus.

EDLM—Enable Data Breakpoint for the Lower Middle Data Byte

If set, this bit enables the data breakpoint trigger based on the high-order byte of the low-order word of the processor's local data bus.

EDUM—Enable Data Breakpoint for the Upper Middle Data Byte

If set, this bit enables the data breakpoint trigger on the low-order byte of the high-order word of the processor's local data bus.

EDUU—Enable Data Breakpoint for the Upper Upper Data Byte

If set, this bit enables the data breakpoint trigger on the high-order byte of the high-order word of the processor's local data bus.

DI—Data Breakpoint Invert

This bit provides a mechanism to invert the logical sense of all the data breakpoint comparators. This can develop a trigger based on the occurrence of a data value not equal to the one programmed into the DBR.

EAI—Enable Address Breakpoint Inverted

If set, this bit enables the address breakpoint based outside the range defined by ABLR and ABHR. The assertion of any of the EA bits enables the address breakpoint. If all three bits are cleared, this breakpoint is disabled.

EAR—Enable Address Breakpoint Range

If set, this bit enables the address breakpoint based on the inclusive range defined by ABLR and ABHR.

EAL—Enable Address Breakpoint Low

If set, this bit enables the address breakpoint based on the address contained in the ABLR.

EPC—Enable PC Breakpoint

If set, this bit enables the PC breakpoint.

PCI-PC 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.

16.4.2.6 CONFIGURATION/STATUS REGISTER (CSR). The CSR defines the debug configuration for the processor and memory subsystem. In addition to defining the microprocessor configuration, this register also contains status information from the breakpoint logic. The CSR is cleared during system reset. The CSR can be read and written by the external development system and written by the supervisor programming model. The CSR is accessible in supervisor mode as debug control register \$0 using the WDEBUG instruction and via the BDM port using the RDMREG and WDMREG commands.

BITS	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	STATUS				FOF	TRG	HALT	BKPT	HRL				-	BKD	-	IPW
RST	0				0	0	0	0	-				-	-	-	0
R/W†	R				R	R	R	R	R				-	-	-	R/W

BITS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	MAP	TRC	EMU	DDC	UHE	BTB	-	NPL	IPI	SSM	-					
RESET	0	0	0	0	0	0	0	0	0	0	-					
R/W†	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	-					

NOTE: †The CSR is a write only register from the programming model. It can be read from and written to via the BDM port.

Configuration/Status Register (CSR)

Field Definitions:

STATUS[31:28]—Breakpoint Status

This 4-bit field provides read-only status information concerning the hardware breakpoints. This field is defined as follows:

- 000x = no breakpoints enabled
- 001x = waiting for level 1 breakpoint
- 010x = level 1 breakpoint triggered
- 101x = waiting for level 2 breakpoint
- 110x = level 2 breakpoint triggered

The breakpoint status is also output on the DDATA port when it is not busy displaying other processor data. A write to the TDR resets this field.

FOF[27]—Fault-on-Fault

If this read-only status bit is set, a catastrophic halt has occurred and forced entry into BDM. This bit is cleared on a read from the CSR.

TRG[26]—Hardware Breakpoint Trigger

If this read-only status bit is set, a hardware breakpoint has halted the processor core and forced entry into BDM. This bit is cleared by reading CSR.

HALT[25]—Processor Halt

If this read-only status bit is set, the processor has executed the HALT instruction and forced entry into BDM. This bit is cleared by reading the CSR.

 $\overline{\text{BKPT}}$ [24]—Breakpoint Assert

If this read-only status bit is set, the BKPT signal was asserted, forcing the processor into BDM. This bit is cleared on a read from the CSR.

HRL[23:20]—Hardware Revision Level

This hardware revision level indicates the level of functionality implemented in the Debug Module. This information could be used by an emulator to identify the level of functionality supported. A zero value would indicate the initial debug functionality. For example, a value of 1 would represent Revision B while a value of 0 would represent the earlier release of Revision A.

BKD[18]—Disable the Normal BKPT Input Signal Functionality

This bit is used to disable the normal BKPT input signal functionality, and allow the assertion of this pin to generate a debug interrupt. If set, the assertion of the $\overline{\text{BKPT}}$ pin is treated as an edge-sensitive event. Specifically, a high-to-low edge on the $\overline{\text{BKPT}}$ pin generates a signal to the processor indicating a debug interrupt. The processor makes this interrupt request pending until the next sample point occurs. At that time, the debug interrupt exception is initiated. In the ColdFire architecture, the interrupt sample point occurs once per instruction. There is no support for any type of “nesting” of debug interrupts.

PCD[17]—PSTCLK Disable

If set, this bit disables the generation of the PSTCLK output signal, and forces this signal to remain quiescent.

IPW[16]—Inhibit Processor Writes to Debug Registers

If set, this bit inhibits any processor-initiated writes to the Debug Module’s programming model registers. This bit can only be modified by commands from the external development system.

MAP[15]—Force Processor References in Emulator Mode

If set, this bit forces the processor to map all references while in emulator mode to a special address space, TT = \$2, TM = \$5 or \$6. If cleared, all emulator-mode references are mapped into supervisor code and data spaces.

TRC[14]—Force Emulation Mode on Trace Exception

If set, this bit forces the processor to enter emulator mode when a trace exception occurs.

EMU[13]–Force Emulation Mode

If set, this bit forces the processor to begin execution in emulator mode. Refer to **Section 16.4.1.1 Emulator Mode**.

DDC[12:11]–Debug Data Control

This 2-bit field provides configuration control for capturing operand data for display on the DDATA port. The encoding is:

- 00 = no operand data is displayed
- 01 = capture all M-Bus write data
- 10 = capture all M-Bus read data
- 11 = capture all M-Bus read and write data

In all cases, the DDATA port displays the number of bytes defined by the operand reference size, i.e., byte displays 8 bits, word displays 16 bits, and long displays 32 bits (one nibble at a time across multiple clock cycles.) Refer to **Section 16.2.1.7 Begin Data Transfer (PST = \$8 - \$B)**.

UHE[10]–User Halt Enable

This bit selects the CPU privilege level required to execute the HALT instruction.

- 0 = HALT is a privileged, supervisor-only instruction
- 1 = HALT is a non-privileged, supervisor/user instruction

BTB[9:8]–Branch Target Bytes

This 2-bit field defines the number of bytes of branch target address to be displayed on the DDATA outputs. The encoding is

- 00 = 0 bytes
- 01 = lower two bytes of the target address
- 10 = lower three bytes of the target address
- 11 = entire four-byte target address

Refer to **Section 16.2.1.5 Begin Execution of Taken Branch (PST = \$5)**.

NPL[6]–Non-Pipelined Mode

If set, this bit forces the processor core to operate in a nonpipeline mode of operation. In this mode, the processor effectively executes a single instruction at a time with no overlap.

When operating in non-pipelined mode, performance is severely degraded. For the V3 design, operation in this mode essentially adds 6 cycles to the execution time of each instruction. Given that the measured Effective Cycles per Instruction for V3 is ~2 cycles/instruction, meaning performance in non-pipeline mode would be ~8 cycles/instruction, or approximately 25% compared to the pipelined performance.

Regardless of the state of CSR[6], if a PC breakpoint is triggered, it is always reported before the instruction with the breakpoint is executed. The occurrence of an address and/or data breakpoint trigger is imprecise in normal pipeline operation. When operating in non-

pipeline mode, these triggers are always reported before the next instruction begins execution. In this mode, the trigger reporting can be considered to be precise.

As previously discussed, the occurrence of an address and/or data breakpoint should always happen before the next instruction begins execution. Therefore the occurrence of the address/data breakpoints should be guaranteed.

IPI[5]—Ignore Pending Interrupts

If set, this bit forces the processor core to ignore any pending interrupt requests signalled while executing in single-instruction-step mode.

SSM[4]—Single-Step Mode

If set, this bit forces the processor core to operate in a single-instruction-step mode. While in this mode, the processor executes a single instruction and then halts. While halted, any of the BDM commands may be executed. On receipt of the GO command, the processor executes the next instruction and then halts again. This process continues until the single-instruction-step mode is disabled.

16.4.2.7 BDM ADDRESS ATTRIBUTE (BAAR). The BAAR register defines the address space for memory-referencing BDM commands. Bits [7:5] are loaded directly from the BDM command, while the low-order 5 bits can be programmed from the external development system. To maintain compatibility with the Rev. A implementation, this register is loaded any time the AATR is written. The BAAR is initialized to a value of \$5, setting “supervisor data” as the default address space.

BITS	7	6	5	4	3	2	1	0
FIELD	R	SZ		TT		TM		
RESET	0	0		0		1	0	1
R/W	W	W		W		W		

BDM ADDRESS ATTRIBUTE REGISTER (BAAR)

Field Definitions:

R[7]—Read/Write

- 0 = Write
- 1 = Read

SZ[6:5]—Size

- 00 = Longword
- 01 = Byte
- 10 = Word
- 11 = Reserved

TT[4:3]—Transfer Type

See the TT definition in the AATR description, Section 16.4.2.2.

TM[2:0]—Transfer Modifier

See the TM definition in the AATR description, Section 16.4.2.2.

16.4.3 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 for the operations that access processor/memory registers:

- Read/Write Address and Data Registers
- Read/Write Control Registers

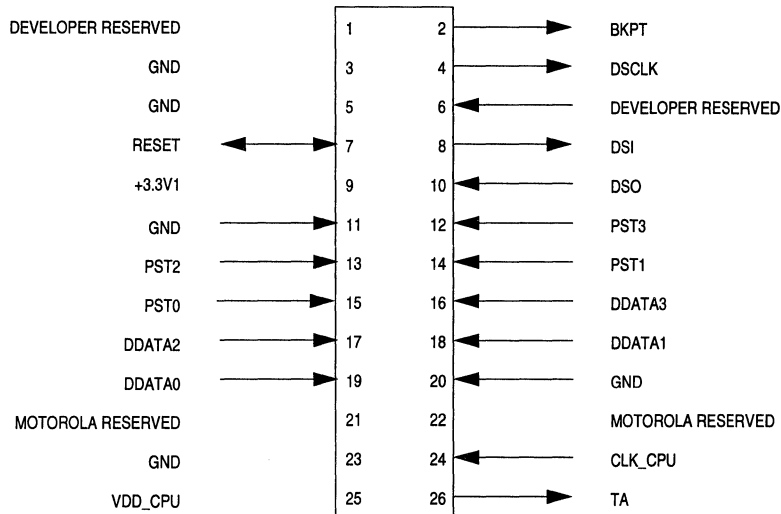
For BDM commands that access memory, the Debug Module requests the processor's local bus. The processor responds by stalling the instruction fetch pipeline and then waiting until all current bus activity is complete. At that time, the processor relinquishes the local bus to allow the Debug Module to perform the required operation. After the conclusion of the Debug Module bus cycle, the processor reclaims ownership of the bus.

The development system must use caution in configuring the breakpoint registers if the processor is executing. The Debug Module does not contain any hardware interlocks, so Motorola recommends that the TDR be disabled while the breakpoint registers are being loaded. At the conclusion of this process, the TDR can be written to define the exact trigger. This approach guarantees that no spurious breakpoint triggers occur.

Because there are no hardware interlocks in the debug unit, no BDM operations are allowed while the CPU is writing the debug's registers (BKPTDCLK must be inactive).

16.4.4 Motorola-Recommended BDM Pinout

The ColdFire BDM connector is a 26-pin Berg Connector arranged 2x13, shown in Figure 16-8.



- NOTES: 1. Supplied by target
 2. Pins reserved for BDM developer use. Contact developer.

Figure 16-8. Recommended BDM Connector

SECTION 17

IEEE 1149.1 TEST ACCESS PORT (JTAG)

The MCF5307 includes dedicated user-accessible test logic that is fully compliant with the IEEE standard 1149.1 *Standard Test Access Port and Boundary Scan Architecture*. 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 implementation currently supports circuit board test strategies that are based on the IEEE standard. This architecture provides access to all of the data and chip control pins from the board edge connector through the standard four-pin test access port (TAP) and the active-low JTAG reset pin, $\overline{\text{TRST}}$. The test logic itself uses a static design and is wholly independent of the system logic, except where the JTAG is subordinate to other complimentary test modes (see the **Debug Support** section for more information). When in subordinate mode, the JTAG test logic is placed in reset and the TAP pins can be used for other purposes in accordance with the rules and restrictions set forth using a JTAG compliance-enable pin.

The MCF5307 JTAG implementation can:

- Perform boundary-scan operations to test circuit board electrical continuity
- Bypass the MCF5307 device by reducing the shift register path to a single cell
- Sample the MCF5307 system pins during operation and transparently shift out the result
- Set the MCF5307 output drive pins to fixed logic values while reducing the shift register path to a single cell
- Protect the MCF5307 system output and input pins from backdriving and random toggling (such as during in-circuit testing) by placing all system signal pins to high-impedance state

NOTE

The IEEE Standard 1149.1 test logic cannot be considered completely benign to those planning not to use JTAG capability. You must observe certain precautions to ensure that this logic does not interfere with system or debug operation. Refer to Section 17.6 **Disabling the IEEE 1149.1 Standard Operation**.

17.1 OVERVIEW

Figure 17-1 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.

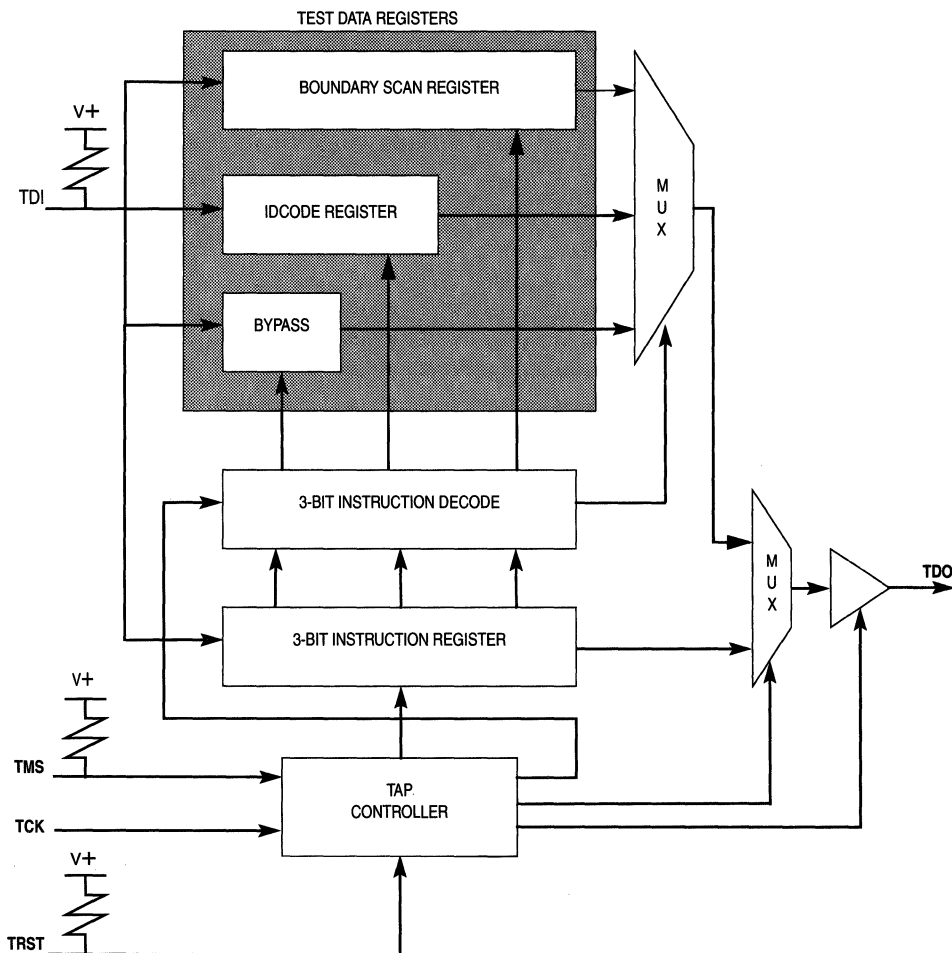


Figure 17-1. JTAG Test Logic Block Diagram

17.2 JTAG SIGNAL DESCRIPTIONS

The JTAG operation on the MCF5307 is enabled when $MTMOD[3:0] = 0001$. The external pin descriptions in Table 17-1 will now apply. Otherwise, the JTAG Test Access Port signals (TCK/TMS/TDI/TDO/TRST) are interpreted as the Debug port pins.

When the compliance-enable state is set for JTAG mode, apply.

Table 17-1. JTAG Pin Descriptions

PIN	DESCRIPTION
TCK	A test clock input that synchronizes test logic operations
TMS	A test mode select input with a default internal pullup resistor that is sampled on the rising edge of TCK to sequence the TAP controller
TDI	A serial test data input with a default internal pullup resistor that is sampled on the rising edge of TCK
TDO	A three-state test data output that is actively driven only in the Shift-IR and Shift-DR controller states and only updates on the falling edge of TCK
TRST	An active-low asynchronous reset with a default internal pullup resistor that forces the TAP controller into the test-logic-reset state.

17.2.1 Test Clock - (TCK)

TCK is the dedicated JTAG test logic clock that is independent of the MCF5307 processor clock. Various JTAG operations occur on the rising or falling edge of TCK. The internal JTAG controller logic is designed such that holding TCK high or low for an indefinite period of time will not cause the JTAG test logic to lose state information. If TCK will not be used, it should be tied to ground.

17.2.2 Test Reset/Development Serial Clock - ($\overline{\text{TRST}}$ /DSCLK)

The MTMOD[3:0] signals determine the function of this dual-purpose pin. If MTMOD[3:0]=0000, the DSCLK function is selected. If MTMOD[3:0]=0001, the TRST function is selected. MTMOD[3:0] should not be changed while $\overline{\text{RSTI}} = 1$. When used as $\overline{\text{TRST}}$, this pin will asynchronously reset the internal JTAG controller to the test logic reset state, causing the JTAG instruction register to choose the “idcode” command. When this occurs, all the JTAG logic is benign and will not interfere with the normal functionality of the MCF5307 processor. Although this signal is asynchronous, Motorola recommends that $\overline{\text{TRST}}$ make only a 0 to 1 (asserted to negated) transition while TMS is held at a logic 1 value. $\overline{\text{TRST}}$ has an internal pullup so that if it is not driven low its value will default to a logic level of 1. However, if $\overline{\text{TRST}}$ will not be used, it can either be tied to ground or, if TCK is clocked, it can be tied to VDD. The former connection will place the JTAG controller in the test logic reset state immediately, while the later connection will cause the JTAG controller (if TMS is a logic 1) to eventually end up in the test logic reset state after 5 clocks of TCK.

This pin is also used as the development serial clock (DSCLK) for the serial interface to the Debug Module. The maximum frequency for the DSCLK signal is 1/2 the BCLKO frequency. See the **Debug Support** section for additional information on this signal.

17.2.3 Test Mode Select/ Breakpoint (TMS/ $\overline{\text{BKPT}}$)

The MTMOD[3:0] signals determine this pin's dual function. If MTMOD[3:0]=0000, the $\overline{\text{BKPT}}$ function is selected. If MTMOD[3:0]=0001, then the TMS function is selected. MTMOD[3:0] should not change while $\overline{\text{RSTI}} = 1$. When used as TMS, this input signal provides the JTAG controller with information to determine which test operation mode should be performed. The value of TMS and current state 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 pullup so that if it is not driven low, its value will default to a logic level of 1. However, if TMS will not be used, it should be tied to VDD. This pin also signals a hardware breakpoint to the processor when in the debug mode. See the **Debug Support** section for additional information on this signal.

17.2.4 Test Data Input/Development Serial Input - (TDI/DSI)

This is a dual-function pin. If $MTMOD[3:0] = 0000$, then DSI is selected. If $MTMOD[3:0] = 0001$, then TDI is selected. When used as TDI, this input signal provides the serial data port for loading the various JTAG shift registers composed of the boundary scan register, the bypass register, and the instruction register. Shifting in of data depends on the state of the JTAG controller state machine and the instruction currently in the instruction register. This data shift occurs on the rising edge of TCK. TDI also has an internal pullup so that if it is not driven low its value will default to a logic level of 1. However, if TDI will not be used, it should be tied to VDD.

This pin also provides the single-bit communication for the debug module commands. See the **Debug Support** section for additional information on this signal.

17.2.5 Test Data Output/Development Serial Output - (TDO/DSO)

This is a dual-function pin. When $MTMOD[3:0] = 0000$, then DSO is selected. When $MTMOD[3:0] = 0001$, TDO is selected. When used as TDO, this output signal provides the serial data port for outputting data from the JTAG logic. Shifting out of data depends on the state of the JTAG controller state machine and the instruction currently in the instruction register. This data shift occurs on the falling edge of TCK. When TDO is not outputting test data, it is three-stated. TDO can also be placed in three-state mode to allow bussed or parallel connections to other devices having JTAG. This signal also provides single-bit communication for the debug module responses. See the **Debug Support** section for additional information on this signal.

17.3 JTAG REGISTER DESCRIPTIONS

17.3.1 JTAG Instruction Shift Register

The MCF5307 IEEE 1149.1 Standard implementation uses a 3-bit instruction-shift register without parity. This register transfers its value to a parallel hold register and applies one of six possible instructions on the falling edge of TCK when the TAP state machine is in the update-IR state. To load the instructions into the shift portion of the register, place the serial data on the TDI pin prior to 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 the TDO pin.

Table 17-2 lists the public customer-usable instructions that are supported along with their encoding.

Table 17-2. JTAG Instructions

INSTRUCTION	ABBR	CLASS	IR[2:0]	INSTRUCTION SUMMARY
EXTEST	EXT	Required	000	Select BS register while applying fixed values to output pins and asserting functional reset
IDCODE	IDC	Optional	001	Selects IDCODE register for shift
SAMPLE/ PRELOAD	SMP	Required	100	Selects BS register for shift, sample, and preload without disturbing functional operation
HIGHZ	HIZ	Optional	101	Selects the bypass register while three-stating all output pins and asserting functional reset
CLAMP	CMP	Optional	110	Selects bypass while applying fixed values to output pins and asserting functional reset
BYPASS	BYP	Required	111	Selects the bypass register for data operations

The IEEE 1149.1 Standard 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.

17.3.1.1 EXTEST INSTRUCTION. The external test instruction (EXTEST) selects the boundary-scan register. The EXTEST instruction 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. The EXTEST instruction can also configure the direction of bidirectional pins and establish high-impedance states on some pins. The EXTEST instruction 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.

17.3.1.2 IDCODE. The IDCODE instruction selects the 32-bit IDcode register for connection as a shift path between the TDI pin and the TDO pin. This instruction lets you interrogate the MCF5307 to determine its version number and other part identification data. The IDcode register has been implemented in accordance with IEEE 1149.1 so that the least significant bit 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 to be shifted out after selecting the IDcode register is always a logic 1. The remaining 31-bits are also set to fixed values (see **17.3.2 IDcode Register**) on the rising edge of TCK following entry into the capture-DR state.

The IDCODE instruction is the default value placed in the instruction register when a JTAG reset is accomplished 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 will cause the TAP state machine to enter the test-logic-reset state (normal operation of the TAP state machine into the test-logic-reset state will also result in placing 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 when in the Capture-IR state and a rising edge of TCK occurs.

17.3.1.3 SAMPLE/PRELOAD INSTRUCTION. The SAMPLE/PRELOAD instruction provides two separate functions. First, it obtains a sample of the system data and control signals present at the MCF5307 input pins and just prior to 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 resident in the instruction register. You can observe this sampled data by shifting it through the boundary-scan register to the output TDO by using the shift-DR state. Both the data capture and the shift operation are transparent to system operation. You are responsible for providing some form of external synchronization to achieve meaningful results because there is no internal synchronization between TCK and the system clock, CLK.

The second function of the SAMPLE/PRELOAD instruction is to initialize the boundary scan register update cells before selecting EXTEST or CLAMP. This is achieved by ignoring the data being shifted out of the TDO pin 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 will be applied to the external output pins when one of the instructions listed above is applied.

17.3.1.4 HIGHZ INSTRUCTION. The HIGHZ instruction anticipates the need to backdrive the output pins and protect the input pins from random toggling during circuit board testing. The HIGHZ instruction selects the bypass register, forcing all output and bidirectional pins to the high-impedance state.

The HIGHZ instruction goes 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 5.

17.3.1.5 CLAMP INSTRUCTION. The CLAMP instruction selects the bypass register and asserts functional reset while simultaneously forcing all output pins and bidirectional pins configured as outputs to the fixed values that are preloaded and held in the boundary-scan update registers. This instruction enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary-scan register. The CLAMP instruction 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 6.

17.3.1.6 BYPASS INSTRUCTION. The BYPASS instruction selects the single-bit bypass register, creating a single-bit shift register path from the TDI pin to the bypass register to the TDO pin. This instruction enhances test efficiency by reducing the overall shift path when a device other than the MCF5307 processor becomes the device under test on a board design with multiple chips on the overall 1149.1 defined boundary-scan chain. The bypass register has been implemented in accordance with 1149.1 so that the shift register stage is set to logic zero on the rising edge of TCK following entry into the capture-DR state. Therefore, the first bit to be shifted out after selecting the bypass register is always a logic zero (to differentiate a part that supports an IDCODE register from a part that supports only the bypass register). The BYPASS instruction goes 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 7.

17.3.2 IDcode Register

An IEEE 1149.1 compliant JTAG identification register has been included on the MCF5307. The MCF5307 JTAG instruction encoded as octal 1 provides for reading the JTAG IDcode register.

ID code Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VERSION NO				0	1	0	0	1	1	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	0	0	0	0	0	0	1	1	1	0	1

Bits 31-28 Version Number

Indicates the revision number of the MCF5307.

Bits 27-22 Design Center

Indicates the ColdFire design center.

Bits 21-12 Device Number

Indicates an MCF5307.

Bits 11-1 JEDEC ID

Indicates the reduced JEDEC ID for Motorola (JEDEC refers to the Joint Electron Device Engineering Council. Refer to JEDEC publication 106-A and chapter 11 of the IEEE 1149.1 Standard for further information on this field).

Bit 0

Differentiates this register as the JTAG IDcode register (as opposed to the bypass register) according to the IEEE 1149.1 Standard.

17.3.3 JTAG BOUNDARY-SCAN REGISTER

The MCF5307 model includes an IEEE 1149.1-compliant boundary-scan register. The boundary-scan register is connected between TDI and TDO when the EXTEST or SAMPLE/PRELOAD instructions are selected. This register captures signal pin data on the input pins, forces fixed values on the output signal pins, and selects the direction and

drive characteristics (a logic value or high impedance) of the bidirectional and three-state signal pins.

Table 17-3. Boundary-Scan Bit Definitions

BIT	CELL TYPE	PINCELL NAME	PIN TYPE
0	O.Ctl	PP(0) enable	-
1	O.Pin	PP(0)	I/O
2	I.Pin	PP(0)	I/O
3	IO.Ctl	PP(1) enable	-
4	O.Pin	PP(1)	I/O
5	I.Pin	PP(1)	I/O
6	IO.Ctl	PP(2) enable	-
7	O.Pin	PP(2)	I/O
8	I.Pin	PP(2)	I/O
9	IO.Ctl	PP(3) enable	-
10	O.Pin	PP(3)	I/O
11	I.Pin	PP(3)	I/O
12	IO.Ctl	PP(4) enable	-
13	O.Pin	PP(4)	I/O
14	I.Pin	PP(4)	I/O
15	IO.Ctl	PP(5) enable	-
16	O.Pin	PP(5)	I/O
17	I.Pin	PP(5)	I/O
18	IO.Ctl	PP(6) enable	-
19	O.Pin	PP(6)	I/O
20	I.Pin	PP(6)	I/O
21	IO.Ctl	PP(7) enable	-
22	O.Pin	PP(7)	I/O
23	I.Pin	PP(7)	I/O
24	O.Pin	PST(3)	O
25	O.Pin	PST(2)	O
26	O.Pin	PST(1)	O
27	O.Pin	PST(0)	O
28	O.Pin	DDATA(3)	O
29	O.Pin	DDATA(2)	O
30	O.Pin	DDATA(1)	O
31	O.Pin	DDATA(0)	O
32	O.Pin	PSTCLK	O
33	I.Pin	CLKIN	I
34	IO.Ctl	XRSTO enable	-
35	O.Pin	XRSTO	I/O
36	I.Pin	XRSTO	I/O
37	O.Pin	BCLKO	O
38	I.Pin	EDGESEL	I
39	O.Pin	TXD1	O
40	I.Pin	RXD1	I
41	O.Pin	RTS1	O
42	I.Pin	CTS1	I
43	O.Pin	TXD2	O

Table 17-3. Boundary-Scan Bit Definitions (Continued)

BIT	CELL TYPE	PINCELL NAME	PIN TYPE
44	I.Pin	RXD2	I
45	O.Pin	RTS2	O
46	I.Pin	CTS2	I
47	I.Pin	XHIZ	I
48	IO.Ctl	DATA enable	-
49	O.Pin	DATA(0)	I/O
50	I.Pin	DATA(0)	I/O
51	O.Pin	DATA(1)	I/O
52	I.Pin	DATA(1)	I/O
53	O.Pin	DATA(2)	I/O
54	I.Pin	DATA(2)	I/O
55	O.Pin	DATA(3)	I/O
56	I.Pin	DATA(3)	I/O
57	O.Pin	DATA(4)	I/O
58	I.Pin	DATA(4)	I/O
59	O.Pin	DATA(5)	I/O
60	I.Pin	DATA(5)	I/O
61	O.Pin	DATA(6)	I/O
62	I.Pin	DATA(6)	I/O
63	O.Pin	DATA(7)	I/O
64	I.Pin	DATA(7)	I/O
65	O.Pin	DATA(8)	I/O
66	I.Pin	DATA(8)	I/O
67	O.Pin	DATA(9)	I/O
68	I.Pin	DATA(9)	I/O
69	O.Pin	DATA(10)	I/O
70	I.Pin	DATA(10)	I/O
71	O.Pin	DATA(11)	I/O
72	I.Pin	DATA(11)	I/O
73	O.Pin	DATA(12)	I/O
74	I.Pin	DATA(12)	I/O
75	O.Pin	DATA(13)	I/O
76	I.Pin	DATA(13)	I/O
77	O.Pin	DATA(14)	I/O
78	I.Pin	DATA(14)	I/O
79	O.Pin	DATA(15)	I/O
80	I.Pin	DATA(15)	I/O
81	O.Pin	DATA(16)	I/O
82	I.Pin	DATA(16)	I/O
83	O.Pin	DATA(17)	I/O
84	I.Pin	DATA(17)	I/O
85	O.Pin	DATA(18)	I/O
86	I.Pin	DATA(18)	I/O
87	O.Pin	DATA(19)	I/O
88	I.Pin	DATA(19)	I/O
89	O.Pin	DATA(20)	I/O
90	I.Pin	DATA(20)	I/O

Table 17-3. Boundary-Scan Bit Definitions (Continued)

BIT	CELL TYPE	PINCELL NAME	PIN TYPE
91	O.Pin	DATA(21)	I/O
92	I.Pin	DATA(21)	I/O
93	O.Pin	DATA(22)	I/O
94	I.Pin	DATA(22)	I/O
95	O.Pin	DATA(23)	I/O
96	I.Pin	DATA(23)	I/O
97	O.Pin	DATA(24)	I/O
98	I.Pin	DATA(24)	I/O
99	O.Pin	DATA(25)	I/O
100	I.Pin	DATA(25)	I/O
101	O.Pin	DATA(26)	I/O
102	I.Pin	DATA(26)	I/O
103	O.Pin	DATA(27)	I/O
104	I.Pin	DATA(27)	I/O
105	O.Pin	DATA(28)	I/O
106	I.Pin	DATA(28)	I/O
107	O.Pin	DATA(29)	I/O
108	I.Pin	DATA(29)	I/O
109	O.Pin	DATA(30)	I/O
110	I.Pin	DATA(30)	I/O
111	O.Pin	DATA(31)	I/O
112	I.Pin	DATA(31)	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	XBE(3)	O
118	O.Pin	XBE(2)	O
119	O.Pin	XBE(1)	O
120	O.Pin	XBE(0)	O
121	O.Pin	SCKE	O
122	O.Pin	SCAS	O
123	O.Pin	SRAS	O
124	O.Pin	XDRAMW	O
125	O.Pin	XCAS(3)	O
126	O.Pin	XCAS(2)	O
127	O.Pin	XCAS(1)	O
128	O.Pin	XCAS(0)	O
129	O.Pin	XRAS(1)	O
130	O.Pin	XRAS(0)	O
131	I.Pin	TIN1	I
132	I.Pin	TIN0	I
133	O.Pin	TOUT0	O
134	O.Pin	TOUT1	O
135	I.Pin	XBG	I
136	O.Pin	XBD	O
137	O.Pin	XBR	O

Table 17-3. Boundary-Scan Bit Definitions (Continued)

BIT	CELL TYPE	PINCELL NAME	PIN TYPE
138	I.Pin	XIRQ1	I
139	I.Pin	XIRQ3	I
140	I.Pin	XIRQ5	I
141	I.Pin	XIRQ7	I
142	I.Pin	XRSTI	I
143	O.Pin	XTS	I/O
144	I.Pin	XTS	I/O
145	IO.Ctl	XTA enable	-
146	O.Pin	XTA	I/O
147	I.Pin	XTA	I/O
148	O.Pin	RW	I/O
149	I.Pin	RW	I/O
150	O.Pin	XAS	I/O
151	I.Pin	XAS	I/O
152	O.Pin	XCS(7)	O
153	O.Pin	XCS(6)	O
154	O.Pin	XCS(5)	O
155	O.Pin	XCS(4)	O
156	O.Pin	XCS(3)	O
157	O.Pin	XCS(2)	O
158	O.Pin	XCS(1)	O
159	O.Pin	XCS(0)	O
160	O.Pin	XOE	O
161	O.Pin	SIZ(1)	I/O
162	I.Pin	SIZ(1)	I/O
163	O.Pin	SIZ(0)	I/O
164	I.Pin	SIZ(0)	I/O
165	IO.Ctl	PP(15) enable	-
166	I.Pin	PP(15)	I/O
167	O.Pin	PP(15)	I/O
168	IO.Ctl	PP(14) enable	-
169	I.Pin	PP(14)	I/O
170	O.Pin	PP(14)	I/O
171	IO.Ctl	PP(13) enable	-
172	I.Pin	PP(13)	I/O
173	O.Pin	PP(13)	I/O
174	IO.Ctl	PP(12) enable	-
175	I.Pin	PP(12)	I/O
176	O.Pin	PP(12)	I/O
177	IO.Ctl	PP(11) enable	-
178	I.Pin	PP(11)	I/O
179	O.Pin	PP(11)	I/O
180	IO.Ctl	PP(10) enable	-
181	I.Pin	PP(10)	I/O
182	O.Pin	PP(10)	I/O
183	IO.Ctl	PP(9) enable	-
184	I.Pin	PP(9)	I/O

Table 17-3. Boundary-Scan Bit Definitions (Continued)

BIT	CELL TYPE	PINCELL NAME	PIN TYPE
185	O.Pin	PP(9)	I/O
186	IO.Ctl	PP(8) enable	-
187	I.Pin	PP(8)	I/O
188	O.Pin	PP(8)	I/O
189	IO.Ctl	XTS / RW / SIZ enable	-
190	IO.Ctl	ADDR enable	-
191	O.Pin	ADDR(23)	I/O
192	I.Pin	ADDR(23)	I/O
193	O.Pin	ADDR(22)	I/O
194	I.Pin	ADDR(22)	I/O
195	O.Pin	ADDR(21)	I/O
196	I.Pin	ADDR(21)	I/O
197	O.Pin	ADDR(20)	I/O
198	I.Pin	ADDR(20)	I/O
199	O.Pin	ADDR(19)	I/O
200	I.Pin	ADDR(19)	I/O
201	O.Pin	ADDR(18)	I/O
202	I.Pin	ADDR(18)	I/O
203	O.Pin	ADDR(17)	I/O
204	I.Pin	ADDR(17)	I/O
205	O.Pin	ADDR(16)	I/O
206	I.Pin	ADDR(16)	I/O
207	O.Pin	ADDR(15)	I/O
208	I.Pin	ADDR(15)	I/O
209	O.Pin	ADDR(14)	I/O
210	I.Pin	ADDR(14)	I/O
211	O.Pin	ADDR(13)	I/O
212	I.Pin	ADDR(13)	I/O
213	O.Pin	ADDR(12)	I/O
214	I.Pin	ADDR(12)	I/O
215	O.Pin	ADDR(11)	I/O
216	I.Pin	ADDR(11)	I/O
217	O.Pin	ADDR(10)	I/O
218	I.Pin	ADDR(10)	I/O
219	O.Pin	ADDR(9)	I/O
220	I.Pin	ADDR(9)	I/O
221	O.Pin	ADDR(8)	I/O
222	I.Pin	ADDR(8)	I/O
223	O.Pin	ADDR(7)	I/O
224	I.Pin	ADDR(7)	I/O
225	O.Pin	ADDR(6)	I/O
226	I.Pin	ADDR(6)	I/O
227	O.Pin	ADDR(5)	I/O
228	I.Pin	ADDR(5)	I/O
229	O.Pin	ADDR(4)	I/O
230	I.Pin	ADDR(4)	I/O
231	O.Pin	ADDR(3)	I/O

Table 17-3. Boundary-Scan Bit Definitions (Continued)

BIT	CELL TYPE	PINCELL NAME	PIN TYPE
232	I.Pin	ADDR(3)	I/O
233	O.Pin	ADDR(2)	I/O
234	I.Pin	ADDR(2)	I/O
235	O.Pin	ADDR(1)	I/O
236	I.Pin	ADDR(1)	I/O
237	O.Pin	ADDR(0)	I/O
238	I.Pin	ADDR(0)	I/O

17.3.4 JTAG BYPASS REGISTER

The MCF5307 includes an IEEE 1149.1-compliant bypass register, which creates a single bit shift register path from TDI to the bypass register to TDO when the BYPASS instruction is selected.

17.4 TAP CONTROLLER

The value of TMS at the rising edge of TCK determines the current state of the TAP controller. There are basically two paths that the TAP controller can follow: The first, for executing JTAG instructions; the second, for manipulating JTAG data based on the JTAG instructions. The various states of the TAP controller are shown in Figure 17-2. For more detail on each state, refer to the IEEE 1149.1 Standard JTAG document. Do note, though, that from any state the TAP controller is in, Test-Logic-Reset can be entered if TMS is held high for at least five rising edges of TCK.

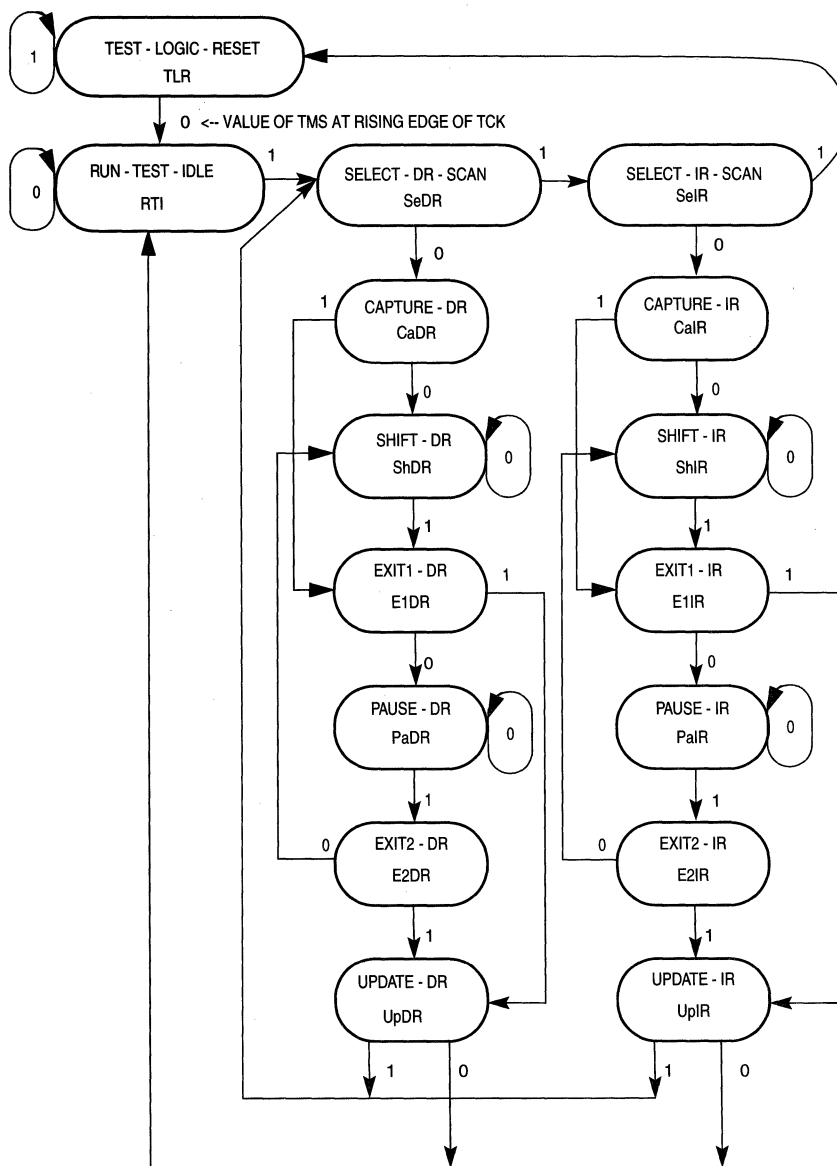


Figure 17-2. JTAG TAP Controller State Machine

17.5 RESTRICTIONS

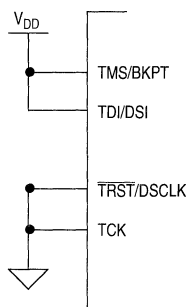
The test logic is implemented using static logic design, and TCK can be stopped in either a high or low state without loss of data. The system logic, however, operates on a different

system clock which is not synchronized to TCK internally. Any mixed operation requiring the use of 1149.1 test logic in conjunction with system functional logic that uses both clocks, must have coordination and synchronization of these clocks done externally to the MCF5307.

17.6 DISABLING THE IEEE 1149.1 STANDARD OPERATION

There are two methods by which the MCF5307 can be used without the IEEE 1149.1 test logic being active: 1) Nonuse of the JTAG test logic by either nontermination (disconnection) or intentional fixing of TAP logic values, and 2) Intentional disabling of the JTAG test logic by **NOT setting MTMOD[3:0]= 0001** (entering Debug mode).

There are several considerations that must be addressed if the IEEE 1149.1 logic is not going to be used once the MCF5307 is assembled onto a board. The prime consideration is to ensure that the IEEE 1149.1 test logic remains transparent and benign to the system logic during functional operation. This requires the minimum of either connecting the $\overline{\text{TRST}}$ pin to logic 0, or connecting the TCK clock pin to a clock source that will supply five rising edges and the falling edge after the fifth rising edge, to ensure that the part enters the test-logic-reset state. The recommended solution is to connect $\overline{\text{TRST}}$ to logic 0. Another consideration is that the TCK pin does not have an internal pullup as is required on the TMS, TDI, and $\overline{\text{TRST}}$ pins; therefore, it should not be left unterminated to preclude mid-level input values. Figure 17-3 shows pin values recommended for disabling JTAG with the MCF5307 in JTAG mode.

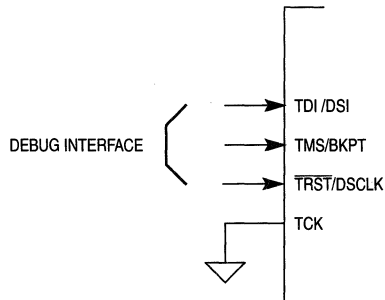


NOTE: MTMOD[3:0] SET TO '0001' ALLOWS JTAG MODE.

Figure 17-3. Disabling JTAG in JTAG Mode

A second method of using the MCF5307 without the IEEE 1149.1 logic being active is to select Debug mode by **setting MTMOD[3:0]= 0000**. The IEEE 1149.1 test controller is now placed in the test-logic-reset state by the internal assertion of the $\overline{\text{TRST}}$ signal to the controller and the TAP pins function as Debug mode pins. While in JTAG mode, input pins

TDI/DSI, TMS/BKPT, and TRST/DSCLK have internal pullups enabled. Figure 17-4 shows pin values recommended for disabling JTAG with the MCF5307 in Debug mode.



NOTE: MTMOD[3:0] NOT SET TO '0001' PROHIBITS JTAG.

Figure 17-4. Disabling JTAG in Debug Mode

17.7 MCF5307 BSDL FILE

```
-- MOTOROLA SSDLT JTAG SOFTWARE
-- BSDL File Generated: Wed Jul 2 18:00:53 1997
--
-- Revision History:
--
```

```
entity MCF5307 is
generic (PHYSICAL_PIN_MAP : string := "TQFP_208");
```

```
port ( BKPT:inbit;
       DSI:inbit;
       DSO:outbit;
       DSCLK:inbit;
       TCK:inbit;
       ADDR:inoutbit_vector(0 to 23);
       SIZ:inoutbit_vector(0 to 1);
       XOE:bufferbit;
       XCS:bufferbit_vector(0 to 7);
       XAS:inoutbit;
       RW:inoutbit;
       XTA:inoutbit;
       XTS:inoutbit;
       XRSTI:inbit;
       XBR:bufferbit;
       XBD:bufferbit;
       XBG:inbit;
       XIRQ7:inbit;
       XIRQ5:inbit;
       XIRQ3:inbit;
       XIRQ1:inbit;
       TOUT1:bufferbit;
```

```

TOUT0:bufferbit;
TIN0:inbit;
TIN1:inbit;
XNAS:bufferbit_vector(0 to 1);
XCAS:bufferbit_vector(0 to 3);
XDRAMW:bufferbit;
SRAS:bufferbit;
SCAS:bufferbit;
SCKE:bufferbit;
  XBE:bufferbit_vector(0 to 3);
  SCL:inoutbit;
  SDA:inoutbit;
DATA:inoutbit_vector(0 to 31);
XHIZ:inbit;
MTMOD:linkagebit_vector(0 to 3);
CTS2:inbit;
RTS2:bufferbit;
RXD2:inbit;
TXD2:bufferbit;
CTS1:inbit;
RTS1:bufferbit;
RXD1:inbit;
TXD1:bufferbit;
EDGESEL:inbit;
BCKO:bufferbit;
XRSTO:inoutbit;
CLKIN:inbit;
PSTCLK:bufferbit;
DDATA:bufferbit_vector(0 to 3);
  PST:bufferbit_vector(0 to 3);
  PP:inoutbit_vector(0 to 15);
  VDD:linkagebit_vector(0 to 27);
  GND:linkagebit_vector(0 to 28);
  PVDD:linkagebit_vector(0 to 1);
  PGND:linkagebit_vector(0 to 1);
  PLLTPA:linkagebit);

use STD_1149_1_1994.all;

attribute COMPONENT_CONFORMANCE of MCF5307 : entity is "STD_1149_1_1993";

attribute PIN_MAP of MCF5307 : entity is PHYSICAL_PIN_MAP;

constant TQFP_208 : PIN_MAP_STRING :=
  "VDD:      (1, 7, 13, 21, 29, 37, 45, 52, 57, 65, 73, 81, 89, 97, 105,
113, 121, 129, 137,  " &
"145, 152, 157, 165, 173, 177, 191, 197, 205), " &
  "ADDR:      (2, 3, 5, 6, 8, 9, 11, 12, 14, 15, 16, 18, 19, 20, 22, 23,
24, 26, 27, 28, 30,  " &
"31, 32, 34), " &
  "GND:      (4, 10, 17, 25, 33, 41, 48, 53, 61, 69, 77, 85, 93, 101,
104, 109, 117, 125,  " &
"133, 141, 148, 156, 161, 169, 175, 188, 194, 201, 208), " &

```

IEEE 1149.1 TEST ACCESS PORT (JTAG)

```
"PP:          (207, 206, 204, 203, 202, 200, 199, 198, 35, 36, 38, 39, 40,
42, 43, 44), " &
"SIZ:        (46, 47), " &
"XOE:        49, " &
"XCS:        (50, 51, 54, 55, 56, 58, 59, 60), " &
"XAS:        62, " &
"RW:         63, " &
"XTA:        64, " &
"XTS:        66, " &
"XRSTI:      67, " &
"XIRQ7:      68, " &
"XIRQ5:      70, " &
"XIRQ3:      71, " &
"XIRQ1:      72, " &
"XBR:        74, " &
"XBD:        75, " &
"XBG:        76, " &
"TOUT1:      78, " &
"TOUT0:      79, " &
"TTN1:       80, " &
"TTN0:       82, " &
"XRAS:       (83, 84), " &
"XCAS:       (86, 87, 88, 90), " &
"XDRAMW:     91, " &
"SRAS:       92, " &
"SCAS:       94, " &
"SCKE:       95, " &
"XBE:        (96, 98, 99, 100), " &
"SCL:        102, " &
"SDA:        103, " &
"DATA:       (147, 146, 144, 143, 142, 140, 139, 138, 136, 135, 134, 132,
131, 130, 128, 127, " &
"126, 124, 123, 122, 120, 119, 118, 116, 115, 114, 112, 111, 110, 108, 107, 106),
" &
"DSCLK:      149, " &
"TCK:        150, " &
"DSO:        151, " &
"DSI:        153, " &
"BKPT:       154, " &
"XHIZ:       155, " &
"MTMOD:      (158, 159, 160, 162), " &
"CTS2:       163, " &
"RTS2:       164, " &
"RXD2:       166, " &
"TXD2:       167, " &
"CTS1:       168, " &
"RTS1:       170, " &
"RXD1:       171, " &
"TXD1:       172, " &
"EDGESEL:    174, " &
"BCLKO:      176, " &
"XRSTO:      178, " &
"PVDD:       (179, 185), " &
"CLKIN:      180, " &
```

```

"PGND:      (181, 183), " &
"PLLTPA:    182, " &
"PSTCLK:    184, " &
"DDATA:     (186, 187, 189, 190), " &
"PST:       (192, 193, 195, 196) ";

attribute TAP_SCAN_IN    of DSI : signal is true;
attribute TAP_SCAN_OUT  of DSO : signal is true;
attribute TAP_SCAN_MODE of BKPT : signal is true;
attribute TAP_SCAN_RESET of DSCLK : signal is true;
attribute TAP_SCAN_CLOCK of TCK : signal is (20.0e6, BOTH);

attribute INSTRUCTION_LENGTH of MCF5307 : entity is 3;

attribute INSTRUCTION_OPCODE of MCF5307 : entity is
"EXTEST (000)," &
"SAMPLE (100)," &
"IDCODE (001)," &
"CLAMP (110)," &
"HIGHZ (101)," &
"BYPASS (111)";

attribute INSTRUCTION_CAPTURE of MCF5307 : entity is "001";
attribute IDCODE_REGISTER    of MCF5307 : entity is
"0000"          & -- version
"010011"       & -- manufacturer's use
"0000000011"   & -- sequence number
"00000001110"  & -- manufacturer identity
"1";          -- 1149.1 requirement

attribute BOUNDARY_LENGTH of MCF5307 : entity is 239;

attribute BOUNDARY_REGISTER of MCF5307 : entity is
-- num  cell  port  func  safe [ccell  dis  rslt]
"0  (BC_2, *,      controlr,  1)," &
"1  (BC_2, PP(0),  output3,   X,    0,  1,  Z)," &
"2  (BC_4, PP(0),  input,     X)," &
"3  (BC_2, *,      controlr,  1)," &
"4  (BC_2, PP(1),  output3,   X,    3,  1,  Z)," &
"5  (BC_4, PP(1),  input,     X)," &
"6  (BC_2, *,      controlr,  1)," &
"7  (BC_2, PP(2),  output3,   X,    6,  1,  Z)," &
"8  (BC_4, PP(2),  input,     X)," &
"9  (BC_2, *,      controlr,  1)," &
"10 (BC_2, PP(3),  output3,   X,    9,  1,  Z)," &
"11 (BC_4, PP(3),  input,     X)," &
"12 (BC_2, *,      controlr,  1)," &
"13 (BC_2, PP(4),  output3,   X,   12,  1,  Z)," &
"14 (BC_4, PP(4),  input,     X)," &
"15 (BC_2, *,      controlr,  1)," &
"16 (BC_2, PP(5),  output3,   X,   15,  1,  Z)," &
"17 (BC_4, PP(5),  input,     X)," &
"18 (BC_2, *,      controlr,  1)," &
"19 (BC_2, PP(6),  output3,   X,   18,  1,  Z)," &

```

IEEE 1149.1 TEST ACCESS PORT (JTAG)

```

-- num    cell    port    func                safe [ccell dis rslt]
"20      (BC_4, PP(6),    input,                X)," &
"21      (BC_2, *,        controlr,              1)," &
"22      (BC_2, PP(7),    output3,               X,    21,    1,    Z)," &
"23      (BC_4, PP(7),    input,                X)," &
"24      (BC_2, PST(3),    output2,               X)," &
"25      (BC_2, PST(2),    output2,               X)," &
"26      (BC_2, PST(1),    output2,               X)," &
"27      (BC_2, PST(0),    output2,               X)," &
"28      (BC_2, DDATA(3), output2,               X)," &
"29      (BC_2, DDATA(2), output2,               X)," &
"30      (BC_2, DDATA(1), output2,               X)," &
"31      (BC_2, DDATA(0), output2,               X)," &
"32      (BC_2, PSTCLK,    output2,               X)," &
"33      (BC_4, CLKIN,     input,                X)," &
"34      (BC_2, *,        controlr,              1)," &
"35      (BC_2, XRSTO,     output3,               X,    34,    1,    Z)," &
"36      (BC_4, XRSTO,     input,                X)," &
"37      (BC_2, BCLKO,     output2,               X)," &
"38      (BC_4, EDGESEL,   input,                X)," &
"39      (BC_2, TXD1,      output2,               X)," &
-- num    cell    port    func                safe [ccell dis rslt]
"40      (BC_4, RXD1,      input,                X)," &
"41      (BC_2, RTS1,      output2,               X)," &
"42      (BC_4, CTS1,      input,                X)," &
"43      (BC_2, TXD2,      output2,               X)," &
"44      (BC_4, RXD2,      input,                X)," &
"45      (BC_2, RTS2,      output2,               X)," &
"46      (BC_4, CTS2,      input,                X)," &
"47      (BC_4, XHIZ,      input,                X)," &
"48      (BC_2, *,        controlr,              1)," &
"49      (BC_2, DATA(0),  output3,               X,    48,    1,    Z)," &
"50      (BC_4, DATA(0),  input,                X)," &
"51      (BC_2, DATA(1),  output3,               X,    48,    1,    Z)," &
"52      (BC_4, DATA(1),  input,                X)," &
"53      (BC_2, DATA(2),  output3,               X,    48,    1,    Z)," &
"54      (BC_4, DATA(2),  input,                X)," &
"55      (BC_2, DATA(3),  output3,               X,    48,    1,    Z)," &
"56      (BC_4, DATA(3),  input,                X)," &
"57      (BC_2, DATA(4),  output3,               X,    48,    1,    Z)," &
"58      (BC_4, DATA(4),  input,                X)," &
"59      (BC_2, DATA(5),  output3,               X,    48,    1,    Z)," &
-- num    cell    port    func                safe [ccell dis rslt]
"60      (BC_4, DATA(5),  input,                X)," &
"61      (BC_2, DATA(6),  output3,               X,    48,    1,    Z)," &
"62      (BC_4, DATA(6),  input,                X)," &
"63      (BC_2, DATA(7),  output3,               X,    48,    1,    Z)," &
"64      (BC_4, DATA(7),  input,                X)," &
"65      (BC_2, DATA(8),  output3,               X,    48,    1,    Z)," &
"66      (BC_4, DATA(8),  input,                X)," &
"67      (BC_2, DATA(9),  output3,               X,    48,    1,    Z)," &
"68      (BC_4, DATA(9),  input,                X)," &
"69      (BC_2, DATA(10), output3,               X,    48,    1,    Z)," &
"70      (BC_4, DATA(10), input,                X)," &

```

```

"71 (BC_2, DATA(11), output3, X, 48, 1, Z)," &
"72 (BC_4, DATA(11), input, X)," &
"73 (BC_2, DATA(12), output3, X, 48, 1, Z)," &
"74 (BC_4, DATA(12), input, X)," &
"75 (BC_2, DATA(13), output3, X, 48, 1, Z)," &
"76 (BC_4, DATA(13), input, X)," &
"77 (BC_2, DATA(14), output3, X, 48, 1, Z)," &
"78 (BC_4, DATA(14), input, X)," &
"79 (BC_2, DATA(15), output3, X, 48, 1, Z)," &
-- num cell port func safe [ccell dis rslt]
"80 (BC_4, DATA(15), input, X)," &
"81 (BC_2, DATA(16), output3, X, 48, 1, Z)," &
"82 (BC_4, DATA(16), input, X)," &
"83 (BC_2, DATA(17), output3, X, 48, 1, Z)," &
"84 (BC_4, DATA(17), input, X)," &
"85 (BC_2, DATA(18), output3, X, 48, 1, Z)," &
"86 (BC_4, DATA(18), input, X)," &
"87 (BC_2, DATA(19), output3, X, 48, 1, Z)," &
"88 (BC_4, DATA(19), input, X)," &
"89 (BC_2, DATA(20), output3, X, 48, 1, Z)," &
"90 (BC_4, DATA(20), input, X)," &
"91 (BC_2, DATA(21), output3, X, 48, 1, Z)," &
"92 (BC_4, DATA(21), input, X)," &
"93 (BC_2, DATA(22), output3, X, 48, 1, Z)," &
"94 (BC_4, DATA(22), input, X)," &
"95 (BC_2, DATA(23), output3, X, 48, 1, Z)," &
"96 (BC_4, DATA(23), input, X)," &
"97 (BC_2, DATA(24), output3, X, 48, 1, Z)," &
"98 (BC_4, DATA(24), input, X)," &
"99 (BC_2, DATA(25), output3, X, 48, 1, Z)," &
-- num cell port func safe [ccell dis rslt]
"100 (BC_4, DATA(25), input, X)," &
"101 (BC_2, DATA(26), output3, X, 48, 1, Z)," &
"102 (BC_4, DATA(26), input, X)," &
"103 (BC_2, DATA(27), output3, X, 48, 1, Z)," &
"104 (BC_4, DATA(27), input, X)," &
"105 (BC_2, DATA(28), output3, X, 48, 1, Z)," &
"106 (BC_4, DATA(28), input, X)," &
"107 (BC_2, DATA(29), output3, X, 48, 1, Z)," &
"108 (BC_4, DATA(29), input, X)," &
"109 (BC_2, DATA(30), output3, X, 48, 1, Z)," &
"110 (BC_4, DATA(30), input, X)," &
"111 (BC_2, DATA(31), output3, X, 48, 1, Z)," &
"112 (BC_4, DATA(31), input, X)," &
"113 (BC_2, SDA, output2, 1, 113, 1, Weak1)," &
"114 (BC_4, SDA, input, X)," &
"115 (BC_2, SCL, output2, 1, 115, 1, Weak1)," &
"116 (BC_4, SCL, input, X)," &
"117 (BC_2, XBE(3), output2, X)," &
"118 (BC_2, XBE(2), output2, X)," &
"119 (BC_2, XBE(1), output2, X)," &
-- num cell port func safe [ccell dis rslt]
"120 (BC_2, XBE(0), output2, X)," &
"121 (BC_2, SCKE, output2, X)," &

```

```

"122 (BC_2, SCAS, output2, X)," &
"123 (BC_2, SRAS, output2, X)," &
"124 (BC_2, XDRAMW, output2, X)," &
"125 (BC_2, XCAS(3), output2, X)," &
"126 (BC_2, XCAS(2), output2, X)," &
"127 (BC_2, XCAS(1), output2, X)," &
"128 (BC_2, XCAS(0), output2, X)," &
"129 (BC_2, XRAS(1), output2, X)," &
"130 (BC_2, XRAS(0), output2, X)," &
"131 (BC_4, TIN1, input, X)," &
"132 (BC_4, TIN0, input, X)," &
"133 (BC_2, TOUT0, output2, X)," &
"134 (BC_2, TOUT1, output2, X)," &
"135 (BC_4, XBG, input, X)," &
"136 (BC_2, XBD, output2, X)," &
"137 (BC_2, XBR, output2, X)," &
"138 (BC_4, XIRQ1, input, X)," &
"139 (BC_4, XIRQ3, input, X)," &
-- num cell port func safe [ccell dis rslt]
"140 (BC_4, XIRQ5, input, X)," &
"141 (BC_4, XIRQ7, input, X)," &
"142 (BC_4, XRSTI, input, X)," &
"143 (BC_2, XTS, output3, X, 189, 1, Z)," &
"144 (BC_4, XTS, input, X)," &
"145 (BC_2, *, controlr, 1)," &
"146 (BC_2, XTA, output3, X, 145, 1, Z)," &
"147 (BC_4, XTA, input, X)," &
"148 (BC_2, RW, output3, X, 189, 1, Z)," &
"149 (BC_4, RW, input, X)," &
"150 (BC_2, XAS, output3, X, 189, 1, Z)," &
"151 (BC_4, XAS, input, X)," &
"152 (BC_2, XCS(7), output2, X)," &
"153 (BC_2, XCS(6), output2, X)," &
"154 (BC_2, XCS(5), output2, X)," &
"155 (BC_2, XCS(4), output2, X)," &
"156 (BC_2, XCS(3), output2, X)," &
"157 (BC_2, XCS(2), output2, X)," &
"158 (BC_2, XCS(1), output2, X)," &
"159 (BC_2, XCS(0), output2, X)," &
-- num cell port func safe [ccell dis rslt]
"160 (BC_2, XOE, output2, X)," &
"161 (BC_2, SIZ(1), output3, X, 189, 1, Z)," &
"162 (BC_4, SIZ(1), input, X)," &
"163 (BC_2, SIZ(0), output3, X, 189, 1, Z)," &
"164 (BC_4, SIZ(0), input, X)," &
"165 (BC_2, *, controlr, 1)," &
"166 (BC_4, PP(15), input, X)," &
"167 (BC_2, PP(15), output3, X, 165, 1, Z)," &
"168 (BC_2, *, controlr, 1)," &
"169 (BC_4, PP(14), input, X)," &
"170 (BC_2, PP(14), output3, X, 168, 1, Z)," &
"171 (BC_2, *, controlr, 1)," &
"172 (BC_4, PP(13), input, X)," &
"173 (BC_2, PP(13), output3, X, 171, 1, Z)," &

```

```

"174 (BC_2, *, controlr, 1)," &
"175 (BC_4, PP(12), input, X)," &
"176 (BC_2, PP(12), output3, X, 174, 1, Z)," &
"177 (BC_2, *, controlr, 1)," &
"178 (BC_4, PP(11), input, X)," &
"179 (BC_2, PP(11), output3, X, 177, 1, Z)," &
-- num cell port func safe [ccell dis rslt]
"180 (BC_2, *, controlr, 1)," &
"181 (BC_4, PP(10), input, X)," &
"182 (BC_2, PP(10), output3, X, 180, 1, Z)," &
"183 (BC_2, *, controlr, 1)," &
"184 (BC_4, PP(9), input, X)," &
"185 (BC_2, PP(9), output3, X, 183, 1, Z)," &
"186 (BC_2, *, controlr, 1)," &
"187 (BC_4, PP(8), input, X)," &
"188 (BC_2, PP(8), output3, X, 186, 1, Z)," &
"189 (BC_2, *, controlr, 1)," &
"190 (BC_2, *, controlr, 1)," &
"191 (BC_2, ADDR(23), output3, X, 190, 1, Z)," &
"192 (BC_4, ADDR(23), input, X)," &
"193 (BC_2, ADDR(22), output3, X, 190, 1, Z)," &
"194 (BC_4, ADDR(22), input, X)," &
"195 (BC_2, ADDR(21), output3, X, 190, 1, Z)," &
"196 (BC_4, ADDR(21), input, X)," &
"197 (BC_2, ADDR(20), output3, X, 190, 1, Z)," &
"198 (BC_4, ADDR(20), input, X)," &
"199 (BC_2, ADDR(19), output3, X, 190, 1, Z)," &
-- num cell port func safe [ccell dis rslt]
"200 (BC_4, ADDR(19), input, X)," &
"201 (BC_2, ADDR(18), output3, X, 190, 1, Z)," &
"202 (BC_4, ADDR(18), input, X)," &
"203 (BC_2, ADDR(17), output3, X, 190, 1, Z)," &
"204 (BC_4, ADDR(17), input, X)," &
"205 (BC_2, ADDR(16), output3, X, 190, 1, Z)," &
"206 (BC_4, ADDR(16), input, X)," &
"207 (BC_2, ADDR(15), output3, X, 190, 1, Z)," &
"208 (BC_4, ADDR(15), input, X)," &
"209 (BC_2, ADDR(14), output3, X, 190, 1, Z)," &
"210 (BC_4, ADDR(14), input, X)," &
"211 (BC_2, ADDR(13), output3, X, 190, 1, Z)," &
"212 (BC_4, ADDR(13), input, X)," &
"213 (BC_2, ADDR(12), output3, X, 190, 1, Z)," &
"214 (BC_4, ADDR(12), input, X)," &
"215 (BC_2, ADDR(11), output3, X, 190, 1, Z)," &
"216 (BC_4, ADDR(11), input, X)," &
"217 (BC_2, ADDR(10), output3, X, 190, 1, Z)," &
"218 (BC_4, ADDR(10), input, X)," &
"219 (BC_2, ADDR(9), output3, X, 190, 1, Z)," &
-- num cell port func safe [ccell dis rslt]
"220 (BC_4, ADDR(9), input, X)," &
"221 (BC_2, ADDR(8), output3, X, 190, 1, Z)," &
"222 (BC_4, ADDR(8), input, X)," &
"223 (BC_2, ADDR(7), output3, X, 190, 1, Z)," &
"224 (BC_4, ADDR(7), input, X)," &

```


IEEE 1149.1 TEST ACCESS PORT (JTAG)

```
"225 (BC_2, ADDR(6), output3, X, 190, 1, Z), " &
"226 (BC_4, ADDR(6), input, X), " &
"227 (BC_2, ADDR(5), output3, X, 190, 1, Z), " &
"228 (BC_4, ADDR(5), input, X), " &
"229 (BC_2, ADDR(4), output3, X, 190, 1, Z), " &
"230 (BC_4, ADDR(4), input, X), " &
"231 (BC_2, ADDR(3), output3, X, 190, 1, Z), " &
"232 (BC_4, ADDR(3), input, X), " &
"233 (BC_2, ADDR(2), output3, X, 190, 1, Z), " &
"234 (BC_4, ADDR(2), input, X), " &
"235 (BC_2, ADDR(1), output3, X, 190, 1, Z), " &
"236 (BC_4, ADDR(1), input, X), " &
"237 (BC_2, ADDR(0), output3, X, 190, 1, Z), " &
"238 (BC_4, ADDR(0), input, X);
```

end MCF5307;

17.8 OBTAINING THE IEEE 1149.1 STANDARD

The IEEE 1149 Standard 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

SECTION 18

ELECTRICAL SPECIFICATIONS

Table 18-1. 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	°C

Table 18-2. Operating Temperature

CHARACTERISTIC	SYMBOL	VALUE	UNITS
Maximum Operating Junction Temperature	T_j	TBD	°C
Maximum Operating Ambient temperature	T_{Amax}	70 ¹	°C
Minimum Operating Ambient Temperature	T_{Amin}	0	°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.

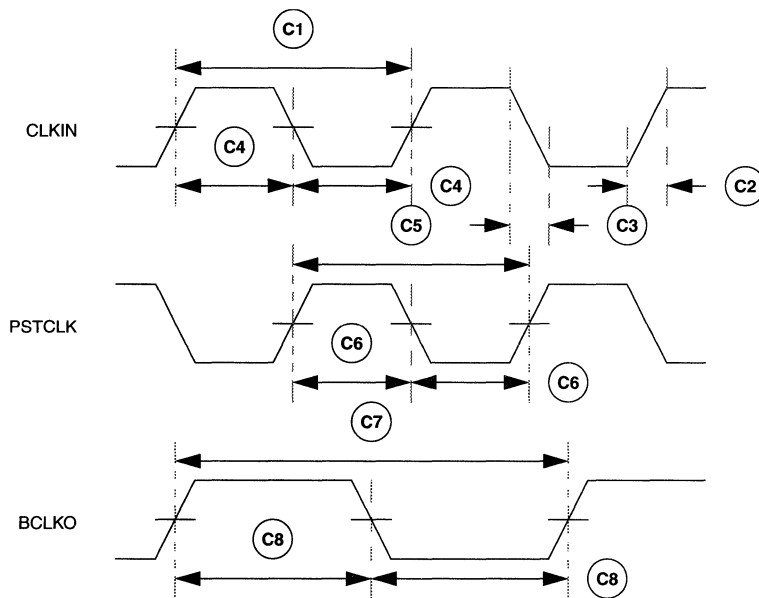
Table 18-3. DC Electrical Specifications ($V_{CC} = 3.3 \text{ Vdc} \pm 0.3 \text{ Vdc}$)

CHARACTERISTIC	SYMBOL	MIN	MAX	UNITS
Operation Voltage Range	V_{CC}	3.0	3.6	V
Input High Voltage	V_{IH}	2	5.5	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
Hi-Impedance (Three-State) Leakage Current @ 0.5/2.4 V During Normal Operation	I_{TSI}	-	20	μA
Signal Low Input Current, $V_{IL} = 0.8 \text{ V}^3$	I_{IL}	0	1	mA
Signal High Input Current, $V_{IH} = 2.0 \text{ V}^3$	I_{IH}	0	1	mA
Output High Voltage $I_{OH} = 8\text{mA}^1, 16\text{mA}^2$	V_{OH}	2.4	-	V
Output Low Voltage $I_{OL} = 8\text{mA}^1, 16\text{mA}^2$	V_{OL}	-	0.5	V
Load Capacitance (All Outputs)	C_L	-	50	pF
Capacitance ⁴ , $V_{in} = 0 \text{ V}$, $f = 1 \text{ MHz}$	C_{IN}	-	TBD	pF

1. DATA[31:0], ADDR[23:0], PP[15:0], TS, TA, SIZE[1:0], R/W, BR, BD, RSTO, AS, CS[7:0], BE[3:0], OE, PSTCLK, PST[3:0], DDATA[3:0], DSO, TOUT[1:0], SCL, SDA, RTS[2:1], TXD[2:1]
2. BCLKO, RAS[1:0], CAS[3:0], DRAMW, SCKE, SRAS, SCAS
3. BKPT/TMS, DSI/TDI, DSCLK/TRST
4. Capacitance C_{IN} is periodically sampled rather than 100% tested.

Table 18-4. Clock Timing Specification

NUM	CHARACTERISTIC	66 MHZ		90 MHZ		UNITS
		MIN	MAX	MIN	MAX	
C1	CLKIN cycle time	30	-	22	-	nsec
C2	CLKIN rise time (.5V to 2.4 V)	-	5	-	5	nsec
C3	CLKIN fall time (2.4V to 0.5 V)	-	5	-	5	nsec
C4	CLKIN duty cycle (at 1.5 V)	40	60	40	60	%
C5	PSTCLK cycle time	15	-	11	-	nsec
C6	PSTCLK duty cycle (at 1.5 V)	40	60	40	60	%
C7	BCLKO cycle time	30	-	22	-	nsec
C8	BCLKO duty cycle (at 1.5 V)	45	55	45	55	%

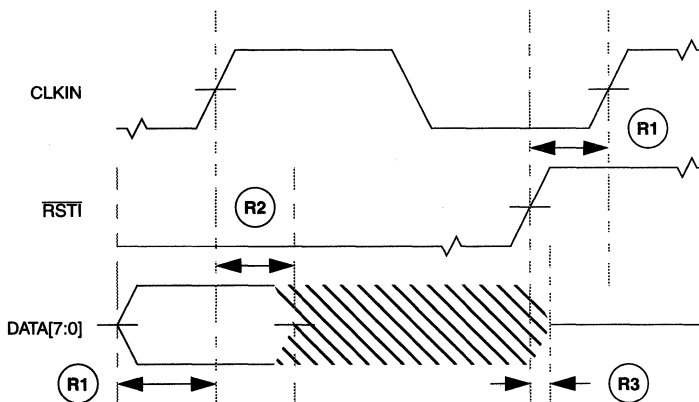


Note: Signals above are shown in relation to the clock. No relationship between signals is implied or intended.

Table 18-5. Reset Timing Specification

NUM	CHARACTERISTIC	66 MHZ		90 MHZ		UNITS
		MIN	MAX	MIN	MAX	
R1 ¹	Valid to CLKIN (setup)	7.5	-	5.5	-	nsec
R2	CLKIN to Invalid (hold)	3	-	2	-	nsec
R3	$\overline{\text{RSTI}}$ to invalid (hold)	3	-	2	-	nsec

$\overline{\text{RSTI}}$ and Data[7:0] are internally synchronized. This setup time must be met only if recognition on a particular clock is required.



Note: Mode Selects are registered on the previous rising CLKIN edge before the cycle in which $\overline{\text{RSTI}}$ is recognized as being negated.

Table 18-6. Input AC Timing Specification

NUM	CHARACTERISTIC	66 MHZ		90 MHZ		UNITS
		MIN	MAX	MIN	MAX	
B1 ^{1,3}	Valid to BCLKO Rising (setup)	7.5	-	5.5	-	nsec
B2 ¹	BCLKO Rising to Invalid (hold)	3	-	2	-	nsec
B3 ^{2,3}	Valid to BCLKO Falling (setup)	7.5	-	5.5	-	nsec
B4 ²	BCLKO Falling to Invalid (hold)	3	-	2	-	nsec
B5 ¹	BCLKO to Input High Impedance	-	15	-	11	nsec
B6	EDGESEL Valid to BCLKO	7.5	-	5.5	-	nsec
B7	BCLKO to EDGESEL Invalid	0	-	0	-	nsec

- Inputs (rising): \overline{BG} , \overline{TA} , ADDR[23:0], PP[15:0], SIZE[1:0], R/W, \overline{TS} , EDGESEL, DATA[31:0], \overline{IRQ} [7], \overline{IRQ} [5], \overline{IRQ} [3], \overline{IRQ} [1], BKPT
- Inputs (falling): \overline{AS} , EDGESEL
- \overline{IRQ} [7], \overline{IRQ} [5], \overline{IRQ} [3], \overline{IRQ} [1], BKPT, and \overline{AS} are internally synchronized. This setup time must be met only if recognition on a particular clock is required.

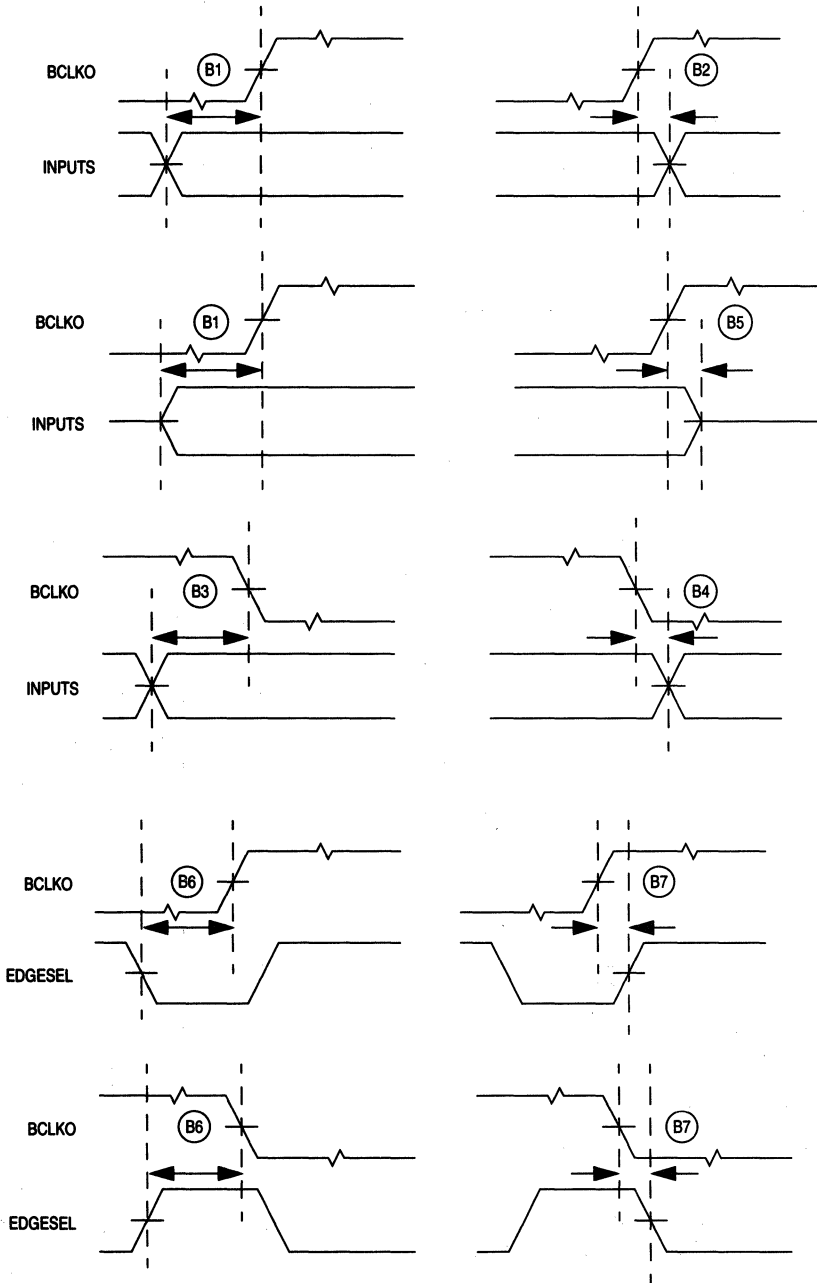
Table 18-7. Output AC Timing Specification

NUM	CHARACTERISTIC	66 MHZ		90 MHZ		UNITS
		MIN	MAX	MIN	MAX	
B10 ¹	BCLKO Rising to Valid	-	15	-	11	nsec
B11 ¹	BCLKO Rising to Invalid (hold)	3	-	2	-	nsec
B12 ⁴	BCLKO to High Impedance (Three-State)	-	15	-	11	nsec
B13 ²	BCLKO Falling to Valid	-	15	-	11	nsec
B14 ²	BCLKO Falling to Invalid (hold)	3	-	2	-	nsec
B15 ³	EDGESEL to Valid	-	18.5	-	13.5	nsec
B16 ³	EDGESEL to Invalid (hold)	3	-	2	-	nsec
H1	\overline{HIZ} to High Impedance	-	60	-	60	nsec
H2	\overline{HIZ} to Low Impedance	-	60	-	60	nsec

- Outputs (rising): \overline{RSTO} , DATA[31:0], \overline{TS} , BR, BD, ADDR[23:0], PP[15:0], R/W, SIZE[1:0], TA, \overline{RAS} [1:0], CAS[3:0], DRAMW, SCKE, SRAS, SCAS
- Outputs (falling): DATA[31:0], ADDR[23:0], PP[15:8], \overline{AS} , \overline{CS} [7:0], BE[3:0], \overline{OE} , \overline{RAS} [1:0], \overline{CAS} [3:0], DRAMW, SCKE, SRAS, SCAS
- Edgesel: DATA[31:0], ADDR[23:0], PP[15:8], \overline{RAS} [1:0], \overline{CAS} [3:0], DRAMW, SCKE, SRAS, SCAS

ELECTRICAL SPECIFICATIONS

4. High Impedance (Three-State): DATA[31:0], ADDR[23:0], PP[15:0], R/W, SIZE[1:0], \overline{TS} , \overline{AS} , \overline{TA}



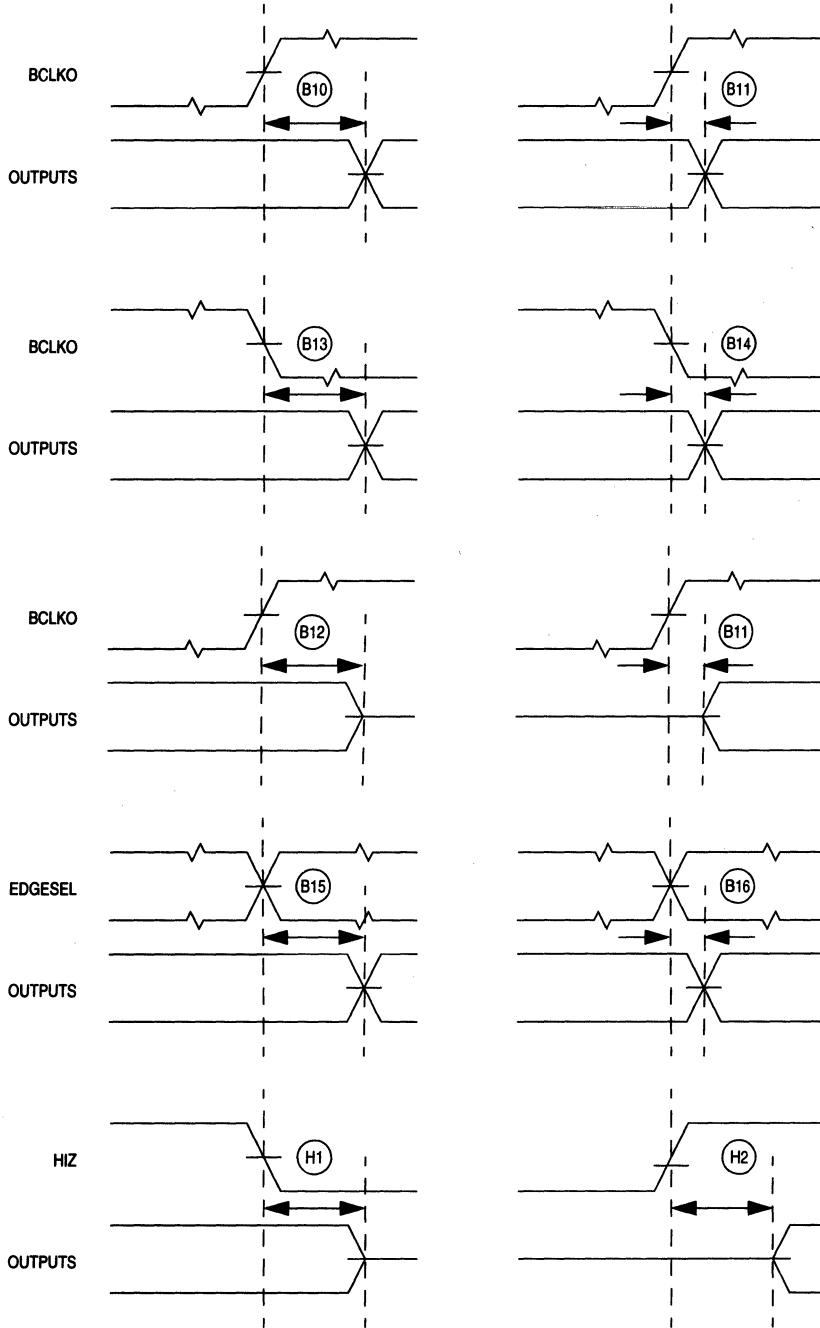


Table 18-8.Debug AC Timing Specification

NUM	CHARACTERISTIC	66 MHZ		90 MHZ		UNITS
		MIN	MAX	MIN	MAX	
D1	PSTCLK to Valid (Output valid)	-	7.5	-	5.5	nsec
D2	PSTCLK to Invalid (Output hold)	3	-	2	-	nsec
D3 ¹	Valid to PSTCLK (Input setup)	7.5	-	5.5	-	nsec
D4	PSTCLK to Invalid (Input hold)	3	-	2	-	nsec

1. DSCLK and DSI are internally synchronized. This setup time must be met only if recognition on a particular clock is required.

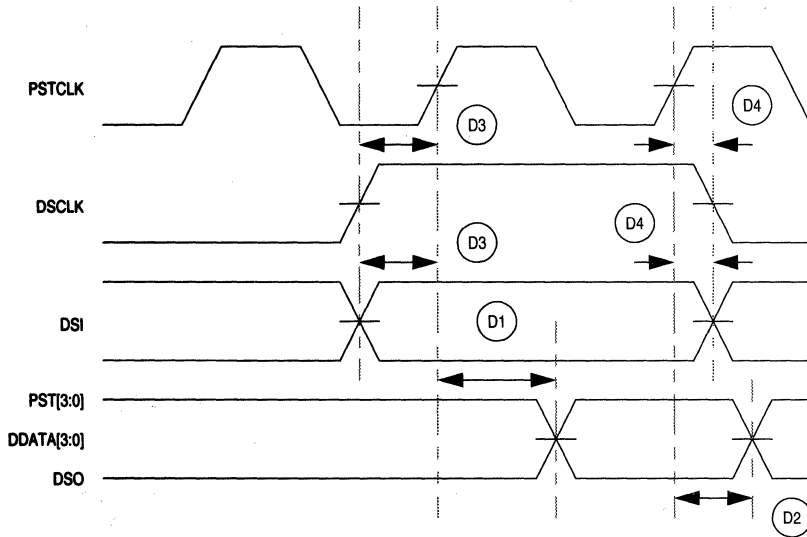


Table 18-9.Timer Module AC Timing Specification

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	-	nsec
T3	BCLKO to TIN Invalid (input hold)	3	-	2	-	nsec
T4	BCLKO to TOUT Valid (output valid)	-	15	-	11	nsec
T5	BCLKO to TOUT Invalid (output hold)	3	-	2	-	nsec
T6	TIN Pulse Width	1	-	1	-	bus clocks
T7	TOUT Pulse Width	1	-	1	-	bus clocks

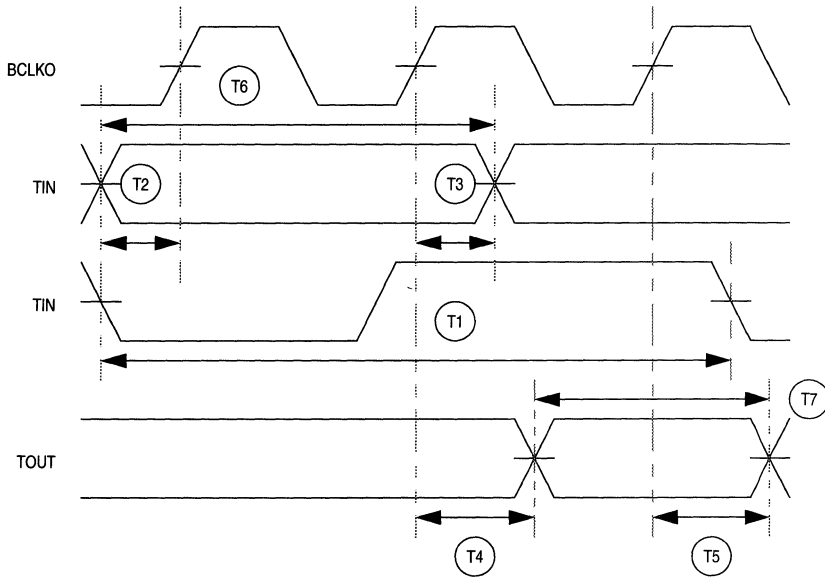


Table 18-10. UART Module AC Timing Specifications

NUM	CHARACTERISTIC	66 MHZ		90 MHZ		UNITS
		MIN	MAX	MIN	MAX	
U1	RXD Valid to BCLKO (input setup)	7.5	-	5.5	-	nsec
U2	BCLKO to RXD Invalid (input hold)	3	-	2	-	nsec
U3	$\overline{\text{CTS}}$ Valid to BCLKO (input setup)	7.5	-	5.5	-	nsec
U4	BCLKO to $\overline{\text{CTS}}$ Invalid (input hold)	3	-	2	-	nsec
U5	BCLKO to TXD Valid (output valid)	-	15	-	11	nsec
U6	BCLKO to TXD Invalid (output hold)	3	-	2	-	nsec
U7	BCLKO to $\overline{\text{RTS}}$ Valid (output valid)	-	15	-	11	nsec
U8	BCLKO to $\overline{\text{RTS}}$ Invalid (output hold)	3	-	2	-	nsec

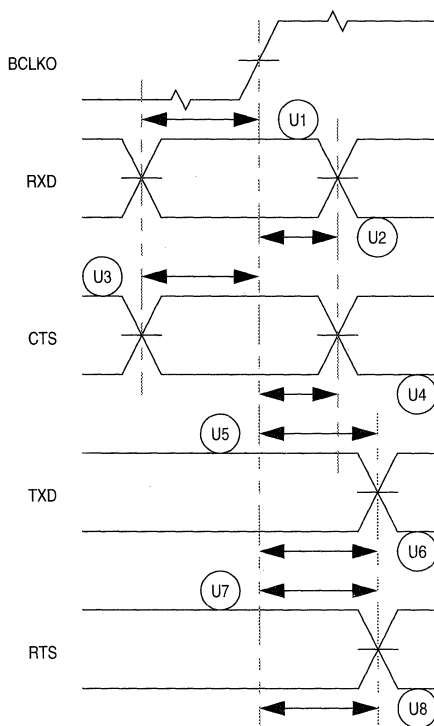


Table 18-11. M-Bus Input Timing Specifications Between SCL and SDA

NUM	CHARACTERISTIC	66 MHZ		90 MHZ		UNITS
		MIN	MAX	MIN	MAX	
M1	Start Condition Hold Time	2	-	2	-	bus clocks
M2	Clock Low Period	8	-	8	-	bus clocks
M3	SCL/SDA Rise Time (V _{IL} = 0.5 V to V _{IH} = 2.4 V)	-	1	-	1	msec
M4	Data Hold Time	0	-	0	-	nsec
M5	SCL/SDA Fall Time (V _{IH} = 2.4 V to V _{IL} = 0.5 V)	-	1	-	1	msec
M6	Clock High time	4	-	4	-	bus clocks
M7	Data Setup Time	0	-	0	-	nsec
M8	Start Condition Setup Time (for repeated start condition only)	2	-	2	-	bus clocks
M9	Stop Condition Setup Time	2	-	2	-	bus clocks

Table 18-12. M-Bus Output Timing Specifications Between SCL and SDA

NUM	CHARACTERISTIC	66 MHZ		90 MHZ		UNITS
		MIN	MAX	MIN	MAX	
M1 ¹	Start Condition Hold Time	6	-	6	-	bus clocks
M2 ¹	Clock Low Period	10	-	10	-	bus clocks
M3 ²	SCL/SDA Rise Time (V _{IL} = 0.5 V to V _{IH} = 2.4 V)	-	-	-	-	msec
M4 ¹	Data Hold Time	7	-	7	-	bus clocks
M5 ³	SCL/SDA Fall Time (V _{IH} = 2.4 V to V _{IL} = 0.5 V)	-	TBD	-	TBD	nsec
M6 ¹	Clock High time	10	-	10	-	bus clocks
M7 ¹	Data Setup Time	2	-	2	-	bus clocks
M8 ¹	Start Condition Setup Time (for repeated start condition only)	20	-	20	-	bus clocks
M9 ¹	Stop Condition Setup Time	10	-	10	-	bus clocks

1. Note: Output numbers are dependent on the value programmed into the MFDR; an MFDR programmed with the maximum frequency (MFDR = 0x20) will result in minimum output timings as shown in the above table.

ELECTRICAL SPECIFICATIONS

The MBUS 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 MFDR; however, numbers given in the above table are the minimum values.

2. Since SCL and SDA are open-collector-type outputs, which the processor can only actively drive low, the time required for SCL or SDA to reach a high level depends on external signal capacitance and pull-up resistor values.
3. Specified at a nominal 50pF load

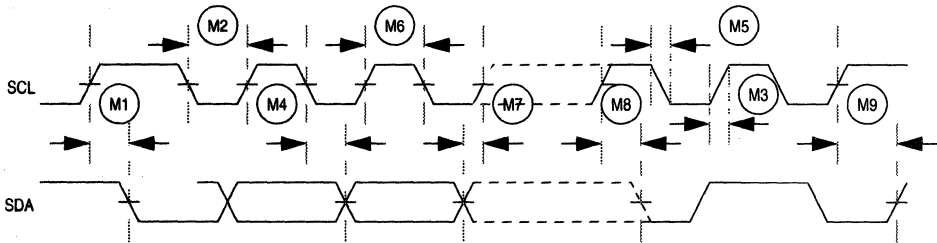


Table 18-13. M-Bus Timing Specifications Between BCLKO and SCL, SDA

NUM	CHARACTERISTIC	66 MHZ		90 MHZ		UNITS
		MIN	MAX	MIN	MAX	
M10 ³	SCL, SDA Valid to BCLKO (input setup)	15	-	11	-	nsec
M11	BCLKO to SCL, SDA Invalid (input hold)	3	-	2	-	nsec
M12 ¹	BCLKO to SCL, SDA Low (output valid)	-	15	-	11	nsec
M13 ²	BCLKO to SCL, SDA Invalid (output hold)	3	-	2	-	nsec

1. Since SCL and SDA are open-collector-type outputs, which the processor can only actively drive low, this specification applies only when SCL or SDA are driven low by the processor. The time required for SCL or SDA to reach a high level depends on external signal capacitance and pull-up resistor values.
2. Since SCL and SDA are open-collector-type outputs, which the processor can only actively drive low, this specification applies only when SCL or SDA are actively being driven or held low by the processor.
3. SCL and SDA are internally synchronized. This setup time must be met only if recognition on a particular clock is required.

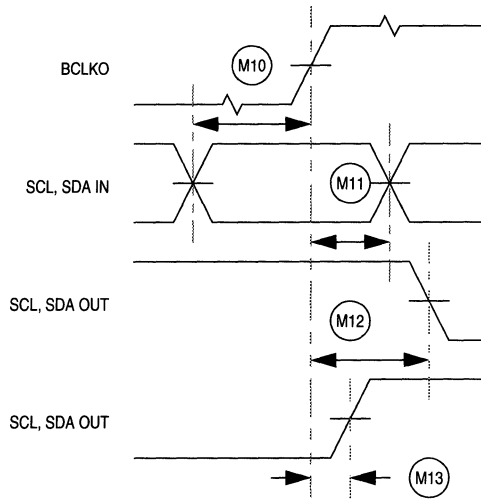


Table 18-14. General-Purpose I/O Port AC Timing Specifications

NUM	CHARACTERISTIC	66 MHZ		90 MHZ		UNITS
		MIN	MAX	MIN	MAX	
P1	PP Valid to BCLKO (input setup)	7.5	-	5.5	-	nsec
P2	BCLKO to PP Invalid (input hold)	3	-	2	-	nsec
P3	BCLKO to PP Valid (output valid)	-	15	-	11	nsec
P4	BCLKO to PP Invalid (output hold)	3	-	2	-	nsec

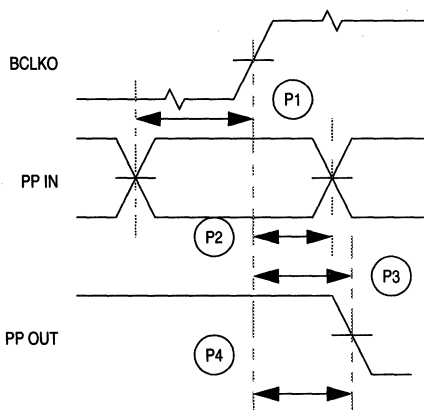
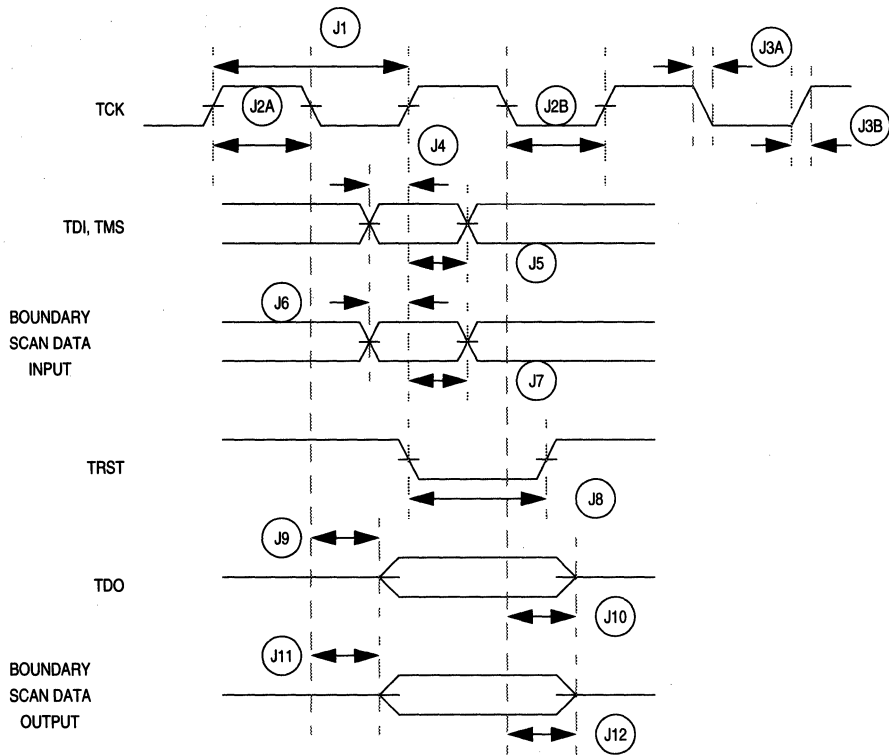


Table 18-15. IEEE 1149.1 (JTAG) AC Timing Specifications

NUM	CHARACTERISTIC	66 MHZ		90 MHZ		UNITS
		MIN	MAX	MIN	MAX	
-	TCK Frequency of Operation	0	10	0	10	MHz
J1	TCK Cycle Time	100	-	100	-	nsec
J2a	TCK Clock Pulse High Width measured at 1.5 V	40	-	40	-	nsec
J2b	TCK Clock Pulse Low Width measured at 1.5 V	40	-	40	-	nsec
J3a	TCK Fall Time ($V_{IH}=2.4$ V to $V_{IL}=0.5$ V)	-	5	-	5	nsec
J3b	TCK Rise Time ($V_{IL}=0.5$ v to $V_{IH}=2.4$ V)	-	5	-	5	nsec
J4	TDI, TMS to TCK rising (Input Setup)	10	-	10	-	nsec
J5	TCK rising to TDI, TMS Invalid (Hold)	15	-	15	-	nsec
J6	Boundary Scan Data Valid to TCK (Setup)	10	-	10	-	nsec
J7	TCK to Boundary Scan Data Invalid (Hold)	15	-	15	-	nsec
J8	TRST Pulse Width (asynchronous to clock edges)	15	-	15	-	nsec
J9	TCK falling to TDO Valid (signal from driven or three-state)	-	30	-	30	nsec
J10	TCK falling to TDO High Impedance	-	30	-	30	nsec
J11	TCK falling to Boundary Scan Data Valid (signal from driven or three-state)	-	30	-	30	nsec
J12	TCK falling to Boundary Scan Data High Impedance	-	30	-	30	nsec



SECTION 19

MECHANICAL DATA

Visit this URL (<http://www.mot.com/SPS/HPESD/prod/coldfire/5307UM.html>) to obtain information on the mechanical characteristics of the MCF5307 integrated microprocessor.

19.1 PACKAGE

The MCF5307 package is a 208-pin, thermally enhanced QFP. Thermal characteristics are not yet included in this documentation.

19.2 PINOUT

The MCF5307 pinout is detailed in the following tables. The table consists of pin number, primary and secondary pin functional names, direction (I/O), and a brief description of the pin. 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.

19.3 PIN OUT TABLES

Table 19-1. Pin-Out Table (Left, Top to Bottom)

	PIN NAME	FUNC PRIMARY	FUNC 2ND	FUNC DIR	DESCRIPTION	DRIVE (MA)	SYNC	NEG
1	VCC	VDD	-	-	-	-	-	-
2	A0	ADDR[0]	-	I/O	Address Bus bit	8	-	SDRAM
3	A1	ADDR[1]	-	I/O	Address Bus bit	8	-	SDRAM
4	GND	GND	-	-	-	-	-	-
5	A2	ADDR[2]	-	I/O	Address Bus bit	8	-	SDRAM
6	A3	ADDR[3]	-	I/O	Address Bus bit	8	-	SDRAM
7	VCC	VDD	-	-	-	-	-	-
8	A4	ADDR[4]	-	I/O	Address Bus bit	8	-	SDRAM
9	A5	ADDR[5]	-	I/O	Address Bus bit	8	-	SDRAM
10	GND	GND	-	-	-	-	-	-
11	A6	ADDR[6]	-	I/O	Address Bus bit	8	-	SDRAM
12	A7	ADDR[7]	-	I/O	Address Bus bit	8	-	SDRAM
13	VCC	VDD	-	-	-	-	-	-
14	A8	ADDR[8]	-	I/O	Address Bus bit	8	-	SDRAM
15	A9	ADDR[9]	-	I/O	Address Bus bit	8	-	SDRAM
16	A10	ADDR[10]	-	I/O	Address Bus bit	8	-	SDRAM
17	GND	GND	-	-	-	-	-	-
18	A11	ADDR[11]	-	I/O	Address Bus bit	8	-	SDRAM
19	A12	ADDR[12]	-	I/O	Address Bus bit	8	-	SDRAM
20	A13	ADDR[13]	-	I/O	Address Bus bit	8	-	SDRAM
21	VCC	VDD	-	-	-	-	-	-
22	A14	ADDR[14]	-	I/O	Address Bus bit	8	-	SDRAM
23	A15	ADDR[15]	-	I/O	Address Bus bit	8	-	SDRAM
24	A16	ADDR[16]	-	I/O	Address Bus bit	8	-	SDRAM
25	GND	GND	-	-	-	-	-	-
26	A17	ADDR[17]	-	I/O	Address Bus bit	8	-	SDRAM
27	A18	ADDR[18]	-	I/O	Address Bus bit	8	-	SDRAM
28	A19	ADDR[19]	-	I/O	Address Bus bit	8	-	SDRAM
29	VCC	VDD	-	-	-	-	-	-
30	A20	ADDR[20]	-	I/O	Address Bus bit	8	-	SDRAM
31	A21	ADDR[21]	-	I/O	Address Bus bit	8	-	SDRAM
32	A22	ADDR[22]	-	I/O	Address Bus bit	8	-	SDRAM
33	GND	GND	-	-	-	-	-	-
34	A23	ADDR[23]	-	I/O	Address Bus bit	8	-	SDRAM
35	PP8	PP[8]	ADDR[24]	I/O	Parallel Port bit/Address Bus bit	8	-	SDRAM
36	PP9	PP[9]	ADDR[25]	I/O	Parallel Port bit/Address Bus bit	8	-	SDRAM
37	VCC	VDD	-	-	-	-	-	-
38	PP10	PP[10]	ADDR[26]	I/O	Parallel Port bit/Address Bus bit	8	-	SDRAM
39	PP11	PP[11]	ADDR[27]	I/O	Parallel Port bit/Address Bus bit	8	-	SDRAM
40	PP12	PP[12]	ADDR[28]	I/O	Parallel Port bit/Address Bus bit	8	-	SDRAM
41	GND	GND	-	-	-	-	-	-
42	PP13	PP[13]	ADDR[29]	I/O	Parallel Port bit/Address Bus bit	8	-	SDRAM
43	PP14	PP[14]	ADDR[30]	I/O	Parallel Port bit/Address Bus bit	8	-	SDRAM
44	PP15	PP[15]	ADDR[31]	I/O	Parallel Port bit/Address Bus bit	8	-	SDRAM

	PIN NAME	FUNC PRIMARY	FUNC 2ND	FUNC DIR	DESCRIPTION	DRIVE (MA)	SYNC	NEG
45	VCC	VDD	-	-	-	-	-	-
46	SIZ0	SIZ[0]	-	I/O	Size Attribute	8	-	-
47	SIZ1	SIZ[1]	-	I/O	Size Attribute	8	-	-
48	GND	GND	-	-	-	-	-	-
49	OE	XOE	-	O	Output Enable for Chip Selects	8	-	Yes
50	CS0	XCS[0]	-	O	Chip Select	8	-	Yes
51	CS1	XCS[1]	-	O	Chip Select	8	-	Yes
52	VCC	VDD	-	-	-	-	-	-

Table 19-2. Pin-Out Table (Bottom, Left to Right)

	PIN NAME	FUNC PRIMARY	FUNC 2ND	FUNC DIR	DESCRIPTION	DRIVE (MA)	SYNC	NEG
53	GND	GND	-	-	-	-	-	-
54	CS2	XCS[2]	-	O	Chip Select	8	-	Yes
55	CS3	XCS[3]	-	O	Chip Select	8	-	Yes
56	CS4	XCS[4]	-	O	Chip Select	8	-	Yes
57	VCC	VDD	-	-	-	-	-	-
58	CS5	XCS[5]	-	O	Chip Select	8	-	Yes
59	CS6	XCS[6]	-	O	Chip Select	8	-	Yes
60	CS7	XCS[7]	-	O	Chip Select	8	-	Yes
61	GND	GND	-	-	-	-	-	-
62	AS	XAS	-	I/O	Address Strobe	8	Yes	Yes
63	R/W	RW	-	I/O	Read / Write	8	-	-
64	TA	XTA	-	I/O	Transfer Acknowledge	8	-	-
65	VCC	VDD	-	-	-	-	-	-
66	TS	XTS	-	I/O	Transfer Start	8	-	-
67	RST1	XRST1	-	I	Reset	-	Yes	-
68	IRQ7	XIRQ[7]	-	I	Interrupt Request	-	Yes	-
69	GND	GND	-	-	-	-	-	-
70	IRQ5	XIRQ[5]	XIRQ[4]	I	Interrupt Request	-	Yes	-
71	IRQ3	XIRQ[3]	XIRQ[6]	I	Interrupt Request	-	Yes	-
72	IRQ1	XIRQ[1]	XIRQ[2]	I	Interrupt Request	-	Yes	-
73	VCC	VDD	-	-	-	-	-	-
74	BR	XBR	-	O	Bus Request	8	-	-
75	BD	XBD	-	O	Bus Driven	8	-	-
76	BG	XBG	-	I	Bus Grant	-	-	-
77	GND	GND	-	-	-	-	-	-
78	TOUT1	TOUT[1]	-	O	Timer Output	8	-	-
79	TOUT0	TOUT[0]	-	O	Timer Output	8	-	-
80	TIN0	TIN[0]	-	I	Timer Input	-	-	-
81	VCC	VDD	-	-	-	-	-	-
82	TIN1	TIN[1]	-	I	Timer Input	-	-	-
83	RAS0	XRAS[0]	-	O	DRAM Row Address Strobe	16	-	SDRAM
84	RAS1	XRAS[1]	-	O	DRAM Row Address Strobe	16	-	SDRAM
85	GND	GND	-	-	-	-	-	-
86	CAS0	XCAS[0]	-	O	DRAM Column Address Strobe	16	-	SDRAM
87	CAS1	XCAS[1]	-	O	DRAM Column Address Strobe	16	-	SDRAM
88	CAS2	XCAS[2]	-	O	DRAM Column Address Strobe	16	-	SDRAM
89	VCC	VDD	-	-	-	-	-	-
90	CAS3	XCAS[3]	-	O	DRAM Column Address Strobe	16	-	SDRAM
91	DRAMW	XDRAMW	-	O	DRAM Write	16	-	SDRAM
92	SRAS	XSRAS	-	O	SyncDRAM Row Address Strobe	16	-	SDRAM
93	GND	GND	-	-	-	-	-	-
94	SCAS	XSCAS	-	O	SyncDRAM Column Address Strobe	16	-	SDRAM
95	SCKE	SCKE	-	O	SyncDRAM Clock Enable	16	-	SDRAM
96	BE0	XBE[0]	XBWE[0]	O	Byte Enable / Byte Write Enable	8	-	-
97	VCC	VDD	-	-	-	-	-	-

	PIN NAME	FUNC PRIMARY	FUNC 2ND	FUNC DIR	DESCRIPTION	DRIVE (MA)	SYNC	NEG
98	BE1	XBE[1]	XBWE[1]	O	Byte Enable / Byte Write Enable	8	-	-
99	BE2	XBE[2]	XBWE[2]	O	Byte Enable / Byte Write Enable	8	-	-
100	BE3	XBE[3]	XBWE[3]	O	Byte Enable / Byte Write Enable	8	-	-
101	GND	GND	-	-	-	-	-	-
102	SCL	SCL	-	I/OD	Serial Clock Line	8	Yes	-
103	SDA	SDA	-	I/OD	Serial Data Line	8	Yes	-
104	GND	GND	-	-	-	-	-	-

OD - Open-Drain Output

Table 19-3. Pin-Out Table (Right, Bottom to Top)

	PIN NAME	FUNC PRIMARY	FUNC 2ND	FUNC DIR	DESCRIPTION	DRIVE (MA)	SYNC	NEG
105	VCC	VDD	-	-	-	-	-	-
106	DATA31	DATA[31]	-	I/O	Data Bus	8	-	SDRAM
107	DATA30	DATA[30]	-	I/O	Data Bus	8	-	SDRAM
108	DATA29	DATA[29]	-	I/O	Data Bus	8	-	SDRAM
109	GND	GND	-	-	-	-	-	-
110	DATA28	DATA[28]	-	I/O	Data Bus	8	-	SDRAM
111	DATA27	DATA[27]	-	I/O	Data Bus	8	-	SDRAM
112	DATA26	DATA[26]	-	I/O	Data Bus	8	-	SDRAM
113	VCC	VDD	-	-	-	-	-	-
114	DATA25	DATA[25]	-	I/O	Data Bus	8	-	SDRAM
115	DATA24	DATA[24]	-	I/O	Data Bus	8	-	SDRAM
116	DATA23	DATA[23]	-	I/O	Data Bus	8	-	SDRAM
117	GND	GND	-	-	-	-	-	-
118	DATA22	DATA[22]	-	I/O	Data Bus	8	-	SDRAM
119	DATA21	DATA[21]	-	I/O	Data Bus	8	-	SDRAM
120	DATA20	DATA[20]	-	I/O	Data Bus	8	-	SDRAM
121	VCC	VDD	-	-	-	-	-	-
122	DATA19	DATA[19]	-	I/O	Data Bus	8	-	SDRAM
123	DATA18	DATA[18]	-	I/O	Data Bus	8	-	SDRAM
124	DATA17	DATA[17]	-	I/O	Data Bus	8	-	SDRAM
125	GND	GND	-	-	-	-	-	-
126	DATA16	DATA[16]	-	I/O	Data Bus	8	-	SDRAM
127	DATA15	DATA[15]	-	I/O	Data Bus	8	-	SDRAM
128	DATA14	DATA[14]	-	I/O	Data Bus	8	-	SDRAM
129	VCC	VDD	-	-	-	-	-	-
130	DATA13	DATA[13]	-	I/O	Data Bus	8	-	SDRAM
131	DATA12	DATA[12]	-	I/O	Data Bus	8	-	SDRAM
132	DATA11	DATA[11]	-	I/O	Data Bus	8	-	SDRAM
133	GND	GND	-	-	-	-	-	-
134	DATA10	DATA[10]	-	I/O	Data Bus	8	-	SDRAM
135	DATA9	DATA[9]	-	I/O	Data Bus	8	-	SDRAM
136	DATA8	DATA[8]	-	I/O	Data Bus	8	-	SDRAM
137	VCC	VDD	-	-	-	-	-	-
138	DATA7	DATA[7]	CS_CONF[2]	I/O	Data Bus / Chip Select Config	8	Yes	SDRAM
139	DATA6	DATA[6]	CS_CONF[1]	I/O	Data Bus / Chip Select Config	8	Yes	SDRAM
140	DATA5	DATA[5]	CS_CONF[0]	I/O	Data Bus / Chip Select Config	8	Yes	SDRAM
141	GND	GND	-	-	-	-	-	-
142	DATA4	DATA[4]	ADDR_CONF	I/O	Data Bus / Addr Config	8	Yes	SDRAM
143	DATA3	DATA[3]	FREQ[1]	I/O	Data Bus / Freq Ctrl PLL	8	Yes	SDRAM
144	DATA2	DATA[2]	FREQ[0]	I/O	Data Bus / Freq Ctrl PLL	8	Yes	SDRAM
145	VCC	VDD	-	-	-	-	-	-
146	DATA1	DATA[1]	DIVIDE[1]	I/O	Data Bus / Divide Ctrl Pclk to Bclk	8	Yes	SDRAM
147	DATA0	DATA[0]	DIVIDE[0]	I/O	Data Bus / Divide Ctrl Pclk to Bclk	8	Yes	SDRAM
148	GND	GND	-	-	-	-	-	-
149	DSCLK	DSCLK	TRST	I	Debug Serial Clock / JTAG Reset	-	Yes	-
150	TCK	TCK	TCK	I	JTAG Clock	-	-	-

	PIN NAME	FUNC PRIMARY	FUNC 2ND	FUNC DIR	DESCRIPTION	DRIVE (MA)	SYNC	NEG
151	DSO	DSDO	TDO	O	Debug Serial Out/JTAG Data Out	8	-	-
152	VCC	VDD	-	-	-	-	-	-
153	DSI	DSDI	TDI	I	Debug Serial Input / JTAG Data In	-	Yes	-
154	BKPT	XBKPT	TMS	I	Debug Breakpoint /JTAG Mode Sel	-	Yes	-
155	HIZ	XHIZ	-	I	High Impedence Override	-	-	-
156	GND	GND	-	-	-	-	-	-

Table 19-4. Pin-Out Table (Top, Right -to- Left)

	PIN NAME	FUNC PRIMARY	FUNC SECONDARY	FUNC DIR	DESCRIPTION	DRIVE (MA)	SYNC	NEG
157	VCC	VDD	-	-	-	-	-	-
158	CTS2	CTS2	-	I	UART Clear-To-Send	-	-	-
159	RTS2	RTS2	-	O	UART Request-To-Send	8	-	-
160	RXD2	RXD2	-	I	UART Receive Data	-	-	-
161	TXD2	TXD2	-	O	UART Transmit Data	8	-	-
162	GND	GND	-	-	-	-	-	-
163	CTS1	CTS1	-	I	UART Clear-To-Send	-	-	-
164	RTS1	RTS1	-	O	UART Request-To-Send	8	-	-
165	RXD1	RXD1	-	I	UART Receive Data	-	-	-
166	TXD1	TXD1	-	O	UART Transmit Data	8	-	-
167	VCC	VDD	-	-	-	-	-	-
168	EDGESEL	EDGESEL	-	I	SDRAM Bus Clock Edge Select	-	-	-
169	GND	GND	-	-	-	-	-	-
170	BCLKO	BCLKO	-	O	System Bus Clock	16	-	-
171	VCC	VDD	-	-	-	-	-	-
172	RSTO	XRSTO	-	O	Processor Reset Output	8	-	-
173	GND	GND	-	-	-	-	-	-
174	CLKIN	CLKIN	-	I	System Clock Input (Fin)	-	-	-
175	VCC	VDD	-	-	-	-	-	-
176	MTMOD0	MTMOD[0]	-	I	Motorola Test Mode bit	-	-	-
177	MTMOD1	MTMOD[1]	-	I	Motorola Test Mode bit	-	-	-
178	GND	PGND	-	-	Analog Ground for PLL	-	-	-
179	NC	-	-	-	Analog Test Point for PLL	-	-	-
180	VCC	PVDD	-	-	Analog Power for PLL	-	-	-
181	MTMOD2	MTMOD[2]	-	I	Motorola Test Mode bit	-	-	-
182	MTMOD3	MTMOD[3]	-	I	Motorola Test Mode bit	-	-	-
183	GND	GND	-	-	-	-	-	-
184	PSTCLK	PSTCLK	-	O	Processor Status Clock	8	-	-
185	VCC	VDD	-	-	-	-	-	-
186	DDATA0	DDATA[0]	-	O	Debug Data bit / Bist Data bit	8	-	-
187	DDATA1	DDATA[1]	-	O	Debug Data bit / Bist Data bit	8	-	-
188	GND	GND	-	-	-	-	-	-
189	DDATA2	DDATA[2]	-	O	Debug Data bit / Bist Data bit	8	-	-
190	DDATA3	DDATA[3]	-	O	Debug Data bit / Bist Data bit	8	-	-
191	VCC	VDD	-	-	-	-	-	-
192	PST0	PST[0]	-	O	Debug Processor Status / Bist Done	8	-	-
193	PST1	PST[1]	-	O	Debug Processor Status / Bist Fail	8	-	-
194	GND	GND	-	-	-	-	-	-
195	PST2	PST[2]	-	O	Debug Processor Status	8	-	-
196	PST3	PST[3]	-	O	Debug Processor Status	8	-	-
197	VCC	VDD	-	-	-	-	-	-
198	PP7	PP[7]	XTIP	I/O	Parallel Port bit / Xtip	8	-	-

	PIN NAME	FUNC PRIMARY	FUNC SECONDARY	FUNC DIR	DESCRIPTION	DRIVE (MA)	SYNC	NEG
199	PP6	PP[6]	DREQ[0]	I/O	Parallel Port bit / DMA Request	8	-	-
200	PP5	PP[5]	DREQ[1]	I/O	Parallel Port bit / DMA Request	8	-	-
201	GND	GND	-	-	-	-	-	-
202	PP4	PP[4]	TM[2]	I/O	Parallel Port bit / Transfer Modifier	8	-	-
203	PP3	PP[3]	TM[1]	I/O	Parallel Port bit / Transfer Modifier	8	-	-
204	PP2	PP[2]	TM[0]	I/O	Parallel Port bit / Transfer Modifier	8	-	-
205	VCC	VDD	-	-	-	-	-	-
206	PP1	PP[1]	TT[1]	I/O	Parallel Port bit / Transfer Type	8	-	-
207	PP0	PP[0]	TT[0]	I/O	Parallel Port bit / Transfer Type	8	-	-
208	GND	GND	-	-	-	-	-	-

APPENDIX A

REGISTER MEMORY MAP

The following lists several keynotes regarding the Register Memory Map table:

- **Bold** letters mark registers that are restricted to supervisor access. While in user mode, supervisor access registers can not be written. If a supervisor register is written to in user mode, the contents of the register will not be changed. If a supervisor register is read in user mode, the data from the register will be valid.
- Underlined letters mark registers which are status or event registers. In these registers, the SBC sets the bits and the users clear the registers. To clear a bit, the users must write a one to that bit location; writing a zero has no effect.
- Normal letters mark registers which have accesses controlled by the address space mask bits contained in the MBAR register (see next section).
- Addresses not assigned to a register and undefined register bits are reserved for future expansion. Write accesses to these reserved address spaces and reserved register bits have no effect.

Table A-1. Register Memory Map

ADDRESS	NAME	BYTE0	BYTE1	BYTE2	BYTE3
CPU + \$002	Cache Control Reg	CACR			
CPU + \$004	Access Control Reg 0	ACR0			
CPU + \$005	Access Control Reg 1	ACR1			
CPU + \$801	Vector Base Reg	VBR			
CPU + \$C04	RAM Base Address Reg	RAMBAR			
CPU + \$C0F	Module Base Address Reg	MBAR			
MBAR + \$000	System Control Reg	RSR	SYPGR	SWIVR	SWSR
MBAR + \$004	Pin Assignment Reg	PAR		IRQPAR	Reserved
MBAR + \$008	PLL Control Reg	PLLCR	Reserved		
MBAR + \$00C	Bus Master Control Reg	MPARK	Reserved		

REGISTER MEMORY MAP

Table A-1. Register Memory Map (Continued)

ADDRESS	NAME	BYTE0	BYTE1	BYTE2	BYTE3				
MBAR + \$010	-	Reserved							
MBAR + \$014	-								
MBAR + \$018	-								
MBAR + \$01C	-								
MBAR + \$020	-								
MBAR + \$024	-								
MBAR + \$028	-								
MBAR + \$02C	-								
MBAR + \$030	-								
MBAR + \$034	-								
MBAR + \$038	-								
MBAR + \$03C	-								
MBAR + \$040	Interrupt Pending Reg					IPR			
MBAR + \$044	Interrupt Mask Reg					IMR			
MBAR + \$048	Auto Vector Control Reg	Reserved			AVCR				
MBAR + \$04C	Interrupt Control Reg	ICR0	ICR1	ICR2	ICR3				
MBAR + \$050	Interrupt Control Reg	ICR4	ICR5	ICR6	ICR7				
MBAR + \$054	Interrupt Control Reg	ICR8	ICR9	ICR10	ICR11				
MBAR + \$058	-	Reserved							
MBAR + \$05C	-								
MBAR + \$060	-								
MBAR + \$064	-								
MBAR + \$068	-								
MBAR + \$06C	-								
MBAR + \$070	-								
MBAR + \$074	-								
MBAR + \$078	-								
MBAR + \$07C	-								
MBAR + \$080	Chip-Select Address Reg 0					CSAR0		Reserved	
MBAR + \$084	Chip-Select Mask Reg 0	CSMR0							
MBAR + \$088	Chip-Select Control Reg 0	Reserved		CSCR0					
MBAR + \$08C	Chip-Select Address Reg 1	CSAR1		Reserved					
MBAR + \$090	Chip-Select Mask Reg 1	CSMR1							
MBAR + \$094	Chip-Select Control Reg 1	Reserved		CSCR1					
MBAR + \$098	Chip-Select Base Address Reg	CSBAR	Reserved						
MBAR + \$09C	Chip-Select Mask Reg 2	CSMR2							
MBAR + \$0A0	Chip-Select Control Reg 2	Reserved		CSCR2					
MBAR + \$0A4	-	Reserved							
MBAR + \$0A8	Chip-Select Mask Reg 3	Reserved		CSMR3					
MBAR + \$0AC	Chip-Select Control Reg 3			CSCR3					
MBAR + \$0B0	-	Reserved							
MBAR + \$0B4	Chip-Select Mask Reg 4	Reserved		CSMR4					
MBAR + \$0B8	Chip-Select Control Reg 4			CSCR4					
MBAR + \$0BC	-	Reserved							

A, B

Table A-1. Register Memory Map (Continued)

ADDRESS	NAME	BYTE0	BYTE1	BYTE2	BYTE3				
MBAR + \$0C0	Chip-Select Mask Reg 5	Reserved		CSMR5					
MBAR + \$0C4	Chip-Select Control Reg 5			CSCR5					
MBAR + \$0C8	-	Reserved							
MBAR + \$0CC	Chip-Select Mask Reg 6	Reserved		CSMR6					
MBAR + \$0D0	Chip-Select Control Reg 6			CSCR6					
MBAR + \$0D4	-	Reserved							
MBAR + \$0D8	Chip-Select Mask Reg 7	Reserved		CSMR7					
MBAR + \$0DC	Chip-Select Control Reg 7			CSCR7					
MBAR + \$0E0	-	Reserved							
MBAR + \$0E4	-								
MBAR + \$0E8	-								
MBAR + \$0EC	-								
MBAR + \$0F0	-								
MBAR + \$0F4	-								
MBAR + \$0F8	-								
MBAR + \$0FC	-								
MBAR + \$100	DRAMC Control Register					DCR		Reserved	
MBAR + \$104	-					Reserved			
MBAR + \$108	DRAMC Address & Control 0	DACR0							
MBAR + \$10C	DRAMC Mask Reg 0	DMR0							
MBAR + \$110	DRAMC Address & Control 1	DACR1							
MBAR + \$114	DRAMC Mask Reg 1	DMR1							
MBAR + \$118	-	Reserved							
MBAR + \$11C	-								
MBAR + \$120	-								
MBAR + \$124	-								
MBAR + \$128	-								
MBAR + \$12C	-								
MBAR + \$130	-								
MBAR + \$134	-								
MBAR + \$138	-								
MBAR + \$13C	-								
MBAR + \$140	Timer Mode Reg 0	TMR0		Reserved					
MBAR + \$144	Timer Reference Reg 0	TRR0							
MBAR + \$148	Timer Capture Reg 0	TCR0							
MBAR + \$14C	Timer Counter 0	TCN0							
MBAR + \$150	Timer Event Reg 0	Reserved	TER0						

REGISTER MEMORY MAP

Table A-1. Register Memory Map (Continued)

ADDRESS	NAME	BYTE0	BYTE1	BYTE2	BYTE3
MBAR + \$154	-	Reserved			
MBAR + \$158	-				
MBAR + \$15C	-				
MBAR + \$160	-				
MBAR + \$164	-				
MBAR + \$168	-				
MBAR + \$16C	-				
MBAR + \$170	-				
MBAR + \$174	-				
MBAR + \$178	-				
MBAR + \$17C	-				
MBAR + \$180	Timer Mode Reg 1	TMR1		Reserved	
MBAR + \$184	Timer Reference Reg 1	TRR1			
MBAR + \$188	Timer Capture Reg 1	TCR1			
MBAR + \$18C	Timer Counter 1	TCN1			
MBAR + \$190	Timer Event Reg 1	Reserved	TER1		
MBAR + \$194	-	Reserved			
MBAR + \$198	-				
MBAR + \$19C	-				
MBAR + \$1A0	-				
MBAR + \$1A4	-				
MBAR + \$1A8	-				
MBAR + \$1AC	-				
MBAR + \$1B0	-				
MBAR + \$1B4	-				
MBAR + \$1B8	-				
MBAR + \$1BC	-				
MBAR + \$1C0	UART Mode Reg 0	UMR10/UMR20		Reserved	
MBAR + \$1C4	UART Status 0/Clock Select Reg 10	USR0/UCSR0			
MBAR + \$1C8	UART Command Reg 0	UCR0			
MBAR + \$1CC	UART Receive 0/Transmit Buffer 0	URB0/UTB0			
MBAR + \$1D0	UART Change 0/Aux Control Reg 0	UIPCR0/UACR0			
MBAR + \$1D4	UART Interrupt Status 0/Mask Reg 0	UISR0/UIMR0			
MBAR + \$1D8	UART Baud Rate Generator MSB's 0	UBG10			
MBAR + \$1DC	UART Baud Rate Generator LSB's 0	UBG20			
MBAR + \$1E0	-	Do Not Access			
MBAR + \$1E4	-				
MBAR + \$1E8	-				
MBAR + \$1EC	-				
MBAR + \$1F0	UART Interrupt Vector Reg 0	UIVR0			
MBAR + \$1F4	UART Input Port 0	UIP0			
MBAR + \$1F8	UART RTS Output Port 0	UOP10			
MBAR + \$1FC	UART Output Port 0	UOP00			

A, B

Table A-1. Register Memory Map (Continued)

ADDRESS	NAME	BYTE0	BYTE1	BYTE2	BYTE3
MBAR + \$200	UART Mode Reg 1	UMR11/UMR21	Reserved		
MBAR + \$204	UART Status 1/Clock Select Reg 1	USR1/UCSR1			
MBAR + \$208	UART Command Reg 1	UCR1			
MBAR + \$20C	UART Receive 1/Transmit Buffer 1	URB1/UTB1			
MBAR + \$210	UART Change 1/Aux Control Reg 1	UIPCR1/UACR1			
MBAR + \$214	UART Interrupt Status 1/Mask Reg 1	UISR1/UIMR1			
MBAR + \$218	UART Baud Rate Generator MSB's 1	UBG11			
MBAR + \$21C	UART Baud Rate Generator LSB's 1	UBG21			
MBAR + \$220	-	Do Not Access			
MBAR + \$224	-				
MBAR + \$228	-				
MBAR + \$22C	-				
MBAR + \$230	UART Interrupt Vector Reg 1	UIVR1			
MBAR + \$234	UART Input Port 1	UIP1			
MBAR + \$238	UART RTS Output Port 1	UOP11			
MBAR + \$23C	UART Output Port 1	UOP01			
MBAR + \$240	-	Reserved			
MBAR + \$244	Parallel Port Data Direction Reg	PADDR	Reserved		
MBAR + \$248	Parallel Port Data Reg	PADAT			
MBAR + \$24C	-	Reserved			
MBAR + \$250	-				
MBAR + \$254	-				
MBAR + \$258	-				
MBAR + \$25C	-				
MBAR + \$260	-				
MBAR + \$264	-				
MBAR + \$268	-				
MBAR + \$26C	-				
MBAR + \$270	-				
MBAR + \$274	-				
MBAR + \$278	-				
MBAR + \$27C	-				
MBAR + \$280	Mbus Address Reg	MADR	Reserved		
MBAR + \$284	Mbus Frequency Reg	MFDR			
MBAR + \$288	Mbus Control Reg	MBCR			
MBAR + \$28C	Mbus Status Reg	MBSR			
MBAR + \$290	Mbus Data Reg	MBDR			

REGISTER MEMORY MAP

Table A-1. Register Memory Map (Continued)

ADDRESS	NAME	BYTE0	BYTE1	BYTE2	BYTE3
MBAR + \$294	-				
MBAR + \$298	-				
MBAR + \$29C	-				
MBAR + \$2A0	-				
MBAR + \$2A4	-				
MBAR + \$2A8	-				
MBAR + \$2AC	-				
MBAR + \$2B0	-				
MBAR + \$2B4	-				
MBAR + \$2B8	-				
MBAR + \$2BC	-				
MBAR + \$2C0	-				
MBAR + \$2C4	-				
MBAR + \$2C8	-				
MBAR + \$2CC	-				
MBAR + \$2D0	-				
MBAR + \$2D4	-				
MBAR + \$2D8	-				
MBAR + \$2DC	-				
MBAR + \$2E0	-				
MBAR + \$2E4	-				
MBAR + \$2E8	-				
MBAR + \$2EC	-				
MBAR + \$2F0	-				
MBAR + \$2F4	-				
MBAR + \$2F8	-				
MBAR + \$2FC	-				
MBAR + \$300	DMA Source Add Reg 0	SAR0			
MBAR + \$304	DMA Destination Addr Reg 0	DAR0			
MBAR + \$308	DMA Control Reg 0	DCR0		Reserved	
MBAR + \$30C	DMA Byte Count Reg 0	BCR0			
MBAR + \$310	DMA Status Reg 0	DSR0	Reserved		
MBAR + \$314	DMA Vector Reg 0	DIVR0			
MBAR + \$318	-				
MBAR + \$31C	-				
MBAR + \$320	-				
MBAR + \$324	-				
MBAR + \$328	-				
MBAR + \$32C	-				
MBAR + \$330	-				
MBAR + \$334	-				
MBAR + \$338	-				
MBAR + \$33C	-				

A,B

Table A-1. Register Memory Map (Continued)

ADDRESS	NAME	BYTE0	BYTE1	BYTE2	BYTE3
MBAR + \$340	DMA Source Add Reg 1	SAR1			
MBAR + \$344	DMA Destination Addr Reg 1	DAR1			
MBAR + \$348	DMA Control Reg 1	DCR1		Reserved	
MBAR + \$34C	DMA Byte Count Reg 1	BCR1			
MBAR + \$350	DMA Statue Reg 1	DSR1	Reserved		
MBAR + \$354	DMA Vector Reg 1	DIVR1			
MBAR + \$358	-	Reserved			
MBAR + \$35C	-				
MBAR + \$360	-				
MBAR + \$364	-				
MBAR + \$368	-				
MBAR + \$36C	-				
MBAR + \$370	-				
MBAR + \$374	-				
MBAR + \$378	-				
MBAR + \$37C	-				
MBAR + \$380	DMA Source Add Reg 2	SAR2			
MBAR + \$384	DMA Destination Addr Reg 2	DAR2			
MBAR + \$388	DMA Control Reg 2	CR2		Reserved	
MBAR + \$38C	DMA Byte Count Reg 2	BCR2			
MBAR + \$390	DMA Statue Reg 2	DSR2	Reserved		
MBAR + \$394	DMA Vector Reg 2	DIVR2			
MBAR + \$398	-	Reserved			
MBAR + \$39C	-				
MBAR + \$3A0	-				
MBAR + \$3A4	-				
MBAR + \$3A8	-				
MBAR + \$3AC	-				
MBAR + \$3B0	-				
MBAR + \$3B4	-				
MBAR + \$3B8	-				
MBAR + \$3BC	-				
MBAR + \$3C0	DMA Source Add Reg 3	SAR3			
MBAR + \$3C4	DMA Destination Addr Reg 3	DAR3			
MBAR + \$3C8	DMA Control Reg 3	DCR3		Reserved	
MBAR + \$3CC	DMA Byte Count Reg 3	BCR3			
MBAR + \$3D0	DMA Statue Reg 3	DSR3	Reserved		
MBAR + \$3D4	DMA Vector Reg 3	DIVR3			

REGISTER MEMORY MAP

Table A-1. Register Memory Map (Continued)

ADDRESS	NAME	BYTE0	BYTE1	BYTE2	BYTE3
MBAR + \$3D8	-	Reserved			
MBAR + \$3DC	-				
MBAR + \$3E0	-				
MBAR + \$3E4	-				
MBAR + \$3E8	-				
MBAR + \$3EC	-				
MBAR + \$3F0	-				
MBAR + \$3F4	-				
MBAR + \$3F8	-				
MBAR + \$3FC	-				

A,B

APPENDIX B

MCF5307 MEMORY MAP SUMMARY

This table below is a summary chart of the entire memory map for the MCF5307.

Table B-1. MCF5307 User Programming Model

ADDRESS	NAME	WIDTH	DESCRIPTION	RESET VALUE	ACCESS
MBAR+\$000	RSR	8	RESET STATUS REGISTER	\$80 OR \$20	R/W
MBAR+\$001	SYPCR	8	SYSTEM PROTECTION CONTROL REGISTER	\$00	R/W
MBAR+\$002	SWIVR	8	SOFTWARE WATCHDOG INTERRUPT VECTOR REGISTER	\$0F	R/W
MBAR+\$003	SWSR	8	SOFTWARE WATCHDOG SERVICE REGISTER	uninitialized	W
MBAR+\$004	PAR	16	PIN ASSIGNMENT REGISTER	\$0000	R/W
MBAR+\$006	IRQPAR	8	INTERRUPT ASSIGNMENT REGISTER	\$00	R/W
MBAR+\$008	PLLCR	8	PLL CONTROL REGISTER	\$00	R/W
MBAR+\$00C	MARBCR	8	MARB CONTROL REGISTER	\$00	R/W
MBAR+\$040	IPR	32	INTERRUPT PENDING REGISTER	\$00000000	R/W
MBAR+\$044	IMR	32	INTERRUPT MASK REGISTER	\$0000FFFE	R/W
MBAR+\$04B	AVCR	8	AUTOVECTOR CONTROL REGISTER	\$00	R/W
MBAR+\$04C	ICR0	8	INTERRUPT CONTROL REGISTER 0	\$00	R/W
MBAR+\$04D	ICR1	8	INTERRUPT CONTROL REGISTER 1	\$00	R/W
MBAR+\$04E	ICR2	8	INTERRUPT CONTROL REGISTER 2	\$00	R/W
MBAR+\$04F	ICR3	8	INTERRUPT CONTROL REGISTER 3	\$00	R/W
MBAR+\$050	ICR4	8	INTERRUPT CONTROL REGISTER 4	\$00	R/W
MBAR+\$051	ICR5	8	INTERRUPT CONTROL REGISTER 5	\$00	R/W
MBAR+\$052	ICR6	8	INTERRUPT CONTROL REGISTER 6	\$00	R/W
MBAR+\$053	ICR7	8	INTERRUPT CONTROL REGISTER 7	\$00	R/W
MBAR+\$054	ICR8	8	INTERRUPT CONTROL REGISTER 8	\$00	R/W
MBAR+\$055	ICR9	8	INTERRUPT CONTROL REGISTER 9	\$00	R/W
MBAR+\$056	ICR10	8	INTERRUPT CONTROL REGISTER 10	\$00	R/W
MBAR+\$057	ICR11	8	INTERRUPT CONTROL REGISTER 11	\$00	R/W
MBAR+\$080	CSAR0	16	CHIP-SELECT 0 BASE ADDRESS REGISTER	uninitialized	R/W
MBAR+\$084	CSMR0	32	CHIP-SELECT 0 MASK REGISTER	uninitialized(except V=0)	R/W
MBAR+\$08A	CSCR0	16	CHIP-SELECT 0 CONTROL REGISTER	uninitialized (except BEM=BSTR=BSTW=0)	R/W
MBAR+\$08C	CSAR1	16	CHIP-SELECT 1 BASE ADDRESS REGISTER	uninitialized	R/W
MBAR+\$090	CSMR1	32	CHIP-SELECT 1 MASK REGISTER	uninitialized (except V=0)	R/W
MBAR+\$096	CSCR1	16	CHIP-SELECT 1 CONTROL REGISTER	uninitialized (except BEM=BSTR=BSTW=0)	R/W

A,B

Table B-1. MCF5307 User Programming Model

ADDRESS	NAME	WIDTH	DESCRIPTION	RESET VALUE	ACCESS
MBAR+\$098	CSBAR	8	CHIP-SELECT 2-7 BASE ADDRESS REGISTER	uninitialized	R/W
MBAR+\$09C	CSMR2	16	CHIP-SELECT 2 MASK REGISTER	uninitialized (except V=0)	R/W
MBAR+\$0A2	CSCR2	16	CHIP-SELECT 2 CONTROL REGISTER	uninitialized (except BEM=BSTR=BST W=0)	R/W
MBAR+\$0AA	CSMR3	16	CHIP-SELECT 3 MASK REGISTER	uninitialized (except V=0)	R/W
MBAR+\$0AE	CSCR3	16	CHIP-SELECT 3 CONTROL REGISTER	uninitialized (except BEM=BSTR=BST W=0)	R/W
MBAR+\$0B6	CSMR4	16	CHIP-SELECT 4 MASK REGISTER	uninitialized (except V=0)	R/W
MBAR+\$0BA	CSCR4	16	CHIP-SELECT 4 CONTROL REGISTER	uninitialized (except BEM=BSTR=BST W=0)	R/W
MBAR+\$0C2	CSMR5	16	CHIP-SELECT 5 MASK REGISTER	uninitialized (except V=0)	R/W
MBAR+\$0C6	CSCR5	16	CHIP-SELECT 5 CONTROL REGISTER	uninitialized (except BEM=BSTR=BST W=0)	R/W
MBAR+\$0CE	CSMR6	16	CHIP-SELECT 6 MASK REGISTER	uninitialized (except V=0)	R/W
MBAR+\$0D2	CSCR6	16	CHIP-SELECT 6 CONTROL REGISTER	uninitialized (except BEM=BSTR=BST W=0)	R/W
MBAR+\$0DA	CSMR7	16	CHIP-SELECT 7 MASK REGISTER	uninitialized (except V=0)	R/W
MBAR+\$0DE	CSCR7	16	CHIP-SELECT 7 CONTROL REGISTER	uninitialized (except BEM=BSTR=BST W=0)	R/W
MBAR+\$100	DCR	16	DRAMC CONTROL REGISTER	uninitialized (except [15] = 0)	R/W
MBAR+\$108	DACR0	32	DRAMC0 ADDRESS & CONTROL REGISTER	uninitialized (except [15] = 0)	R/W
MBAR+\$10C	DMR0	32	DRAMC0 MASK REGISTER	uninitialized (except [0] = 0)	R/W
MBAR+\$110	DACR1	32	DRAMC1 ADDRESS & CONTROL REGISTER	uninitialized (except [15] = 0)	R/W
MBAR+\$114	DMR1	32	DRAMC1 MASK REGISTER	uninitialized (except [0] = 0)	R/W
MBAR+\$140	TMR1	16	TIMER1 MODE REGISTER	\$0000	R/W
MBAR+\$144	TRR1	16	TIMER1 REFERENCE REGISTER	\$FFFF	R/W
MBAR+\$148	TCR1	16	TIMER1 CAPTURE REGISTER	\$0000	R
MBAR+\$14C	TCN1	16	TIMER1 COUNTER REGISTER	\$0000	R/W
MBAR+\$151	TER1	8	TIMER1 EVENT REGISTER	\$00	R/W
MBAR+\$180	TMR2	16	TIMER2 MODE REGISTER	\$0000	R/W
MBAR+\$184	TRR2	16	TIMER2 REFERENCE REGISTER	\$FFFF	R/W

A, B

Table B-1. MCF5307 User Programming Model

ADDRESS	NAME	WIDTH	DESCRIPTION	RESET VALUE	ACCESS
MBAR+\$188	TCR2	16	TIMER2 CAPTURE REGISTER	\$0000	R
MBAR+\$18C	TCN2	16	TIMER2 COUNTER RETGISTER	\$0000	R/W
MBAR+\$191	TER2	8	TIMER2 EVENT REGISTER	\$00	R/W
MBAR+\$1C0	UMR11	8	UART1 MODE REGISTER	\$00	R/W
MBAR+\$1C0	UMR21	8	UART1 MODE REGISTER	\$00	R/W
MBAR+\$1C4	USR1	8	UART1 STATUS REGISTER	\$00	R
MBAR+\$1C4	UCSR1	8	UART1 CLOCK SELECT REGISTER	\$DD	W
MBAR+\$1C8	UCR1	8	UART1 COMMAND REGISTER	\$00	W
MBAR+\$1CC	URB1	8	UART1 RECEIVE BUFFER REGISTER	\$FF	R
MBAR+\$1CC	UTB1	8	UART1 TRANSMIT BUFFER REGISTER	\$00	W
MBAR+\$1D0	UIPCR1	8	UART1 INPUT PORT CHANGE REGISTER	\$0F	R
MBAR+\$1D0	UACR1	8	UART1 AUCILARY CONTROL REGISTER	\$00	W
MBAR+\$1D4	UISR1	8	UART1 INTERRUPT STATUS REGISTER	\$00	R
MBAR+\$1D4	UIMR1	8	UART1 INTERRUPT MASK REGISTER	\$00	W
MBAR+\$1D8	UBG11	8	UART1 BUAD RATE PRESCALE (MSB) REGISTER	uninitialized	W
MBAR+\$1DC	UBG21	8	UART1 BUAD RATE PRESCALE (LSB) REGISTER	uninitialized	W
MBAR+\$1F0	UIVR1	8	UART1 INTERRUPT VECTOR REGISTER	\$0F	R/W
MBAR+\$1F4	UIP1	8	UART1 INTERRUPT PORT REGISTER	\$FF	R
MBAR+\$1F8	UOP11	8	UART1 OUTPUT PORT BIT SET REGISTER	[7:1]=undefined, [0] = \$0	W
MBAR+\$1FC	UOP01	8	UART1 OUTPUT PORT BIT RESET REGISTER	uninitialized	W
MBAR+\$200	UMR12	8	UART2 MODE REGISTER	\$00	R/W
MBAR+\$200	UMR22	8	UART2 MODE REGISTER	\$00	R/W
MBAR+\$204	USR2	8	UART2 STATUS REGISTER	\$00	R
MBAR+\$204	UCSR2	8	UART2 CLOCK SELECT REGISTER	\$DD	W
MBAR+\$208	UCR2	8	UART2 COMMAND REGISTER	\$00	W
MBAR+\$20C	URB2	8	UART2 RECEIVE BUFFER REGISTER	\$FF	R
MBAR+\$20C	UTB2	8	UART2 TRANSMIT BUFFER REGISTER	\$00	W
MBAR+\$210	UIPCR2	8	UART2 INPUT PORT CHANGE REGISTER	\$0F	R
MBAR+\$210	UACR2	8	UART2 AUCILARY CONTROL REGISTER	\$00	W
MBAR+\$214	UISR2	8	UART2 INTERRUPT STATUS REGISTER	\$00	R
MBAR+\$214	UIMR2	8	UART2 INTERRUPT MASK REGISTER	\$00	W
MBAR+\$218	UBG12	8	UART2 BUAD RATE PRESCALE (MSB) REGISTER	uninitialized	W
MBAR+\$21C	UBG22	8	UART2 BUAD RATE PRESCALE (LSB) REGISTER	uninitialized	W
MBAR+\$230	UIVR2	8	UART2 INTERRUPT VECTOR REGISTER	\$0F	R/W
MBAR+\$234	UIP2	8	UART2 INTERRUPT PORT REGISTER	\$FF	R
MBAR+\$238	UOP12	8	UART2 OUTPUT PORT BIT SET REGISTER	[7:1]=undefined, [0] = \$0	W
MBAR+\$23C	UOP02	8	UART2 OUTPUT PORT BIT RESET REGISTER	uninitialized	W
MBAR+\$244	PADDR	16	PARALLEL PORT DATA DIRECTION REGISTER	\$0000	R/W
MBAR+\$248	PADAT	16	PARALLEL PORT DATA REGISTER	\$0000	R/W
MBAR+\$280	MADR	8	MBUS ADDRESS REGISTER	\$00	R/W
MBAR+\$284	MFDR	8	MBUS FREQUENCY REGISTER	\$00	R/W

A,B

Table B-1. MCF5307 User Programming Model

ADDRESS	NAME	WIDTH	DESCRIPTION	RESET VALUE	ACCESS
MBAR+\$288	MBCR	8	MBUS CONTROL REGISTER	\$00	R/W
MBAR+\$28C	MBSR	8	MBUS STATUS REGISTER	\$00	R/W
MBAR+\$290	MBDR	8	MBUS DATA REGISTER	\$00	R/W
MBAR+\$300	SAR0	32	DMA SOURCE ADDRESS REGISTER 0	\$00000000	R/W
MBAR+\$304	DAR0	32	DMA DESTINATION ADDRESS REGISTER 0	\$00000000	R/W
MBAR+\$308	DCR0	16	DMA CONTROL REGISTER 0	\$0000	R/W
MBAR+\$30C	BCR0	16	DMA BYTE COUNT REGISTER 0	\$0000	R
MBAR+\$310	DSR0	8	DMA STATUS REGISTER 0	\$00	R
MBAR+\$314	DIVR0	8	DMA INTERRUPT VECTOR REGISTER 0	\$00	R/W
MBAR+\$340	SAR1	32	DMA SOURCE ADDRESS REGISTER 1	\$00000000	R/W
MBAR+\$344	DAR1	32	DMA DESTINATION ADDRESS REGISTER 1	\$00000000	R/W
MBAR+\$348	DCR1	16	DMA CONTROL REGISTER 1	\$0000	R/W
MBAR+\$34C	BCR1	16	DMA BYTE COUNT REGISTER 1	\$0000	R
MBAR+\$350	DSR1	8	DMA STATUS REGISTER 1	\$00	R
MBAR+\$354	DIVR1	8	DMA INTERRUPT VECTOR REGISTER 1	\$00	R/W
MBAR+\$380	SAR2	32	DMA SOURCE ADDRESS REGISTER 2	\$00000000	R/W
MBAR+\$384	DAR2	32	DMA DESTINATION ADDRESS REGISTER 2	\$00000000	R/W
MBAR+\$388	DCR2	16	DMA CONTROL REGISTER 2	\$0000	R/W
MBAR+\$38C	BCR2	16	DMA BYTE COUNT REGISTER 2	\$0000	R
MBAR+\$390	DSR2	8	DMA STATUS REGISTER 2	\$00	R
MBAR+\$394	DIVR2	8	DMA INTERRUPT VECTOR REGISTER 2	\$00	R/W
MBAR+\$3C0	SAR3	32	DMA SOURCE ADDRESS REGISTER 3	\$00000000	R/W
MBAR+\$3C4	DAR3	32	DMA DESTINATION ADDRESS REGISTER 3	\$00000000	R/W
MBAR+\$3C8	DCR3	16	DMA CONTROL REGISTER 3	\$0000	R/W
MBAR+\$3CC	BCR3	16	DMA BYTE COUNT REGISTER 3	\$0000	R
MBAR+\$3D0	DSR3	8	DMA STATUS REGISTER 3	\$00	R
MBAR+\$3D4	DIVR3	8	DMA INTERRUPT VECTOR REGISTER 3	\$00	R/W

Table B-2 is a summary chart of the entire internal CPU memory map for the MCF5307. These registers are addressed with the MOVEC command.

Table B-2. Summary Chart of MCF5307 Internal CPU Memory Map

ADDRESS	WIDTH	NAME	DESCRIPTION	RESET VALUE	ACCESS
CPU @ \$002	32	CACR	Cache Control Register	\$0000000	W
CPU @ \$004	32	ACR0	Access Control Register 0	\$0000000	W
CPU @ \$005	32	ACR1	Access Control Register 1	\$0000000	W
CPU @ \$801	32	VBR	Vector Base Register	\$0000000	W
CPU @ \$c04	32	RAMBAR	RAM Base Address Register	\$0000000	W
CPU @ \$c0f	32	MBAR	Module Base Address Register	\$0000000	W

Table B-2 provides the internal location for the base registers and the cache control registers for the MCF5307. Most compilers and assemblers recognize the register name (i.e. MBAR, CACR) and will automatically place the correct internal CPU address in the op-code. There is no need to define the registers at their respective CPU address locations. Below are some examples of how the programmer would use the MOVEC command for the some of the above listed registers:

```
;Example code:
;****SRAM SETUP****
    move.l    $00010000+1,DO
    movec     DO,RAMBAR           ;SRAM base = $10000 and set the V bit
                                   ;bits [5:1] set to 0

;****MBAR SETUP****
    move.l    $10000000,DO
    movec     DO,MBAR           ;Module base = $10000000 and set the V bit
                                   ;bits [4:1] set to 0

;****VBR SETUP****
    move.l    $00200000,DO
    movec     DO,VBR           ;Interrupt vector base = $200000
```


A,B

INDEX

A

accumulator (ACC), 3-11
addressing mode summary, 3-20

B

BDM, 2-14

BDM/JTAG signals

test clock (TCK), 2-14
test data input/development serial input (TDI/DSI), 2-14
test data output/development serial output (TDO/DSO), 2-15
test mode select/break point (TMS/BKPT), 2-14
test reset/development serial clock (TRST/DSCLK), 2-14

branch instruction execution times

BRA, Bcc instruction execution times (*table 3-16*), 3-30

bus arbitration

bus request, 2-7

bus arbitration signals

bus driven, 2-7
bus grant, 2-7

bus operation

accesses by matches in CS and DRAM control registers (*table 7-5*), 7-8
bus and control signals, 7-1
address bus, 7-2
address strobe (AS), 7-3
bus signal summary (*table 7-1*), 7-2
data bus (D[31:0]), 7-3
interrupt request (IRQ7, IRQ5, IRQ3, IRQ1), 7-4
read/write (R/W), 7-3
transfer acknowledge (TA), 7-3
transfer start (TS), 7-3
bus arbitration, 7-25
bus arbitration signals, 7-26
ColdFire bus signal summary (*table 7-7*), 7-26
bus driven (BD), 7-26
bus grant (BG), 7-26
bus request (BR), 7-26
alternate master ownership state, 7-39
three-wire bus arbitration protocol state diagram (*figure 7-23*), 7-38

three-wire bus arbitration with external to master control bit asserted (*figure 7-22*), 7-36
multiple external bus master arbitration protocol (three-wire mode), 7-33
three-wire bus arbitration protocol transition conditions (*table 7-10*), 7-38
three-wire implicit and explicit bus ownership (*figure 7-21*), 7-35
reset state, 7-39
two-master bus arbitration protocol (two-wire mode), 7-27
alternate master ownership state, 7-32
implicit ownership state, 7-33
two-wire arbitration protocol state diagram (*table 7-9*), 7-32
two-wire bus arbitration protocol state diagram (*figure 7-20*), 7-31
two-wire bus arbitration protocol transition conditions (*table 7-8*), 7-32
two-wire bus arbitration with bus request asserted (*figure 7-18*), 7-29
two-wire implicit and explicit bus ownership (*figure 7-19*), 7-30
two-wire mode bus arbitration interface (*figure 7-17*), 7-27

bus characteristics, 7-6

signal relationships to BCLKO (*figure 7-1*), 7-6

clock and reset signals, 7-5

CF-bus signal summary (*table 7-4*), 7-5
clock input (CLKIN), 7-5
reset (RSTI), 7-5
reset out (RSTO), 7-5
system bus clock output (BCLKO), 7-5

connections for external memory port sizes (*figure 7-2*), 7-7

data transfer operation, 7-6

data transfer operations

back-to-back bus cycles, 7-14
allowable line access patterns (*table 7-6*), 7-16
back-to-back bus cycles (*figure 7-9*), 7-15

fast termination cycles

read cycle with fast termination (*figure 7-7*), 7-12

- write cycle with fast termination
(*figure 7-8*), 7-13
- line bus cycles, 7-15
 - allowable line access patterns
(*table 7-6*), 7-16
 - line read bus cycles, 7-17
 - line read burst (2-1-1-1)
(*figure 7-10*), 7-17
 - line read burst (3-2-2-2)
(*figure 7-11*), 7-18
 - line read burst-inhibited with fast
termination (*figure 7-12*), 7-19
 - line write bus cycles, 7-20
 - line write burst (2-1-1-1)
(*figure 7-13*), 7-20
 - line write burst (4-2-2-2) with one
wait state (*figure 7-14*), 7-21
 - line write burst-inhibited
(*figure 7-15*), 7-22
- read cycle, 7-8
 - read cycle flowchart (*figure 7-3*), 7-8
- write cycle
 - write cycle flowchart (*figure 7-5*), 7-10
- features, 7-1
- interrupt exceptions
 - interrupt acknowledge cycle, 7-24
 - interrupt-acknowledge cycle flowchart
(*figure 7-16*), 7-25
 - level 7 interrupts, 7-24
- reset operation, 7-40
 - master reset, 7-40
 - master reset timing (*figure 7-24*), 7-41
 - software watchdog reset, 7-41
 - software watchdog reset timing
(*figure 7-25*), 7-42

C

cache

- access control registers, 5-7
- cache coherency, 5-12
- cache control register (CACR), 5-5
- cache management, 5-8
- cache operation, 5-2
- cache operation summary, 5-14
 - cache line state diagram (*figure 5-4*), 5-15
 - cache line state transitions (*table 5-1*), 5-15
- cache organization, 5-2
 - cache organization and line format
(*figure 5-2*), 5-2
- cache protocol, 5-11
 - read hit, 5-11
 - read miss, 5-11
 - write hit, 5-12

- write miss, 5-11
- caching modes, 5-9
 - cacheable accesses, 5-10
 - copyback mode, 5-10
 - writethrough mode, 5-10
 - cache-inhibited accesses, 5-10
 - copyback mode, 5-9
 - MCF5307 unified cache block diagram
(*figure 5-1*), 5-1
 - memory accesses for cache maintenance, 5-12
 - cache filling, 5-12
 - cache pushes, 5-13
 - push and store buffer bus operation, 5-14
 - push and store buffers, 5-13
 - writethrough mode, 5-9
 - caching operation (*figure 5-3*), 5-3
- cache-inhibited, 5-9
- cache mode, 5-8
- chip-select module**
 - chip-select operation, 9-4
 - chip-select module, 9-4
 - general chip-select operation, 9-5
 - 8-, 16-, and 32-bit port sizing, 9-6
 - accesses by matches in CS control
registers (*table 9-2*), 9-5
 - global chip-select operation, 9-6
 - chip-select signals, 9-1
 - output enable (OE), 9-1
 - introduction, 9-1
 - features, 9-1
 - programming model, 9-7
 - chip-select module registers, 9-9
 - chip-select address register (CASR0,
CSAR1, and CSBAR)
 - chip-select 2-7 decoding
(*table 9-6*), 9-10
 - chip-select address registers
(CSAR0, CSAR1, and CSBAR), 9-9
 - chip-select control register
(CSCR0-CSCR7), 9-13
 - chip-select mask register
(CSMR0-CSMR7), 9-10
 - code example, 9-15
 - chip-select registers memory map, 9-7
 - memory map of chip-select registers
(*table 9-5*), 9-8
 - timing diagrams, 9-16
 - chip-select module outputs timing diagram
(*figure 9-2*), 9-16
- chip-selects**, 2-9
 - output enable (OE), 2-10
- clock and reset signals**, 2-7
 - clock input, 2-7
 - frequency control PLL, 2-8
 - reset, 2-7

- reset out, 2-8
- system bus clock output, 2-8
- ColdFire core**
 - addressing mode summary, 3-20
 - effective addressing modes and categories (*table 3-5*), 3-21
 - branch instruction execution times, 3-30
 - enhancements, 3-1
 - bus clock at 1/2, 1/3, 1/4 processor clock, 3-2
 - change of flow acceleration, 3-5
 - clock generation block diagram (*figure 3-1*), 3-2
 - clock-doubled microprocessor core, 3-1
 - debug module enhancements, 3-7
 - enhanced pipeline, 3-2
 - hardware multiply/accumulate and divide execution units, 3-6
 - multiply/accumulate and divide block diagram (*figure 3-4*), 3-6
 - illegal opcode handling, 3-6
 - exception processing overview, 3-11
 - exception vector assignments (*table 3-1*), 3-13
 - exception stack frame definition, 3-13
 - exception stack frame form (*figure 3-7*), 3-13
 - fault status encodings (*table 3-3*), 3-14
 - format field encoding (*table 3-2*), 3-14
 - instruction set summary, 3-21
 - MOVE long execution times, 3-26
 - MOVE instruction execution times, 3-26
 - MAC MOVE long instruction execution time (*table 3-11*), 3-27
 - MOVE byte and word execution times (*table 3-9*), 3-26
 - notational conventions (*table 3-6*), 3-21
 - timing assumptions, 3-25
 - misaligned operand references (*table 3-8*), 3-26
 - instruction set summary (*table 3-7*), 3-23
 - integer data formats, 3-18
 - integer data formats (*table 3-4*), 3-18
 - miscellaneous instruction execution times, 3-29
 - miscellaneous instruction execution times (*table 3-14*), 3-29
 - organization of data in registers, 3-18
 - organization of integer data formats in memory, 3-19
 - memory operand addressing (*figure 3-10*), 3-20
 - organization of integer data formats in registers, 3-18
 - organization of integer data formats in registers (*figure 3-8*), 3-18
 - organization of integer data formats in registers (*figure 3-9*), 3-19
 - trace exception, 3-16
 - processor exceptions, 3-14
 - access error exception, 3-14
 - address error exception, 3-15
 - debug interrupt, 3-16
 - fault-on-fault halt, 3-17
 - illegal instruction exception, 3-15
 - interrupt exception, 3-17
 - privilege violation, 3-16
 - reset exception, 3-17
 - RTE and format error exceptions, 3-16
 - TRAP instruction exceptions, 3-17
 - programming model, 3-7
 - MAC programming model, 3-10
 - accumulator (ACC), 3-11
 - MAC status register (MACSR), 3-11
 - MAC unit user programming model (*figure 3-6*), 3-10
 - mask register (MASK), 3-11
 - supervisor programming model, 3-9
 - status register, 3-9
 - status register (illustration), 3-10
 - vector base register (VBR), 3-10
 - supervisor programming model (illustration), 3-9
 - user programming mode, 3-7
 - user programming model
 - address registers (A0-A6), 3-7
 - condition code register (CCR), 3-8
 - data registers (D0-D7), 3-7
 - program counter, 3-8
 - stack pointer (A7), 3-8
 - user programming model (*figure 3-5*), 3-8
 - standard one operand instruction execution times, 3-27
 - one operand instruction execution times (*table 3-12*), 3-27
 - standard two operand instruction execution times, 3-28
 - two operand instruction execution times (*table 3-13*), 3-28
- ColdFire2M**, 3-10
- CPUSH instruction, 5-13

D

debug support, 16-1

- background debug mode (BDM), 16-6
 - BDM command set 16-10
 - BDM command set summary 16-10
 - command set descriptions, 16-14

- dump memory block (DUMP), 16-19
- fill memory block (FILL), 16-21
- no operation (NOP), 16-23
- read A/D register (RAREG/RDREG), 16-14
- read control register (RCREG), 16-24
 - control register map (table 16-5), 16-25
- read debug module register (RDMREG), 16-26
 - definition of DRc encoding - read (table 16-6), 16-27
- read memory location (READ), 16-16
- resume execution (GO), 16-23
- unassigned opcodes, 16-28
- synchronize PC to the PST/DDATA lines (SYNC_PC), 16-24
- write A/D register (WAREG/WDREG), 16-15
- write memory location (WRITE), 16-17
- write control register (WCREG), 16-26
- write debug module register (WDMREG), 16-27
 - definition of DRc encoding-write (table 16-7), 16-28
- ColdFire BDM commands, 16-11
 - BDM command summary (table 16-3), 16-11
 - BDM size field encoding (table 16-4), 16-12
- command sequence diagram (figure 16-6), 16-14
- BDM serial interface, 16-8
 - BDM serial transfer (figure 16-3), 16-8
 - receive packet format ,16-9
 - CPU-generated message encoding (figure 16-4), 16-9
 - receive BDM packet (figure 16-4), 16-9
 - transmit packet format, 16-9
 - transmit BDM packet (figure 16-5), 16-10
- CPU halt, 16-7
- processor/debug module interface, 16-1
- processor/debug module interface (figure 16-1), 16-1
- real-time debug support, 16-28
 - concurrent BDM and processor operation, 16-44
 - Motorola-recommended BDM pinout, 16-44
 - Motorola-recommended BDM pinout (figure 16-8), 16-45
 - programming model, 16-31
 - address attribute trigger register (AATR), 16-33
 - address breakpoint registers (ABLR, ABHR), 16-32
 - BDM address attribute (BAAR), 16-43
 - configuration/status register (CSR), 16-40
 - data breakpoint register (DBR, DBMR), 16-36
 - access size and operand data location (table 16-10), 16-37
 - debug programming model (figure 16-7), 16-32
 - program counter breakpoint register (PBR, PBMR), 16-35
 - trigger definition register (TDR), 16-37
 - theory of operation, 16-29
 - debug module hardware, 16-30
 - shared BDM/breakpoint hardware (table 16-9), 16-31
 - reuse of debug module hardware (Rev. A), 16-31
 - the new debug module hardware (Rev. B), 16-31
 - emulator mode, 16-30
- real-time trace support ,16-3
 - processor status signal encoding, 16-4
 - begin data transfer (PST=\$8-\$B), 16-5
 - begin execution of an instruction (PST=\$1), 16-4
 - begin execution of PULSE or WDDATA instructions (PST=\$4), 16-4
 - example PST/DDATA diagram (figure 16-2), 16-5
 - begin execution of RTE instruction (PST=\$7), 16-5
 - begin execution of taken branch (PST=\$5), 16-4
 - continue execution (PST=\$0), 16-4
 - emulator mode exception processing (PST=\$D), 16-6
 - entry into user mode (PST=\$3), 16-4
 - exception processing (PST = \$C), 16-6
 - processor halted (PST=\$F), 16-6
 - processor stopped (PST=\$E), 16-6
- signal description, 16-2
 - breakpoint (BKPT), 16-2
 - Rev A functionality, 16-2
 - Rev B enhancement, 16-2

- development serial clock (DSCLK), 16-2
 - development serial input (DSI), 16-2
 - development serial output (DSO), 16-2
 - processor status clock (PSTCLK), 16-2
 - processor status encoding (*table 16-1*), 16-3
 - debug and test signals**
 - high impedance (HIZ), 2-13
 - processor clock output (PSTCLK), 2-13
 - DMA controller**
 - overview, 1-6
 - DMA controller module, 13-1**
 - data transfer modes, 13-12
 - dual address transactions, 13-13
 - dual address writes, 13-13
 - dual-address transactions, 13-13
 - dual-address reads, 13-13
 - single address transactions, 13-13
 - DMA controller module functional description, 13-13
 - channel initialization and startup, 13-14
 - channel prioritization, 13-14
 - programming the DMA controller module, 13-14
 - channel termination, 13-17
 - error conditions, 13-17
 - interrupts, 13-17
 - data transfers, 13-15
 - auto-alignment, 13-16
 - bandwidth control, 13-17
 - external request operation, 13-15
 - external request timing
 - cycle-steal mode,
 - dual-address mode (*figure 13-7*), 13-16
 - external request timing—cycle-steal mode, single-address mode (*figure 13-6*), 13-16
 - DMA controller module programming model, 13-5
 - byte count register (BCR), 13-7
 - destination address register (DAR), 13-7
 - DMA control register, 13-8
 - BWC encoding (*table 13-2*), 13-9
 - DMA controller module channel offsets (*figure 13-4*), 13-5
 - DMA controller module register model per channel (*figure 13-5*), 13-5
 - DMA interrupt vector register, 13-12
 - DMA status register (DSR), 13-10
 - source address register (SAR), 13-6
 - DMA module overview, 13-3
 - dual-address transfer (*figure 13-3*), 13-5
 - single-address transfers (*figure 13-2*), 13-4
 - DMA signal description, 13-3
 - DMA signals (*table 13-1*), 13-3
 - introduction, 13-1
 - DMA signal diagram (*figure 13-1*), 13-2
 - transfer request generation, 13-12
 - continuous mode, 13-12
 - cycle-steal mode, 13-12
 - DRAM controller**
 - overview, 1-5
 - DRAM controller signals**
 - DRAM read/write (DRAMW), 2-10
 - synchronous DRAM clock enable (SCKE), 2-10
 - synchronous DRAM column address strobe (SCAS), 2-10
 - synchronous DRAM row address strobe (SRAS), 2-10
 - synchronous edge select (EDGESEL), 2-10
 - DREQ, 10-1, 10-2
 - dual address**
 - transfer, 13-5
 - DUART module**
 - overview, 1-6
- ## E
- electrical specs, 18-1**
 - clock timing specification (*table 18-4*), 18-3
 - DC electrical specs (*table 18-3*), 18-2
 - debug AC timing specification (*table 18-8*), 18-8
 - general-purpose I/O port AC timing specifications (*table 18-14*), 18-14
 - IEEE 1149.1 (JTAG) AC timing specification (*table 18-15*), 18-15
 - input AC timing specification (*table 18-6*), 18-5
 - maximum ratings (*table 18-1*), 18-1
 - M-bus input timing specifications between SCL and SDA (*table 18-11*), 18-11
 - M-bus output timing specifications between SCL and SDA (*table 18-12*), 18-11
 - M-bus timing specifications between BCLKO and SCL, SDA (*table 18-13*), 18-13
 - operating temperature (*table 18-2*), 18-1
 - output AC timing specification (*table 18-7*), 18-5
 - reset timing specification (*table 18-5*), 18-4
 - timer module AC timing specification (*table 18-9*), 18-9
 - UART module AC timing specifications (*table 18-10*), 18-10
 - enhancements**
 - enhanced pipeline
 - pipelined instruction sequence, 3-4
 - external bus master, 7-25

I

- instruction set summary, 3-21
- integer programming model, 3-9
- internal 4 Kbyte SRAM**
 - overview, 1-5
- interrupt control signals**
 - interrupt request, 2-7
- interrupt exception, 7-23
- interrupt masking, 7-23

J**JTAG**

- boundary-scan register, 17-7
- BYPASS instruction, 17-6
- CLAMP instruction, 17-6
- disabling the IEEE 1149.1 standard operation, 17-15
 - disabling JTAG in JTAG mode (*figure 17-3*), 17-15
- disabling the IEEE 1149.1 standard operation (*figure 17-4*), 17-16
- HIGHZ instruction, 17-6
- IDCODE instruction, 17-5
- IDcode register, 17-7
- JTAG register descriptions, 17-4
 - ID code register, 17-7
 - JTAG boundary-scan register, 17-7
 - boundary-scan bit definitions (*table 17-3*), 17-8
 - JTAG bypass register, 17-13
 - JTAG instruction shift register, 17-4
 - BYPASS instruction, 17-6
 - CLAMP instruction, 17-6
 - extext instruction, 17-5
 - HIGHZ instruction, 17-6
 - JTAG instructions (*table 17-2*), 17-5
 - sample/preload instruction, 17-5
 - JTAG instruction shift register (ID code), 17-5
- JTAG signal descriptions, 17-2
- JTAG pin descriptions (*table 17-1*), 17-3
- test data input/development serial input (TDI/DSI), 17-4
- test mode select/breakpoint (TMS/BKPT), 17-3
- test reset/development serial clock (TRST/DSCLK), 17-3
- test clock (TCK), 17-3
- obtaining the IEEE 1149.1 standard, 17-24
- overview, 17-2
 - JTAG test logic block diagram (*figure 17-1*), 17-2
- restrictions, 17-14

- SAMPLE/PRELOAD instruction, 17-5
- TAP controller, 17-13
 - JTAG TAP controller state machine (*figure 17-2*), 17-14
- JTAG MCF5307 BSDL file, 17-16

M**MAC programming model**

- Accumulator (ACC), 3-11
- MAC status register (MACSR), 3-11

M-Bus

- initialization sequence, 15-11
- protocol, 15-3
- slave address transmission, 15-3
- START signal, 15-3
- system configuration, 15-2
- M-Bus module, 15-1**
 - interface features, 15-1
 - M-Bus module block diagram (*figure 15-1*), 15-2
 - M-Bus programming examples, 15-11
 - arbitration lost, 15-14
 - flow chart of typical M-Bus interrupt routine (*figure 15-4*), 15-15
 - generation of repeated START, 15-14
 - generation of START, 15-11
 - generation of STOP, 15-13
 - post-transfer software response, 15-12
 - slave mode, 15-14
 - M-Bus protocol, 15-3
 - arbitration procedure, 15-4
 - clock stretching, 15-5
 - clock synchronization, 15-5
 - synchronized clock SCL (*figure 15-3*), 15-5
 - data transfer, 15-4
 - handshaking, 15-5
 - M-Bus standard communication protocol (*figure 15-2*), 15-3
 - repeated START signal, 15-4
 - slave address transmission, 15-3
 - M-Bus system configuration, 15-2
 - overview, 15-1
 - programming model, 15-6
 - M-Bus address register (MADR), 15-6
 - M-Bus control register (MBCR), 15-8
 - M-Bus data I/O register (MBDR), 15-11
 - M-Bus frequency divider register (MFDR), 15-6
 - M-Bus prescalar values (*table 15-2*), 15-7
 - M-Bus interface programmer's model (*table 15-1*), 15-6
 - M-Bus status register (MBSR), 15-9

M-Bus module signals

M-Bus serial clock (SCL), 2-12

M-Bus serial data (SDA), 2-12

MCF5307 bus signals

address bus, 2-3

address strobe, 2-4

read/write, 2-4

transfer acknowledge (TA), 2-5

transfer in progress, 2-5

transfer start, 2-4

MCF5307 introduction, 1-1

overview, 1-4

8 Kbyte unified cache, 1-5

ColdFire processor core, 1-4

DMA controller, 1-6

DRAM controller, 1-5

DUART module, 1-6

internal 4 Kbyte SRAM, 1-5

MCF5307 block diagram (*figure 1-1*), 1-4

Motorola bus (M-Bus) module, 1-6

multiply and accumulate (MAC) module, 1-5

system interface, 1-6

16-bit parallel-port interface, 1-7

chip-selects, 1-7

external bus interface, 1-6

interrupt controller, 1-7

JTAG, 1-7

system debug interface, 1-7

timer module, 1-6

mechanical data, 19-1

package, 19-1

pinout, 19-1

pinout tables (tables 19-1 through 19-4), 19-2

memory operand addressing diagram, 3-20

memory-to-memory transfer, 13-5

MFDR, 15-6

Motorola bus (M-Bus) module

overview, 1-6

MOVEC instruction, 5-8

multiply and accumulate (MAC) module

overview, 1-5

N

normal reset, 7-40

notational conventions, 3-21

O**organization of data in registers**

organization of integer data formats in memory, 3-19

P

PADAT, 10-4

PADDR, 10-3

PAR, 10-1

parallel port, 10-1

introduction, 10-1

parallel port operation, 10-3

port A data direction register (PADDR), 10-3

port A data register (PADAT), 10-4

signal descriptions

signals selected with PAR (*table 10-1*), 10-1

signals descriptions, 10-1

example code, 10-4

pin assignment register, 10-1

port A data direction register, 10-3

port A data register, 10-4

phase locked loop

PLL features, 4-1

block diagram of PLL module, 4-1

PLL operation, 4-2

normal mode, 4-2

phase locked loop control register

(PLLCR), 4-2

phase locked loop control register (PLLCR) (*figure 4-2*), 4-3

reduced-power mode, 4-2

reset/initialization, 4-2

PLL port list, 4-3

inputs, 4-3

BCLKO/PSTCLK divide ratios (*table*), 4-4CLKIN frequency (*table*), 4-4

outputs, 4-4

PLL specifications, 4-1

timing diagrams, 4-5

PSTCLK and BCLKO, 4-5

CLKIN, PSTCLK, and BCLK0 timing (*figure 4-3*), 4-5

RSTI timing, 4-5

reset and initialization timing diagram (*figure 4-4*), 4-6

pin assignment register, 10-1

port A data direction register, 10-3

port A data register, 10-4

processor exceptions

access error exception, 3-14

programming model

MAC programming model, 3-10

supervisor programming model, 3-9

R

registers

- MAC unit
 - ACC 3-11
 - MACSR 3-11

reset

- operation, 7-40
- RSTI pin, 7-41
- RSTI pin, 7-40

S

SDRAMC module

- asynchronous operation, 11-6
 - asynchronous memory map, 11-6
 - address and control registers, 11-10
 - address and control registers, (DACR0 and DACR1) 11-8
 - CAS encoding (*table 11-5*), 11-9
 - extended data out (EDO), 11-10
 - EDO encoding (*table 11-9*), 11-10
 - page mode (PM), 11-11
 - PM encoding (*table 11-11*), 11-11
 - port size (PS), 11-10
 - PS encoding (*table 11-10*), 11-10
 - RAS negate to CAS negate (RNCN), 11-9
 - RNCN encoding (*table 11-7*), 11-9
 - RAS precharge (PR) encoding (*table 11-6*), 11-9
 - RAS to CAS delay (RCD), 11-10
 - RCD encoding (*table 11-8*), 11-10
 - RE encoding (*table 11-4*), 11-8
 - refresh enable (RE), 11-8
- DRAM control register (DCR), 11-6
 - no address multiplexing (NAM), 11-7
 - synchronous operation (SO), 11-7
 - address modifier masks, 11-12
 - address modifier bit definitions (*table 11-14*), 11-12
 - base address mask encoding (*table 11-12*), 11-11
 - valid (V), 11-12
 - valid bit encoding (*table 11-16*), 11-13
 - WP encoding (*table 11-13*), 11-12
 - write protect (WP) 11-12

- DRAM controller memory map (*table 11-1*), 11-6
- asynchronous operation, 11-13
 - burst page-mode operation, 11-17
 - burst page-mode read operation (*figure 11-7*), 11-18
 - burst page-mode write operation (*figure 11-8*), 11-19
 - continuous page mode, 11-19
 - continuous page-mode operation (*figure 11-9*), 11-20
 - write hit in continuous page mode (*figure 11-10*), 11-21
 - extended data out (EDO) operation, 11-21
 - EDO read operation, 11-22
 - external master support, 11-23
 - general operation guidelines, 11-13
 - addressing muxing schemes (*table 11-17*), 11-13
 - DRAM addressing for 16-bit wide memories (*table 11-19*), 11-15
 - DRAM addressing for 32-bit wide memories (*table 11-20*), 11-16
 - DRAM addressing for byte-wide memories (*table 11-18*), 11-14
 - nonpage-mode operation, 11-16
 - basic nonpage-mode operation (RCD=0, RNCN=1, *figure 11-5*), 11-17
 - refresh operation, 11-22
 - DRAM access delayed by refresh (*figure 11-12*), 11-23
- block diagram, 11-1
 - address mux, 11-3
 - control logic and state machine, 11-3
 - DRAM controller block diagram bank register blocks (*figure 11-1*), 11-2
 - hit logic, 11-3
 - page hit logic, 11-3
 - refresh counter, 11-3
 - refresh register block, 11-3
- features, 11-1
- introduction, 11-1
- signal list, 11-3
 - synchronous DRAM clock enable (SCKE), 11-4
 - synchronous DRAM column address strobe (SCAS), 11-4
 - synchronous DRAM row address strobe (SRAS), 11-4
 - synchronous edge select (EDGESEL), 11-4
 - edge select output cell, 11-4
 - SDRAM interface (*figure 11-2*),

- 11-5
 - edge-select cell (*figure 11-3*), 11-5
 - edge-select cell output waveforms (*figure 11-4*), 11-6
- synchronous operation, 11-23
 - burst-page mode
 - burst write SDRAM access (*figure 11-15*), 11-37
 - synchronous memory map, 11-23
 - address and control registers (DACR0 and DACR1), 11-26
 - initiate mode register set command (IMRS), 11-27
 - IMRS bit encoding, (*table 11-28*) 11-27
 - initiate precharge all command (IP), 11-28
 - IP bit encoding (*table 11-31*), 11-29
 - page mode (PM), 11-29
 - synchronous PM encoding (*table 11-32*), 11-29
 - port size (PS) 11-28
 - PS encoding (*table 11-30*), 11-28
 - refresh enable (RE), 11-26
 - RE encoding (*table 11-26*), 11-26
 - DRAM control register (DCR), 11-23
 - COC bit encoding, 11-24
 - command on clock enable (COC), 11-24
 - initiate self-refresh command (IS), 11-25
 - IS bit encoding (*table 11-24*), 11-25
 - no address multiplexing (NAM), 11-24
 - NAM bit encoding (*table 11-22*), 11-24
 - refresh count (RC), 11-25
 - refresh timing (RTIM), 11-25
 - RTIM encoding (*table 11-25*), 11-25
 - SO bit encoding (*table 11-21*), 11-24
 - synchronous operation (SO), 11-24
 - DRAM controller mask registers (DMR0 and DMR1), 11-29
 - address multiplexing, 11-29
 - controller synchronous operation, 11-29
 - auto-refresh operation, 11-39
 - auto-refresh operation, (*figure 11-18*), 11-40
- burst-page mode, 11-35
 - burst read SDRAM access (*figure 11-14*), 11-36
- continuous-page mode, 11-38
 - synchronous continuous page mode access, read followed by read (*figure 11-16*), 11-38
 - synchronous continuous page-mode access, write followed by read (*figure 11-17*), 11-39
- self-refresh operation, 11-40
- self-refresh operation (*figure 11-19*), 11-41
- synchronous DRAM general operation guidelines, 11-29
 - address multiplexing, 11-30
 - address setup for 128K x 2 bank x 16-bit SDRAM x 1 as 16-bit port (*table 11-37*), 11-32
 - address setup for 1M x 4 bank x 16-bit SDRAM x 2 as 32-bit port (*table 11-38*), 11-32
 - address setup for 2M x 2 bank x 4-bit SDRAM x 4 as 16-bit port (*table 11-34*), 11-30
 - address setup for 2M x 2 bank x 4-bit SDRAM x 8 as 32-bit port (*table 11-35*), 11-31
 - address setup for 512K x 2 bank x 16-bit SDRAM x 1 as 16-bit port (*table 11-36*), 11-31
 - general details, 11-34
 - mode register settings, 11-33
 - mode register set commands (*figure 11-13*), 11-34
 - power-on sequence, 11-33
 - address setup for 2M x 2 bank x 4-bit SDRAM x 2 as 8-bit port (*table 11-33*), 11-30
- selection of CS0 automatic acknowledge (*table 9-4*), 9-7
- selection of CS0 port size (*table 9-3*), 9-7
- signal description, 2-1**
 - BDM/JTAG signals, 2-14
 - test clock (TCK), 2-14
 - test data input/development serial input (TDI/DSI), 2-14
 - test data output/development serial output (TDO/DSO), 2-15

- test mode select/breakpoint (TMS/BKPT), 2-14
 - test reset/development serial clock (TRST/DSCLK), 2-14
 - bus arbitration, 2-7
 - bus request, 2-7
 - bus arbitration signals, 2-7
 - bus driven, 2-7
 - bus grant, 2-7
 - chip-selects, 2-9
 - output enable (OE), 2-10
 - clock and reset signals, 2-7
 - clock input, 2-7
 - frequency control PLL, 2-8
 - CLKIN frequency (*table 2-7*), 2-8
 - reset, 2-7
 - reset out, 2-8
 - system bus clock output, 2-8
 - debug and test signals, 2-12
 - high impedance (HIZ), 2-13
 - processor clock output (PSTCLK), 2-13
 - DMA module signals, 2-11
 - DRAM controller signals
 - DRAM read/write (DRAMW), 2-10
 - synchronous DRAM clock enable (SCKE), 2-10
 - synchronous DRAM column address strobe (SCAS), 2-10
 - synchronous DRAM row address strobe (SRAS), 2-10
 - synchronous edge select (EDGESEL), 2-10
 - interrupt control signals, 2-7
 - interrupt request, 2-7
 - introduction, 2-1
 - MCF5307 block diagram (*figure 2-1*), 2-1
 - MCF5307 signal index (*table 2-1*), 2-2
 - M-Bus module signals, 2-12
 - M-Bus serial clock (SCL), 2-12
 - M-Bus serial data (SDA), 2-12
 - MCF5307 bus signals, 2-3
 - address bus, 2-3
 - address strobe, 2-4
 - read/write, 2-4
 - transfer acknowledge (TA), 2-5
 - transfer in progress, 2-5
 - transfer start, 2-4
 - SDRAM controller signals, 2-10
 - serial module signals, 2-11
 - timer module signals, 2-11
 - timer output (TOUT1, TOUT0), 2-12
 - SRAM 6-1**
 - SRAM features, 6-1
 - SRAM operation, 6-1
 - SRAM programming model, 6-1
 - power management, 6-4
 - examples of typical RAMBAR settings (*table 6-1*), 6-4
 - SRAM base address register (RAMBAR), 6-1
 - SRAM initialization, 6-3
 - SRAM initialization code, 6-4
 - STR Bit, 13-4
 - supervisor programming model**
 - status register, 3-9
 - system bus controller, 5-1
 - system debug interface**
 - overview, 1-7
 - system integration module**
 - introduction, 8-1
 - features, 8-1
 - programming model, 8-2
 - SIM registers memory map, 8-2
 - SIM memory map (*table 8-1*), 8-2
 - SIM programming and configuration, 8-2
 - module base address register (MBAR), 8-2
 - system interface**
 - overview, 1-6
 - system reset, 8-3
- T**
- three-wire mode, 7-33
 - timer, 2-11
 - timer module, 12-1**
 - module operation, 12-3
 - general purpose timer units
 - reference compare, 12-3
 - general-purpose timer units, 12-3
 - capture mode, 12-3
 - output mode, 12-3
 - prescaler, 12-3
 - overview, 1-6, 12-1
 - key features, 12-1
 - timer block diagram (*figure 12-1*), 12-2
 - programming model, 12-3
 - calculated timeout values (*table 12-2*), 12-7
 - general-purpose timer registers, 12-3
 - code example, 12-13
 - programming model for timers (*table 12-1*), 12-4
 - table of timeout values, 12-6
 - timer capture register (TCR), 12-5
 - timer counter (TCN), 12-6
 - timer mode register (TMR), 12-4
 - timer reference registers (TRR), 12-5
 - timer module, 12-1**
 - block diagram, 12-2
 - programming model
 - general-purpose timer registers
 - timer event register (TER), 12-6

timer module signals

timer output (TOUT1, TOUT0), 2-12
 TIP, 10-1, 10-2
 TM, 10-1, 10-3
 TMR1, TMR2, 12-4
 transparent translation registers, 5-7
 TRR1, TRR2, 12-5
 TT ,10-1, 10-3
 two-wire mode, 7-27

U**UART modules, 14-1**

operation, 14-6
 baud-rate generator logic, 14-6
 baud-rate generator/timer, 14-6
 bus operation, 14-17
 interrupt acknowledge cycles, 14-17
 read cycles, 14-17
 write cycles, 14-17
 calculating baud rates, 14-7
 external clock, 14-7
 system bus clock, 14-7
 looping modes, 14-13
 automatic echo mode, 14-13
 local loopback mode, 14- 13
 looping modes functional diagram
 (*figure 14-6*), 14-14
 remote loopback mode, 14-13
 multidrop mode, 14-15
 multidrop mode timing diagram
 (*figure 14-7*), 14-16
 transmitter and receiver operating modes,
 14-7
 receiver
 receiver timing diagram
 (*figure 14-5*), 14-11
 transmitter, 14-7
 register description and programming, 14-17
 interrupt mask register (UIMR), 14-31
 programming, 14-34
 I/O driver example, 14-35
 interrupt handling, 14-35
 UART module initialization, 14-35
 register description, 14-17
 auxiliary control register (UACR), 14-30
 clock-select register (UCSR), 14-24
 command register (UCR)
 RCx control bits (*table 14-10*),
 14- 28
 input port change register (UIPCR),
 14- 29
 input port register (UIP), 14-33
 interrupt vector register (UIVR), 14-33
 mode register 1 (UMR1), 14-18

B/CX control bits (*table 14-3*),
 14-20
 PMX and PT control bits
 (*table 14-2*), 14-20
 mode register 2 (UMR2), 14-20
 CMx control bits (*table 14-4*),
 14-21
 SBx control bits (*table 14-5*),
 14-22
 output port data registers (UPO1,
 UPO0), 14-34
 receiver buffer (URB), 14-28
 status register (USR), 14-22
 timer upper preload register (UBG1),
 14-32
 transmitter buffer (UTB), 14-29
 UART module programming model
 (*table 14-1*), 14-18
 timer upper preload register (UBG2),
 14-32
 auxiliary control register (UACR)
 timer mode and source select bits
 (*table 14-11*), 14-30
 serial module overview, 14-2

 comparison of UART module to MC68681,
 14-3
 interrupt control logic, 14-3
 serial communications channel, 14-2
 simplified block diagram (*figure 14-1*) , 14-1
 UART module initialization sequence, 14-35
 UART mode programming flowchart
 (*figure 14-8*), 14-36
 UART module signal definitions, 14-4
 clear to send (CTS), 14-4
 receiver serial data input (RxD), 14-4
 request to send (RTS), 14-4
 external and internal interface signals
 (*figure 14-2*), 14-5
 transmitter serial data output (TxD), 14-4

V

VBR, 3-10

Introduction	1
Signal Description	2
ColdFire Core	3
PLL	4
Cache	5
SRAM	6
Bus Operation	7
System Integration Module (SIM)	8
Chip-Select Module	9
Parallel Port (General-Purpose I/O)	10
DRAM Controller	11
Timer Module	12
DMA Controller	13
UART Module	14
M-Bus Module	15
Debug Module	16
IEEE 1149.1 JTAG	17
Electrical Characteristics	18
Mechanical Characteristics	19
Appendixes A and B	A,B

1	Introduction
2	Signal Description
3	ColdFire Core
4	PLL
5	Cache
6	SRAM
7	Bus Operation
8	System Integration Module
9	Chip-Select Module
10	Parallel Port (General-Purpose I/O)
11	DRAM Controller
12	Timer Module
13	DMA Module
14	UART Module
15	M-Bus Module
16	Debug Support
17	IEEE 1149.1 JTAG
18	Electrical Characteristics
19	Mechanical Characteristics
A,B	Appendixes A and B



How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution;
P.O. Box 5405; Denver, Colorado 80217, 1-800-441-2447 or 303-675-2140
Mfax™: RMFAX0@email.sps.mot.com-TOUCHTONE 602-244-6609
INTERNET: <http://Design-NET.com>

JAPAN: Nippon Motorola Ltd.; 4-32-1, Tatsumi-SPD-JLDC, 6F Salbu-Butsuryu-Center,
3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 81-3-3521-8315

ASIA-PACIFIC: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park,
51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

